Distrib. Comput. (2011) 23:331–340
DOI 10.1007/s00446-010-0121-5

An optimal bit complexity randomized distributed MIS algorithm

Y. Métivier · J. M. Robson · N. Saheb-Djahromi · A. Zemmari

Received: 9 July 2009 / Accepted: 10 November 2010 / Published online: 27 November 2010

Abstract We present a randomized distributed maximal independent set (MIS) algorithm for arbitrary graphs of size n that halts in time $O(\log n)$ with probability $1 - o(n^{-1})$, and only needs messages containing 1 bit. Thus, its bit complexity par channel is $O(\log n)$. We assume that the graph is anonymous: unique identities are not available to distinguish between the processes; we only assume that each vertex distinguishes between its neighbours by locally known channel names. Furthermore we do not assume that the size (or an upper bound on the size) of the graph is known. This algorithm is optimal (modulo a multiplicative constant) for the bit complexity and improves the best previous randomized distributed MIS algorithms (deduced from the randomized PRAM algorithm due to Luby (SIAM J. Comput. 15:1036–1053, 1986)) for general graphs which is $O(\log^2 n)$ per channel (it halts in time $O(\log n)$ and the size of each message is $\log n$). This result is based on a powerful and general technique for converting unrealistic exchanges of messages containing real numbers drawn at random on each vertex of a network into exchanges of bits. Then we consider a natural question: what is the impact of a vertex inclusion in the MIS on distant vertices? We prove that this impact vanishes rapidly as the distance grows for bounded-degree vertices and we provide a counter-example that shows this result does not hold in general. We prove also that these results remain valid for Luby's algorithm presented by Lynch (Distributed algorithms. Morgan Kaufman 1996) and by Wattenhofer (http://dcg.ethz.ch/lectures/fs08/distcomp/lecture/chapter4.pdf, 2007). This question remains open for the variant given by Peleg (Distributed computing—a locality-sensitive approach 2000).

This work was supported by grant No ANR-06-SETI-015-03 awarded by Agence Nationale de la Recherche.

An extended abstract of this paper was presented in the 16th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2009).

Y. Métivier · J. M. Robson · N. Saheb-Djahromi · A. Zemmari (✉)
Université de Bordeaux, LaBRI UMR CNRS 5800,
351 cours de la Libération, 33405 Talence, France
e-mail: zemmari@labri.fr

Y. Métivier
e-mail: metivier@labri.fr

J. M. Robson
e-mail: robson@labri.fr

N. Saheb-Djahromi
e-mail: saheb@labri.fr

1 Introduction

1.1 The problem

Let $G = (V, E)$ be a simple connected undirected graph. An independent set of G is a subset I of V such that no two members of I are adjacent. An independent set I is maximal if any vertex of G is in I or adjacent to a vertex of I.

In this paper we discuss how greedy selection for the computation of a maximal independent set (MIS) in a network of processors can be accomplished by the exchange of messages between adjacent processors. The distributed complexity (time and bit) of computing an MIS is of fundamental interest for the study and the analysis of distributed computing. The computation of an MIS is a building block for many distributed algorithms: topology control, routing, coloring. Thus an MIS provides an initial clustering which can be used as an initial structure. Furthermore an MIS induces a set of processors which can operate in parallel. A survey of results concerning the MIS problem can be found in [16] (Chap. 4, pp. 71–76) and in [19] (Chap. 8).

1.2 The model

The network: We consider the standard message passing model for distributed computing. The communication model consists of a point-to-point communication network described by a simple connected undirected graph $G = (V, E)$ where the vertices V represent network processors and the edges represent bidirectional communication channels. Processes communicate by message passing: a process sends a message to another by depositing the message in the corresponding channel. We assume the system to be synchronous and a synchronous wake-up of processors: processors have access to a global clock and all processors start the algorithm at the same time.

Time complexity: A round (cycle) of each processor is composed of the following three steps: 1. Send messages to (some of) the neighbours, 2. Receive messages from (some of) the neighbours, 3. Perform some local computation. As usual (see for example Peleg [19]) the time complexity is the maximum number of rounds needed until every node has completed its computation.

Bit complexity: As is explained by Santoro in [20] (Chap. 6) (see also [6], Chap. 3) the cost of a synchronous distributed algorithm is both time and bits. By definition, the bit complexity of a distributed algorithm (per channel) is the total number of bits exchanged (per channel) during its execution. As in [9], in each bit round each node can send/receive at most one bit to/from each neighbour. Finally the bit complexity of an algorithm $\mathcal{A}$ is the number of bit rounds to achieve algorithm $\mathcal{A}$. (This may be less than the per channel bit complexity because, in some channels, some bit rounds are "wasted" with no bits sent.) Thus it is considered as a finer measure of communication complexity and it has been studied for breaking and achieving symmetry for coloring in [3,5,9]. Dinitz et al. explain in [5] that it can be viewed as a natural extension of communication complexity (introduced by Yao [23]) to the analysis of tasks in a distributed setting. An introduction to this area can be found in Kushilevitz and Nisan [13].

Network and processes knowledge: The network is anonymous: unique identities are not available to distinguish the processes. We do not assume any global knowledge of the network, not even its size or an upper bound on its size. The processors do not require any position or distance information. Each processor knows from which channel it receives a message. An important fact (see [19] p. 93) due to the initial symmetry is: *there is no deterministic distributed algorithm for arbitrary anonymous graphs for computing an MIS assuming all vertices wake up simultaneously.*

1.3 Our contribution

As for a number of randomized algorithms for MIS, our algorithms work in the following way. They operate in synchronous rounds grouped into phases. At the end of each phase, some vertices join the MIS and some others know they will never be in the MIS: these two sets of vertices erase themselves from the graph.

One of the main contributions of this paper is the development of a powerful and general technique for converting unrealistic exchanges of messages containing real numbers drawn at random on each vertex into efficient exchanges of bits, drawn at random on each vertex, and we apply this technique to the computation of an MIS.

First we consider in Sect. 2 the model of exchange of messages containing real numbers and we deduce a very simple randomized algorithm (Algorithm $\mathcal{A}$) for the computation of an MIS. We derive logarithmic bounds on the number of exchanges required; one such bound on the average and another which holds with probability $1 - o(n^{-1})$; we deduce that Algorithm $\mathcal{A}$ computes an MIS for arbitrary graphs of size n in time $O(\log n)$ with probability $1 - o(n^{-1})$.

Then we discuss in Sect. 3 how, in the model of exchange of single bit messages, the real number exchanges can be simulated in finite time (Algorithm $\mathcal{B}$) and we show logarithmic bounds on the number of bits used to complete all the real exchanges and finally Algorithm $\mathcal{B}$ computes an MIS for arbitrary graphs of size n in time $O(\log^2 n)$ with probability $1 - o(n^{-1})$.

In Sect. 4 we show how the simulated real number exchanges can be overlapped (Algorithm $\mathcal{C}$), in this way we prove Theorem 3:

There exists a randomized distributed MIS algorithm for arbitrary graphs of size n that halts in time $O(\log n)$ with probability $1 - o(n^{-1})$, each message containing 1 bit.

We conclude that the bit complexity per channel of algorithm $\mathcal{C}$ is $O(\log n)$.

Algorithm $\mathcal{C}$ is optimal for the bit complexity as a direct consequence of two results. First, Kothapalli et al. show in [9] that if only one bit can be sent along each edge in a round, then every distributed vertex colouring algorithm (in which every node has the same initial state and initially only knows its own edges) needs at least $\Omega(\log n)$ rounds with high probability[1] (w.h.p. for short) to colour the cycle of size n with any finite number of colours. Second, Wattenhofer in [22] (p. 36) shows that a colouring algorithm can be obtained from an MIS algorithm. The colouring algorithm ([22] Algorithm 18) has the same (time and bit) complexity as the MIS algorithm, up to a multiplicative constant, for any family of graphs of bounded degree and in particular for the family of rings.

A fundamental question about distributed computation is: *What can (or cannot) be computed locally?* (see [11,18]). In the context of randomized distributed algorithms, we may consider a linked natural question: what is the impact of a vertex inclusion in the MIS on distant vertices?

[1] With high probability means with probability $1 - o(n^{-1})$.

We prove in Sect. 5 that this impact vanishes rapidly as the distance grows for bounded-degree vertices. We provide a counter-example that shows this result does not hold in general. We prove also that these results remain valid for Luby's algorithms presented by Lynch [16] and by Wattenhofer [22] and described in the next subsection. This question remains open for the variant given by Peleg [19].

1.4 Related works: comparisons and comments

The computation of an MIS has been the object of extensive research on parallel and distributed complexity [1,2,14,15]; Karp and Widgerson [8] have proved that the MIS problem is in NC. Some links with distributed graph coloring and some recent results on this problem can be found in [12]. The complexity of some special classes of graphs such as growth-bounded graphs is studied in [10]. Results have been obtained also for radio networks [17].

A major contribution is due to Luby [15]: he presented a randomized PRAM algorithm which requires a linear number of processors and runs in $O(\log^2 n)$ time. He assumes that the number of vertices is known. The main idea is to obtain for each vertex a *local total order* or a local election which breaks the local symmetry and then each vertex can decide locally whether it joins the MIS or not.

In the context of distributed computation, a first application of this idea corresponds to algorithm $\mathcal{A}$ of this paper and to the presentation of Luby's algorithm (called LubyMIS) given by Lynch [16] (Chap. 4, pp. 71–76). In this presentation, the knowledge of the size n of the network is used to enable each vertex to make a random choice of an integer from $\{0, \ldots, n^4\}$ using the uniform distribution. The analysis of this presentation is summarised by (Theorem 4.9, p. 76): *With probability one, LubyMIS eventually terminates. Moreover, the expected number of rounds until termination is* $O(\log n)$. Each message contains $O(\log n)$ bits thus the bit complexity per channel is $O(\log^2 n)$.

A variant of Luby's algorithm is presented in [19] (Chap. 8) which allows a simpler analysis. The local election is done in the following way: at each phase, each vertex v computes the maximal vertex degree of its 2-neighbourhood $D(v)$ (vertices at distance 1 or 2) and then draws uniformly at random a bit with probability depending on $D(v)$. If v draws 1 and every neighbour draws 0 then v joins the MIS. Peleg shows that this algorithm halts in time $O(\log^2 n)$ with probability $1 - o(n^{-1})$. It needs messages containing $\log n$ bits thus the bit complexity per channel is $O(\log^3 n)$.

Wattenhofer [22] presents and analyses another distributed implementation of Luby's algorithm. Each vertex needs at each phase the knowledge of the degrees of its neighbours, and marks itself with probability $1/(2d(v))$, where $d(v)$ is the current degree of v. If no higher degree neighbour of v is also marked then v joins the MIS. If a higher degree neighbour of v is marked v unmarks itself. If the neighbours have the same degree ties are broken arbitrarily. This algorithm terminates in expected $O(\log n)$ time and the size of messages is $\log n$, so its bit complexity per channel is $O(\log^2 n)$.

Alon et al. [1] presents a parallel randomized algorithm to find an MIS, its expected time on a PRAM is $O(\log n)$. As for the previous one, each vertex needs at each phase the degrees of its neighbours, so its bit complexity is $O(\log^2 n)$.

The analysis of previous algorithms has concentrated on the probability of a vertex being eliminated from the graph in any round. Our analysis, instead, concentrates on the probability of an edge being eliminated, and thereby reduces the complexity of both the algorithm and its analysis.

As is explained in [20] (p. 18), the knowledge of vertices in distributed computing is fundamental. For example, there exists a deterministic election algorithm for an anonymous network minimal for the covering relation (see [4]) if the size or a bound on the size is known and no such algorithm exists if the size (or a bound) is not known. In the same way, it is shown that it is possible to break the symmetry in anonymous networks but that it is not possible to detect termination unless the network size is known. About the computation of the network size one can cite the following impossibility result for anonymous networks. There exists no process-terminating algorithm for computing the size that is correct with probability $r > 0$. ([21], Chap. 9).

The table below summarises the comparison between the various MIS algorithms and Algorithm $\mathcal{C}$ of this paper.

	Knowledge	Time	Message size (number of bits)	Bit complexity (per channel)
Luby (Lynch)	Size of the graph	$O(\log n)$	$\log n$	$O(\log^2 n)$
Luby (Peleg)	Maximum degree in 2-Neighbourhood	$O(\log^2 n)$	$\log n$	$O(\log^3 n)$
Luby (Wattenhofer)	Maximum of neighbours degrees	$O(\log n)$	$\log n$	$O(\log^2 n)$
Alon et al.	Maximum of neighbours degrees	$O(\log n)$	$\log n$	$O(\log^2 n)$
Algorithm $\mathcal{C}$	no knowledge	$O(\log n)$	1	$O(\log n)$

Remark 1 In the column "Knowledge"of this table:

- the size of the graph is the size of the initial graph,
- maximum neighbours degree and maximum degree in 2−neighbourhoud are computed at each phase.

Notation In this paper, as usual, $Pr(e)$ denotes the probability of the event e and $E(X)$ the expected value of the random variable (r.v.) X.

2 Exchange of real numbers

The first distributed algorithm, denoted $\mathcal{A}$, is very simple. It is composed of phases. At each phase each processor u still in

the graph generates a random variable $x(u)$ and a processor is included in the independent set if its x is a local minimum, i.e., $x(u) < x(v)$ for each neighbour v of u. (Algorithm 1 describes a phase of Algorithm $\mathcal{A}$) It is convenient to take the random variables to be uniformly distributed on [0, 1).

When all local minima have been included in the independent set, each surviving processor (not included in the MIS and not definitely excluded from the MIS) generates a new random variable and again local minima are included in the independent set. The algorithm halts when there is no surviving processor.

The outcome of an MIS computation is defined on each vertex u by a special variable $\eta(u)$. A vertex u joining the MIS sets $\eta(u)$ to 1 and a vertex u not joining the MIS sets $\eta(u)$ to 0; initially $\eta(u) = -1$.

For each vertex v of the graph: *Not-In-MIS-set*(v) and *In-MIS-set*(v) are sets of vertices, initially *Not-In-MIS-set*$(v) = \emptyset$, *In-MIS-set*$(v) = \emptyset$.

Draw uniformly at random a real $x(u)$;
Send $x(u)$ to all neighbours w;
Receive $x(w)$ from all neighbours w;
4: **if** ($x(u) < x(w)$ for each neighbour of u) **then**
  Set $\eta(u) = 1$;
  Send *In-MIS* to each neighbour;
end if
8: Receive a message $mess(w)$ from all neighbours w;
Put w in *In-MIS-set*(u) for each neighbour w such that ($mess(w) =$ *In-MIS*);
if $mess(w) =$ *In-MIS* for at least one neighbour w **then**
  Set $\eta(u) = 0$;
12: Send *Not-In-MIS* to all neighbours;
end if
Receive a message $mess(w)$ from all neighbours w;
Put w in *Not-In-MIS-set*(u) for each neighbour w such that ($mess(w) =$ *Not-In-MIS*);
16: Erase from neighbours of u in the graph each vertex w in *In-MIS-set*(u) or in *Not-In-MIS-set*(u);
if ($\eta(u) \neq -1$ and neighbours$(u) = \emptyset$) **then**
  terminate;
end if

Algorithm 1: A phase of Algorithm $\mathcal{A}$

We have the following lemma:

Lemma 1 *In any phase, the expected number of edges removed from the remaining graph G is at least half the number of edges in G.*

Proof We say that a vertex u *preemptively removes* a neighbour v if $x(u)$ is less than $x(v)$ and $x(w)$ for all other neighbours w of u and v. If this is the case, then u will be included in the independent set and so v and all edges (v, w) incident on v will be removed from the graph. We say that the edges (v, w) are *preemptively removed*. If the degrees are $d(u)$ and $d(v)$, the probability that u preemptively removes v is at least $1/(d(u) + d(v))$. The average number of edges preemptively removed is thus at least $\left(\sum_{(u,v)\in E}\left(\frac{d(v)}{d(u)+d(v)} + \frac{d(u)}{d(u)+d(v)}\right)\right)/2$ since $d(v)$ edges are removed if u removes v and $d(u)$ edges are removed if v removes u and an edge (v, w) can only be preemptively removed twice, once by the removal of v and once by that of w. The sum is $\left(\sum_{(u,v)\in E} 1\right)/2$, that is half the number of edges. □

Remark 2 We introduce the new notion of "to be preemptively removed" for the proof of this lemma and thanks to this analysis, we avoid the need, present in previous algorithms, for a vertex to know the maximal degree of its 2-neighbourhood.

We then obtain:

Corollary 1 *There are constants k_1 and K_1 such that for any graph $G = (V, E)$ of n vertices the number of phases to remove all edges from G is:*

1. *less than $k_1 \log n$ on average,*
2. *less than $K_1 \log n$ with probability $1 - o(n^{-1})$.*

Proof Using t he fact that $|E| < n^2$, the average follows easily by induction.

For the second claim, initially the number of edges is less than $n^2/2$. Therefore after r rounds the expected number of edges remaining is less than $n^2/2^{r+1}$. In particular after $4 \log n$ rounds it is less than $n^{-2}/2$ so that the probability that any edge remains is less than $n^{-2}/2$. □

These results are summarised by:

Theorem 1 *Algorithm $\mathcal{A}$ computes an MIS for arbitrary graphs of size n in time $O(\log n)$ with probability $1-o(n^{-1})$.*

In what follows we assume that the number of phases is in fact less than $K_1 \log n$.

3 Exchange of bits in phases

The main idea: In this section we present and analyse an algorithm which simulates exchanges of real numbers in [0, 1) by exchanges of the bits of their binary representation with the most significant bit first. Note that the binary representation $(x_1, x_2, \ldots)$ of a real number $x \in [0, 1)$ satisfies $x = \sum_{i\geq 1} x_i/2^i$.

3.1 Description of the algorithm

We consider a more realistic algorithm, denoted $\mathcal{B}$, in which processors exchange messages of finite size; in fact we consider only messages of a fixed finite set of types:

- one bit 0 or 1 of Data
- *In-MIS*

– *Not-In-MIS*
– *Ineligible* (when a vertex v sends this message it means that until the end of the current phase v cannot be in the MIS).

At the start of a phase, a processor knows which neighbours are still in the graph and it initialises its set of *active* processors to all these neighbours. The status of a vertex is *Eligible* (the vertex may be included in the independent set during this phase), or *Ineligible* (the vertex cannot be included in the independent set during this phase). All processors still in the graph are initially *Eligible.*

One phase of exchange of real numbers is replaced by a phase composed of a sequence of rounds. Each round is composed of send, receive and internal actions. A message is one of these types. The *Data* messages send the bits of a real number, most significant first. The other messages permit a processor to stop sending *Data* when the real numbers are known accurately enough for further bits to be irrelevant to whether any processor has a local minimum (the exact simulation of the order induced by real numbers implies that during the course of a phase, two neighbours exchange bits as long as they have not determined whether one of them is or is not a local minimum even if other information implies that one of them is no longer eligible).

Initially, for each vertex v of the graph: $\eta(v) = -1$; and *active-set*(v), *Not-In-MIS-set*(v) and *In-MIS-set*(v) are sets of vertices. At the beginning of each phase, for each vertex v such that $\eta(v) = -1$: *status*$(v) =$ *Eligible*, *active-set*(v) contains the set of neighbours w of v satisfying $\eta(w) = -1$, and *Not-In-MIS-set*$(v) = \emptyset$, *In-MIS-set*$(v) = \emptyset$ and *Ineligible-set*$(v) = \emptyset$.

In each round each processor u generates one random bit and sends it to each active neighbour v and then does the following:

– If its bit was 0 and it received 1 from each active neighbour, it is a local minimum. It sets $\eta(u)$ to 1 and it sends *In-MIS* to all neighbours and takes no further part in this or later phases.
– If its bit was different from that received from v, it removes v from its list of active neighbours for this phase (the symmetry is broken; now the order relation between the two vertices is known thus they do not need to exchange more bits).
– If its bit was 1 and it received 0 from a neighbour, it is not a local minimum. Its status, for this phase, becomes *Ineligible* and it sends *Ineligible* to every active neighbour and removes from its active list all ineligible neighbours.
– If it receives *In-MIS* from any neighbour, it sets $\eta(u)$ to 0 and it sends *Not-In-MIS* to all other neighbours. It takes no part in subsequent phases but continues to generate and send bits as long as it has active neighbours.
– If it receives *Not-In-MIS* from a neighbour v, it notes that v is not eligible and will be removed from the graph after this phase. If it is itself ineligible, it removes v from its active list.
– If it receives *Ineligible* from a neighbour v, it notes that v is not eligible in this phase and, if it is itself ineligible, removes v from its active list.
– If it is ineligible and has no eligible active neighbours it is the end of the current phase. It takes no part in this phase and is ready to start another phase if $\eta(u) = -1$.

while ($\eta(v) \neq 1$ and $\eta(v) \neq 0$ and *active-set*(v) is not empty) **do**
  Draw uniformly at random a bit $b(v)$;
  Send $b(v)$ to all active neighbours w;
4: Receive $b(w)$ from all active neighbours w;
  if $b(v) = 0$ **then**
    if all active neighbours w have drawn $b(w) = 1$ **then**
      Set $\eta(v) = 1$;
8: Send *In-MIS* to each neighbour;
    end if
  else
    if there is at least one active neighbour w which has drawn $b(w) = 0$ **then**
12: Set *status*$(v) =$ *Ineligible*;
      Send *Ineligible* to each active neighbour;
    end if
  end if
16: Receive a message *mess*(w) from all active neighbours w;
  Put w in *Ineligible-set*(v) for each neighbour w such that (*mess*$(w) =$ *Ineligible*);
  Put w in *In-MIS-set*(v) for each neighbour w such that (*mess*$(w) = In$-*MIS*);
  if *mess*$(w) = In$-*MIS* for at least one neighbour w **then**
20: Set $\eta(v) = 0$;
    Send *Not-In-MIS* to all neighbours;
  end if
  Receive a message *mess*(w) from all active neighbours w;
24: Put w in *Not-In-MIS-set*(v) for each neighbour w such that (*mess*$(w) =$ *Not-In-MIS*);
  if *status*$(v) =$ *Ineligible* **then**
    Remove w from *active-set*(v) for each neighbour w in *Not-In-MIS-set*(v) or in *In-MIS-set*(v) or in *Ineligible-set*(v) and for each neighbour w such that $b(v) \neq b(w)$;
  end if
28: **end while**
  Erase from neighbours of v in the graph each vertex w in *In-MIS-set*(v) or in *Not-In-MIS-set*(v);

Algorithm 2: A phase of Algorithm $\mathcal{B}$

It follows from this:

Remark 3 After an exchange of bits and all consequent other messages, u and v consider each other as active neighbours if and only if they have generated exactly the same sequence of bits so far in this phase and one of them is still eligible.

This is precisely the condition that they still need to exchange more bits to decide whether one of them is a local minimum.

3.2 Analysis of the algorithm

In this section, we first analyze the time and the bit complexity of Algorithm $\mathcal{B}$. Then we analyse the bit complexity on each vertex.

Time and bit complexity: We have the following theorem:

Theorem 2 *Algorithm $\mathcal{B}$ constructs an MIS for any arbitrary graph of size $n \geq 1$ in $O\left(\log^2 n\right)$ exchanges of bits on average and with high probability.*

Proof Algorithm $\mathcal{B}$ simulates Algorithm $\mathcal{A}$. Thus, by Corollary 1, it needs $O(\log n)$ phases on average and with high probability to achieve the MIS construction. The main concern is to upper-bound the expected number of rounds needed to terminate each phase.

On the other hand, at each round, each edge has a probability $1/2$ to be removed from the graph, then, after $3 \log n$ exchanges of bits, there is no edge in the graph with probability $1 - o\left(n^{-2}\right)$. This ends the proof. □

Local bit complexity: In this section, we investigate the time for any vertex u to terminate a phase. We first prove the following lemma:

Lemma 2 *In any round of exchange of bits, any processor u has probability at least $1/4$ of entering one of the following states (if it has not already done so):*

- *In-MIS*
- *Not-In-MIS*
- *End-of-phase*.

Proof (In this proof we are concerned only with processors which are exchanging bits with their active neighbours; the graph considered is thus that in which edges represent the links between pairs of active neighbours and all graph terminology is with reference to this graph.)

We define a set C of *candidates* as the eligible processors in $\{u\} \cup N(u)$ and a set O of *opponents* as all neighbours, excluding u, of candidates. The behaviour of u during the exchange of bits depends on the bits chosen by u and $C \cup O$. We consider an assignment b of bits to $(C \cup O) \backslash \{u\}$ and argue that for at least half of such assignments, u has probability at least $1/2$ of entering one of the three states.

There are three types of *good* assignments in which u has this probability:

- every vertex v in C has a neighbour w in $C \cup O$ with $b(w) = 0$:
  With probability $1/2, b(u) = 1$: if u was eligible it becomes ineligible since it is in C; any vertex v in C with $b(v) = 0$ is removed from the active neighbour list of u; any vertex v in $C \backslash \{u\}$ with $b(v) = 1$ becomes ineligible because it has a neighbour w with $b(w) = 0$. Now u is ineligible and has no eligible active neighbours; it is in the state *End-of-phase*.
- u has no neighbour v with $b(v) = 0$:
  With probability $1/2$, $b(u) = 0 : u$ sends *In-MIS*.
- Some $v \in C \backslash \{u\}$ has all its neighbours w in $(C \cup O) \backslash \{u\}$ with $b(w) = 1$ and $b(v) = 0$:
  With probability $1/2$ $b(u) = 1 : v$ sends *In-MIS* to its neighbours including u and u sends *Not-In-MIS*.

The remaining *bad* assignments are those where some $v \in C \backslash \{u\}$ has all its neighbours w in $(C \cup O) \backslash \{u\}$ with $b(w) = 1$ and all such v have $b(v) = 1$. We argue that the set B of these bad assignments is at most half of all assignments.

Consider the function f from bad assignments to assignments defined by $f(b)(v) = 0$ where v is the first vertex v with the property stated and $f(b)(x) = b(x)$ otherwise. $f(b)$ is a good assignment because it has a vertex v with $b(v) = 0$ and $b(w) = 1$ for the neighbours of v (other than u); moreover $f(b)$ has no other vertex v' with this property so that f is one-one. Hence $|f(B)| = |B|$, $f(B) \cap B = \emptyset$ and so $|B| = (|f(B) \cup B|)/2$ and B is as claimed at most half of all assignments. □

Then:

Corollary 2 *There exist constants k_2 and K_2 such that the maximum number of Data bits generated by any processor u in all phases is:*

- *less than $k_2 \log n$ on average*
- *less than $K_2 \log n$ with probability $1 - o(n^{-2})$.*

Proof From Sect. 2 we know that u takes part on average in at most $k_1 \log n$ phases. From the lemma we deduce that u performs on average at most 4 bit exchanges per phase. Hence the average total number of bit exchanges is at most $4k_1 \log n$.

From Sect. 2 we can suppose that the total number of phases is at most $K_1 \log n$. For suitably large K_2, we have that a sequence of $K_2 \log n$ bit exchanges, each with probability at least $1/4$ of producing one of the messages of the lemma, will have produced $K_1 \log n$ messages with probability $1 - o(n^{-2})$; hence with probability $1 - o(n^{-2})$, after $K_2 \log n$ bit exchanges, u has finished every phase. □

Yielding:

Corollary 3 *There exists k_3 such that the maximum number of Data bits generated by any processor is less than $k_3 \log n$ on average and with probability $1 - o(n^{-1})$.*

Proof From Corollary 2 we have that with probability $1 - o(n^{-1})$ every processor has finished all phases after $K_2 \log n$ bit exchanges. If $f(n)$ is the worst case average number of bit exchanges, f must be monotonic non-decreasing in n and we have $f(n) \leq K_2 \log n + o(n^{-1}) f(n)$ implying $f(n) = O(\log n)$. □

From these results, we deduce another proof of Theorem 2.

Remark 4 As the system is synchronous, some vertices may need to wait to be synchronised. This explains why the bit complexity is $O(\log n^2)$ although only $O(\log n)$ bits are exchanged.

4 Exchange of bits with desynchronised phases between adjacent edges

We present a method of simulating Algorithm $\mathcal{B}$ by means of 1-bit messages sent along edges between neighbouring vertices, obtaining a new algorithm: Algorithm $\mathcal{C}$. The maximum number of messages sent and received on any edge will be $O(\log n)$ on average and with high probability.

The main idea: Algorithm $\mathcal{B}$ simulates exchange of real numbers by exchange of bits. The exchange of bits is centralised on the vertex: the vertex exchanges bits (corresponding to a real number) with neighbours until the symmetry is broken: at each round a vertex sends the same bit to all its neighbours. In this section, the main idea is, for each vertex u, a desynchronisation between edges incident on u: the vertex u exchanges bits with a given neighbour v until the symmetry is broken between u and v. If, in a round, u breaks the symmetry with v_1 and does not break the symmetry with v_2 then u considers that a phase with v_1 is completed and starts, in anticipation, a new phase with v_1 and it continues the previous phase with v_2. When a bit is drawn by anticipation, it is memorized to be used later by another edge when it accomplishes the same round in the same phase. Thus, in the same round, the vertex u may send the bit b_1 to the vertex v_1 corresponding to the phase t_1 and the bit b_2 to v_2 corresponding to the phase t_2 with $t_1 \neq t_2$.

Remark 5 The fundamental fact is that each round has probability $1/2$ of breaking the symmetry over an edge.

General description of the algorithm on each vertex: Each vertex runs two processes in interleaved fashion, alternating one round of process **calc_win** and one round of **calc_mis**. The process **calc_win** computes for each pair of neighbouring vertices and for each phase, which of the two has the smaller value in the phase. The process **calc_mis** uses this information to find the MIS computed by the algorithm and eventually to halt the first process.

From time to time a vertex running **calc_mis** will decide that it is to be removed from the graph and signal this fact to its neighbours. Any reference to the neighbours of a vertex is to be understood to mean those neighbours from whom no such signal has yet been received.

With the same notation defined previously, initially, for each vertex u of the graph: $\eta(u) = -1$; and *active-set*(v), *Not-In-MIS-set*(v) and *In-MIS-set*(v) are sets of vertices. The variable *active-set*(u) contains the set of neighbours v of u satisfying $\eta(v) = -1$.

while ($\eta(u) \neq 1$ and $\eta(u) \neq 0$ and *active-set*(u) is not empty) **do**
  1 round of **calc_win**;
  1 round of **calc_mis**
4: **end while**

Algorithm 3: Algorithm $\mathcal{C}$

4.1 Variables of algorithm $\mathcal{C}$

For each process u, the following variables are available:

- for each neighbour v of u, $phase_u(v)$ is a nonnegative integer, it is the number of the current phase between u and v; initially $phase_u(v)$ is equal to 1; by symmetry we have: $phase_u(v) = phase_v(u)$;
- for each neighbour v of u, $bit_u(v)$ is a nonnegative integer which denotes the number of the bit which will be sent by u to v in the current phase; initially, $bit_u(v)$ is equal to 1;
- for a phase t, $X_u[t, j]$ is the jth bit used in phase t, provided $j \leq l(t)$; otherwise it is undefined;
- for each neighbour v of u and for each number t of a phase $win_u(v)(t)$ is a boolean which will be true if in phase t the symmetry is broken for the edge between u and v and u has the smaller value.

Let u be a vertex and let v be a neighbour of u; let t be the number of a phase; we denote by $x_u(v, t)$ the word defined by the bits sent by u to v since the beginning of the phase t. The length of the phase t is denoted by $\ell(t)$, it is equal to $Max\{|x(v, t)| \mid v \text{ is a neighbour of } u\}$, where $|x(v, t)|$ is the length of the word $x(v, t)$, initially, $\ell(t) = 0$.

Let u be a vertex, the phase t is active if there exists an active neighbour v of u such that $t = phase_u(v)$ and $x_u(v, t) = x_v(u, t)$.

4.2 Computing the *win* bits: process **calc_win**

At the beginning of a round of **calc_win**, the process u draws uniformly at random a new bit $b(t)$ for each phase t having an active neighbour v with $bit_u(v) = l(t) + 1$ and puts it in X, i.e., $X[t, l(t) + 1] := b(t)$ and $l(t) := l(t) + 1$. For each neighbour v, u will find $b(v) = X[phase_u(v), bit_u(v)]$, send $b(v)$ to v and receive the bit $b(u)$ from v. If $b(v) = b(u)$ then it is necessary to look at succeeding bits to distinguish $x_u(v, phase_u(v))$ and $x_v(u, phase_v(u))$ of phase $phase_u(v) = phase_v(u)$ thus u does $bit_u(v) := bit_u(v) + 1$. Otherwise, the result is recorded and the next phase can be considered; thus u does:
(1) $win_u(v)(phase_u(v)) := (b(v) = 0)$; (2) $phase_u(v) := phase_u(v) + 1$; and (3) $bit_u(v) := 1$.

4.3 Computing the MIS: process **calc_mis**

For a vertex u, the computations for a phase t in which u is active take place in three stages. Initially u knows which neighbours v are active in the phase and it waits until it knows all its $win_u(v)(t)$ variables for them. Then it knows whether it is included in the MIS in the phase and sends an appropriate 1-bit *in* message to each v. In the second stage it waits until it has received an *in* message from each v. Then it knows whether it is excluded from the graph in the phase and sends an appropriate 1-bit *out* message to each v. In the final stage it waits until it has received an *out* message from each v. Then it knows which neighbours are active at the start of phase $t+1$. It updates $\eta(u)$ and its set of active neighbours and it is ready to start phase $t+1$ if it is still active.

4.4 Analysis of algorithm $\mathcal{C}$

Let T be the number of phases of Algorithm $\mathcal{A}$ (which equals the number of phases for algorithms $\mathcal{B}$ and $\mathcal{C}$). We know that on average and with high probability, T is at most $K_1 \log n$. We conclude that after a number $O(\log n)$ of rounds all values $win_u(v)(t)$ of all vertices with $t \leq T$ have been computed. More precisely, we have:

Lemma 3 *There exists a constant k such that, for any edge e, with probability $1 - o\left(n^{-3}\right)$, e has completed $K_1 \log n$ phases after $k \log n$ rounds, if it has not terminated before then.*

Proof Let $e = (u, v)$ be any edge and $X_e(t)$ the number of phases that would be completed by e in t rounds, if e continued bit exchanges even after the termination of algorithm $\mathcal{C}$. $X_e(t)$ is a binomial r.v. with parameters $1/2$ and t.

On the other hand, we have the following form of the Chernoff bound [7]:

$$\mathbb{P}r\left(X_e(t) \leq \mathbb{E}(X_e(t)) - a\right) \leq 2e^{-\frac{a^2}{2t}}, \quad \text{for any } a > 0.$$

We have $\mathbb{E}(X_e(t)) = \frac{t}{2}$, then, taking $t = 16K_1 \log n$ and $a = 7K_1 \log n$, we obtain:

$$\mathbb{P}r\left(X_e(t) \leq K_1 \log n\right) \leq 2e^{-\frac{49(K_1 \log n)^2}{32K_1 \log n}},$$

yielding

$$\mathbb{P}r\left(X_e(t) \leq K_1 \log n\right) = O\left(\frac{1}{n^3}\right).$$

Therefore, it suffices to set $k = 16K_1$. This ends the proof. □

Now we prove the main result of this section:

Theorem 3 *The randomized distributed MIS Algorithm $\mathcal{C}$ for arbitrary graphs of size n halts in time $O(\log n)$ with probability $1 - o(n^{-1})$, each message containing 1 bit.*

Proof By Lemma 3, the probability that after $16K_1 \log n$ rounds any edge e has not completed $K_1 \log n$ phases is $o(\frac{1}{n^3})$. Then the probability that it happens for some edge is $o(n^{-1})$. On the other hand, at any phase, the time for any vertex v to execute the **calc_mis** process is a constant. Therefore the probability that the algorithm has not terminated within $16K_1 \log n$ rounds at all vertices, either because the algorithm requires more than $K_1 \log n$ phases or because some edge has not completed this number of phases, is $o(n^{-1})$. This completes the proof. □

Finally:

Corollary 4 *The bit complexity per channel of Algorithm $\mathcal{C}$ is $O(\log n)$ with high probability and on average.*

Proof Theorem 3 shows immediately that the complexity is $O(\log n)$ with high probability. For the average, we note simply that the probability of exceeding $k \log n$ is $o(n^{-1})$, the remaining time is $O\left(\log^2 n\right)$ on average, since the average number of phases and the average number of bits per phase are $O(\log n)$ and, therefore the contribution of these cases to the average is $o\left(n^{-1} \log^2 n\right)$. □

5 Asymptotic independence of choices in MIS for distant vertices

5.1 Algorithm $\mathcal{C}$

Clearly the choice of a vertex inhibits that of the neighbouring ones and favours those at distance 2. A natural question would be: what is the impact of a vertex inclusion in the MIS on distant vertices? It is tempting to conjecture that the correlation vanishes rapidly as the distance grows. In the sequel we state this assertion for bounded-degree vertices.

We say a vertex u *survives* the kth phase if it remains neither chosen nor removed until the end of the kth phase. The following lemma is easily proved:

Lemma 4 *For a given vertex of degree deg the number of survival phases is dominated by a geometric r.v. of parameter $1/(deg+1)$.*

Proposition 1 *Let u and v be two vertices at distance l in G. We suppose that they have finite fixed degrees. Let $Pr(v|u)$ denote the probability that v is chosen conditioned by u having been chosen and $Pr(v)$ the probability of the same event without conditioning. Then, as $l \to \infty$, $Pr(v|u) = Pr(v) + O(\delta^l)$, for some δ, with $|\delta| < 1$.*

Proof According to the previous lemma, the probability that u survives l phases is at most $[1 - 1/(deg(u)+1)]^l$. On the other hand, the probability that v is chosen at one of the first $l/4$ phases is the *same*, *unconditioned* or *conditioned* by the

choice of u in one of these phases, since it depends on the behaviour of the algorithm in a ball of radius $l/2-1$ centered on v in these phases. Thus the difference between the conditional probability and the unconditional one is bounded by the probability of survival for one of the vertices. Let $deg = \max\{deg(u), deg(v)\}$. Setting $\delta = [1-1/(deg+1)]^{1/4}$, the proposition follows. □

Corollary 5 *The same bounding is true for $Pr(v|\overline{u})$, which is the probability that v is chosen subject to the condition that u is not: $Pr(v|\overline{u}) = Pr(v) + O(\delta^l)$.*

Proof Result of a standard computation, noting that $Pr(u) \geq 1/(deg(u)+1)$. □

Remark 6 It is easy to see that Proposition 1 holds under the weaker assumption that the degrees remain negligible with respect to the distance. However, the following example shows that the asymptotic independence of distant vertices *does not hold in general*.

Example 1 Consider the following graph G with two vertices u and v at distance $l = 4l'+1$: the vertices are $\{u_{i,j}, v_{i,j} | i = 0, \ldots, 2l',\ j = 1, \ldots, l^{3i}\}$ (u is $u_{0,1}$ and v is $v_{0,1}$); the edges are $(u_{i,j}, u_{i,k})$, $(v_{i,j}, v_{i,k})$, $(u_{i,j}, u_{i+1,k})$, $(v_{i,j}, v_{i+1,k})$ and $(u_{2l',j}, v_{2l',k})$ for every i, j, k for which these vertices exist.

We call the *normal* history of the algorithm on G that in which:

- in phase 1, only one vertex is chosen, namely a $u_{2l',j}$ or $v_{2l',j}$; (we consider the case that it is a $u_{2l',j}$). This eliminates from the graph all vertices $u_{2l',k}$, $v_{2l',k}$ and $u_{2l'-1,k}$ (and no others).
- in phase i, $(1 < i < l')$, two vertices are chosen, one $u_{2(l'-i),j}$ and one $v_{2(l'-i)+1,k}$. This eliminates from the graph all vertices $u_{2(l'-i),m}$, $u_{2(l'-i)-1,m}$, $v_{2(l'-i)+1,m}$ and $v_{2(l'-i),m}$ (and no others).
- in phase l', u and one vertex $v_{1,j}$ are chosen eliminating all other vertices.

In the first phase, it is impossible that both a $u_{2l',j}$ and a $v_{2l',k}$ are chosen because they are all neighbours. The possibility of any vertex $u_{i,j}$ or $v_{i,j}$ being chosen in any phase other than according to the normal history is less than $l^{-3(i+1)}$ provided the normal history has been followed in previous phases because all its neighbours in level $i+1$ are still present. Summing over all i and j and all phases we find that the probability of any behaviour other than the normal history is $O(l^{-1})$. If the normal history is followed, either u or v but not both is chosen and by symmetry, each has the same probability. Thus $Pr(v) = Pr(u) = 1/2 + O(l^{-1})$ but $Pr(u \text{ and } v) = O(l^{-1})$.

The authors do not know at present any weaker condition under which the asymptotic independence of choices holds for distant vertices.

Remark 7 It is also possible to extend Proposition 1 for sets of independent vertices. Let U_1 and U_2 be two nonempty sets of pairwise independent vertices of finite degrees. Let l denote the smallest distance between a member of U_1 and a member of U_2. Let us denote by $Pr(U_1)$ the probability that *all* members of U_1 are chosen in the MIS and by $Pr(U_1|U_2)$ the probability of the same event conditioned the event that *all* members of U_2 are. Then, as $l \to \infty$: $Pr(U_1|U_2) = Pr(U_1) + O(\delta^l)$, for some δ, with $|\delta| < 1$.

5.2 Luby's algorithms

The above result will remain valid for any algorithm which obeys two conditions:

1. The behaviour of the algorithm is *local*: in each phase the actions at a vertex depend only on the decisions within a ball of fixed radius and affect only vertices within this same ball.
2. Bounding of the probability of removal: a vertex of initial degree d has probability at least ϵ/d of being removed from the graph in each phase until its removal.

The second condition is called a lower bounding property. In this subsection, we prove that Luby's algorithms presented by Lynch and by Wattenhofer satisfy a lower bounding property and thus Proposition 1 remains valid.

Lemma 5 *The Luby (Lynch) algorithm satisfies a lower bounding property for $\epsilon = 1/3$.*

Proof Recall that vertex i generates a random integer in $[1,\ n^4]$ and joins the independent set if its integer is less than any of its neighbours'. i's integer is different from its neighbours' with probability at least $1 - d(i)/n^4$ and, if so, is the smallest in $\overline{N}(i)$ with probability at least $1/(1+d(i))$. Thus i joins the independent set with probability at least $(1-d(i)/n^4)/(1+d(i)) > 1/(1+d(i)) - 1/n^4 > 1/3d(i)$ since $d(i) < n$. □

Lemma 6 *The Luby (Wattenhofer) algorithm satisfies a lower bounding property for $\epsilon = 1/4$.*

Proof Recall that a vertex i of degree $d(i)$ generates a number 0 or 1 with probability $1/2d(i)$ of being equal to 1; i joins the independent set if it has generated 1 and none of its neighbours have done so, except possibly some neighbours with higher degree than i. Luby has shown [15] (Lemma B) that i is removed because a neighbour joins the independent set with probability at least $min(sum(i)/2, 1)/4$ where $sum(i) = \sum_{j \in N(i)} 1/d(j)$. We divide $N(i)$ into two parts $N_1(i)$ containing neighbours with degree $< d(i)$ and $N_2(i)$ all other neighbours. $sum(i) = sum_1(i) + sum_2(i)$ where the two sums are taken over N_1 and N_2.

If $sum_1(i) > 1$, Luby's lemma shows that i is removed with probability at least $1/8$.

If $sum(i) \leq 1, i$ is included in the independent set if it has generated a 1 and no vertex in N_1 has done so. This has probability $(\prod_{j\in N_1(i)}(1-1/2d(j)))/2d(i)$ which is at least $(1-\sum_{j\in N_1(i)} 1/2d(j))/2d(i)$ or $(1-sum_1(i)/2)/2d(i)$ which is at least $1/4d(i)$. □

Acknowledgments We are grateful to David Peleg and Roger Wattenhofer for helpful and enlightening e-discussions.

References

1. Alon, N., Babai, L., Itai, A.: A fast and simple randomized parallel algorithm for the maximal independent set. J. Algorithms **7**(4), 567–583 (1986)
2. Awerbuch, B., Goldberg, A.V., Luby, M., Plotkin, S.A.: Network decomposition and locality in distributed computation. In: Proceedings of the 30th ACM Symposium on FOCS, pp. 364–369. ACM Press (1989)
3. Bodlaender, H.L., Moran, S., Warmuth, M.K.: The distributed bit complexity of the ring: from the anonymous case to the non-anonymous case. Inf. Comput. **114**(2), 34–50 (1994)
4. Chalopin, J., Métivier, Y.: An efficient message passing election algorithm based on mazurkiewicz's algorithm. Fundam. Inform. **80**(1-3), 221–246 (2007)
5. Dinitz, Y., Moran, S., Rajsbaum, S.: Bit complexity of breaking and achieving symmetry in chains and rings. J. ACM. **55**(1) (2008)
6. Ghosh, S.: Distributed Systems—An Algorithmic Approach. CRC Press, Boca Raton (2006)
7. Habib, M., McDiarmid, C., Ramirez-Alfonsin, J., Reed, B. (eds.): Probabilistic Methods for Algorithmic Discrete Mathematics, of Algorithms and Combinatorics, vol. 16. Springer-Verlag, Berlin (1998)
8. Karp, R.M., Widgerson, A.: A fast parallel algorithm for the maximal independent set problem. In: Proceedings of the 16th ACM Symposium on Theory of Computing (STOC), pp. 266–272. ACM Press (1984)
9. Kothapalli, K., Onus, M., Scheideler, C., Schindelhauer, C.: Distributed coloring in $\tilde{O}(\sqrt{\log n})$ bit rounds. In: 20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings April 2006, Rhodes Island, Greece. IEEE pp. 25–29 (2006)
10. Kuhn, F., Moscibroda, T., Nieberg, T., Wattenhofer, R.: Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In: DISC, pp. 273–287 (2005)
11. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed locally! In: Proceedings of the 24 Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 300–309 (2004)
12. Kuhn, F., Wattenhofer, R.: On the complexity of distributed graph coloring. In: Proceedings of the 25 Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 7–15. ACM Press (2006)
13. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge, MA (1999)
14. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**, 193–201 (1992)
15. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM J. Comput. **15**, 1036–1053 (1986)
16. Lynch, N.A.: Distributed Algorithms. Morgan Kaufman (1996)
17. Moscibroda, T., Wattenhofer, R.: Coloring unstructured radio networks. Distrib. Comput. **21**(4), 271–284 (2008)
18. Naor, M., Stockmeyer, L.J.: What can be computed locally? SIAM J. Comput. **24**(6), 1259–1277 (1995)
19. Peleg, D.: Distributed computing—A Locality-sensitive approach. SIAM Monographs on Discrete Mathematics and Applications (2000)
20. Santoro, N.: Design and Analysis of Distributed Algorithms. Wiley, London (2007)
21. Tel, G.: Introduction to Distributed Algorithms. Cambridge University Press, Cambridge, MA (2000)
22. Wattenhofer, R.: http://dcg.ethz.ch/lectures/fs08/distcomp/lecture/chapter4.pdf. (2007)
23. Yao, A.C.: Some complexity questions related to distributed computing. In: Proceedings of the 11th ACM Symposium on Theory of Computing (STOC), pp. 209–213. ACM Press (1979)