Distrib. Comput. (2010) 22:349–361
DOI 10.1007/s00446-010-0097-1

An optimal maximal independent set algorithm for bounded-independence graphs

Johannes Schneider · Roger Wattenhofer

Received: 18 September 2008 / Accepted: 31 January 2010 / Published online: 10 March 2010

Abstract We present a novel distributed algorithm for the maximal independent set problem (This is an extended journal version of Schneider and Wattenhofer in Twenty-seventh annual ACM SIGACT-SIGOPS symposium on principles of distributed computing, 2008). On bounded-independence graphs our deterministic algorithm finishes in $O(\log^* n)$ time, n being the number of nodes. In light of Linial's $\Omega(\log^* n)$ lower bound our algorithm is asymptotically optimal. Furthermore, it solves the connected dominating set problem for unit disk graphs in $O(\log^* n)$ time, exponentially faster than the state-of-the-art algorithm. With a new extension our algorithm also computes a $\delta + 1$ coloring and a maximal matching in $O(\log^* n)$ time, where δ is the maximum degree of the graph.

Keywords Ad Hoc network · Sensor network · Radio network · Unit disk graph · Growth bounded graph · Bounded-independence graph · Local algorithm · Parallel algorithm · Maximal independent set · Maximal matching · Dominating set · Connected Dominating Set · Coloring · Symmetry breaking

1 Introduction

Minimum dominating sets (MDS) and connected dominating sets (CDS) are well-studied theoretical problems in wireless multi-hop networks, such as ad hoc, mesh, or sensor networks. In hundreds of papers they have been identified as key to efficient routing, media access control, or coverage, to just name three popular examples of usage. Consequently, the networking community has suggested a great number of algorithms towards computing CDS et al.; almost all of these algorithms are distributed, as wireless networks tend to be unreliable and dynamic, and conventional global algorithms seem too slow to cope with this constant churn. However, most algorithms are also heuristic in nature, and have been shown to perform poorly in efficacy and/or efficiency, when analyzed rigorously. However, there are exceptions; we will discuss them in detail in Sect. 2.

Given the huge impetus from the application side, recent research mostly concentrated on special graph classes that represent the geometric nature of wireless networks well. The classic theoretical model for wireless networks is the so-called unit disk graph (UDG) model, where the nodes are points in the plane, and two nodes are neighbors in the graph if and only if their Euclidean distance is at most 1. However, wireless radios will never transmit in perfect circles, and hence the UDG model has recently gotten a lot of stick. Instead the community started looking into generalized models, e.g. the quasi unit disk graph (QUDG) model, or the unit ball graph (UBG) model. In Sect. 3 we adopt the so-called bounded-independence graph (BIG) model (also called growth-bounded graph (GBG) model [14]); the BIG model only restricts the number of independent nodes in each neighborhood and is therefore a generalization of the UDG, QUDG and UBG models.

In Sect. 4 we present a novel distributed algorithm for the maximal independent set (MIS) problem. On bounded-independence graphs our algorithm finishes in $O(\log^* n)$ time, n being the number of nodes. As we will discuss in more detail in Sect. 2, our algorithm beats all existing algorithms for geometric models such as UDG or BIG by an exponential factor. Indeed, thanks to a lower bound argument by Linial

J. Schneider (✉) · R. Wattenhofer
Computer Engineering and Networks Laboratory,
ETH Zurich, 8092 Zurich, Switzerland
e-mail: jschneid@tik.ee.ethz.ch; schneider@tik.ee.ethz.ch

R. Wattenhofer
e-mail: wattenhofer@tik.ee.ethz.ch

[18], our algorithm is asymptotically optimal. In the BIG model a MIS is a constant approximation of a MDS, hence our algorithm gives the fastest constant MDS approximation. In Sect. 6 we will also quickly mention how to compute a CDS, how to obtain a polynomial time approximation scheme (PTAS), and an asymptotically optimal algorithm for computing a $\delta + 1$ coloring, as well as a distance two coloring.

2 Related work

Symmetry breaking is one of the main problems in distributed computing. Deterministic algorithms need a way to distinguish between nodes, i.e. IDs. In their pioneering work Cole and Vishkin [8] established the "deterministic coin tossing" method. They applied it to compute a MIS in a ring graph. For more than two decades their deterministic coin tossing technique has been a method of choice for breaking symmetries. Consequently it has been used in various algorithms [3,11,13–15]. In [8] each node v has a successor $s(v)$, which is a neighbor of v. Based on its own serial number r_v (initially, the ID of node v) and the one of its successor $r_{s(v)}$, it iteratively computes a new serial number r'_v, which is the least bit number i in which both serial numbers differ with the bit at position i in r_v appended. For example, if $r_v := 10110$ and $r_{s(v)} := 11110$, we have $i = 3 = 11$ (binary) for little endian notation with bit 3 of r_v being 0 and thus $r'_v := 110$. After $O(\log^* n)$ computation node v's color corresponds to its serial number. However, this only allows for a node to get a new serial number, which is distinct from its successor (but not necessarily from other neighbors). Thus it seems that the technique is limited to simple graphs of low degree such as rings or other constant-degree graphs. In contrast to Cole/Vishkin our technique extends to unbounded degree. Roughly speaking, in our algorithm a node does not monotonously stick to the same successor but takes into account the serial numbers of all nodes whenever a new serial number is computed based on the previous one. More precisely, a node's "successor" is the node with current minimum serial number. Furthermore, we allow a node to pause for a while, i.e. it does not compute a new serial number for a couple of rounds. Additionally, a node might restart the computation of a serial number, e.g., a new serial number based on its own ID and the IDs of its (non-pausing) neighbors is computed.

The MIS problem has also been studied in graphs beyond the ring (and other constant degree graphs). In sparse graphs, such as planar graphs, an algorithm based on Nash-Williams decompositions computes a MIS in a sublogarithmic number of communication rounds [6]. In general graphs the simple and elegant randomized algorithm by Luby [19] with running time of $O(\log n)$ (see as well [1,12]) outperforms the fastest known deterministic distributed algorithm [20] which is in $O\left(n^{\sqrt{c/\log n}}\right)$ with constant c. In Luby's algorithm neighbors try to enter the MIS based on their degrees, the deterministic algorithm uses network decompositions introduced in [4]. In general graphs every algorithm requires at least $\Omega\left(\sqrt{\log n/\log\log n}\right)$ or $\Omega(\log \Delta/\log\log \Delta)$ communication rounds for computing a MIS [16].

In this paper we concentrate on geometric graph classes that are relevant in wireless networking, breaking the lower bound for general graphs. For these geometric graphs the relevant lower bound is by Linial [18]. He showed that even on a ring topology at least time $\Omega(\log^* n)$ is required to compute a MIS. Only very recently (and probably stirred by the interest in wireless networking) MIS algorithms for geometric graphs (such as UDG or BIG) have been discovered. The currently fastest randomized algorithm by Gfeller and Vicari [10] runs in $O(\log\log n \cdot \log^* n)$ time. Using randomization a set S is created, where every node $v \in S$ has at most $O(\log^5 n)$ neighbors in S. Thereafter the fastest deterministic algorithm up to now [13] with time complexity $O(\log \Delta \cdot \log^* n)$ is used, which computes an $O(\log \Delta)$-ruling independent set in time $O(\log \Delta \cdot \log^* n)$ in the first phase. (For an α-ruling independent set, every two nodes in the set have distance at least two and any node not in the set has a node in the set within distance α.) The $O(\log \Delta)$-ruling independent set is transformed into an 3-ruling independent set, and this set in turn is taken to compute a MIS using again the deterministic coin tossing technique. Our algorithm is not only exponentially faster than the state-of-the-art [10], it is also deterministic, and last not least simpler. Thanks to Linial's lower bound we know that it is asymptotically optimal. If a node knows the distances to its neighbors, then [15] also achieves asymptotic optimality for computing a MIS in BIG. The main idea is to maintain a set of active nodes (initially, all nodes are active) and only consider edges between active nodes of distance at most r. In every iteration the radius r is doubled (up to 1/2) and a MIS is computed on all active nodes. Only nodes in the MIS stay active. The degrees in the considered graphs are small and thus a MIS can be computed efficiently (via a coloring). It is rather surprising that we can match the bound of [15] without any distance information.

3 Model and definitions

The communication network is modeled with a graph $G = (V, E)$. For a node v its neighborhood $N^r(v)$ represents all nodes within r hops of v (not including v itself). A set $T \subseteq V$ is said to be independent in G if no two nodes $u, v \in T$ are neighbors. A set $T \subseteq V$ is said to be independent in G if no two nodes $u, v \in T$ are neighbors. A set is (α, β)-ruling if every two nodes in the set have distance at least α and any node not in the set has a node in the set within distance β. A set $S \subseteq V$ is a maximal independent set (MIS), if it is (2,1)-ruling. A MIS S of maximum cardinality, i.e.

$|S| \geq \max_{\mathrm{MIS}\ T} |T|$, is called a maximum independent set (MaxIS). We consider bounded-independence graphs, which are defined as:

Definition 1 A graph $G = (V, E)$ is of bounded-independence if there is a bounding function $f(r)$ such that for each node $v \in V$, the size of a MaxIS in the neighborhood $N^r(v)$ is at most $f(r), \forall r \geq 0$. We say that G if of polynomially bounded-independence if $f(r)$ is a polynomial $p(r)$.

Equivalently, one might use the notion of a d-local α-doubling metric, i.e. a metric is d-local α-doubling if any ball of radius $r \leq d$ can be covered by at most α balls of radius $r/2$. A graph $G = (V, E)$ is of bounded-independence if it obeys the d-local $f(d)$-doubling metric for some function f.

In particular, this means that for a constant c the value $f(c)$ is also a constant. A subclass of bounded-independence graphs are quasi unit disk graphs and unit disk graphs, which are often used to model wireless communication networks and have $f(r) \in \mathrm{O}(r^2)$. However, we do not require the graph to be of polynomially bounded-independence, e.g., $f(r)$ might as well be exponential in r.

Our algorithm is uniform, i.e. it does not require any knowledge of the total number of nodes n. Communication among nodes is done in synchronous rounds without collisions, i.e. each node can exchange one distinct message of size $\mathrm{O}(\log n)$ bits with each neighbor. We understand that such a powerful communication layer is unrealistic in many application domains; in wireless networks for instance transmission collisions will happen, and must be addressed. We will discuss this in detail in the conclusions.

Definition 2 The function $\log^*()$ is defined recursively as follows $\log^* 0 = \log^* 1 = \log^* 2 = 0$ and $\log^* n = 1 + \log^* \lceil \log n \rceil$ for $n > 2$.

Expressed differently, $\log^* n$ describes how often one has to take the logarithm to end up with at most 2. We denote by $\log^{(j)} n$ the binary logarithm taken j times recursively. Thus $\log^{(1)} n = \log n$, $\log^{(2)} n = \log\log n$, etc.

Every node has a unique ID represented by l bits, where l is upper bounded by $\log n$. An ID and all other binary numbers are in little endian notation and have the form: $x^l, x^{l-1}, \ldots, x^1$, where $x^i \in \{0, 1\}$. Observe that for technical reasons the low order bit has index 1 (not 0).

4 MIS Algorithm

Let us start by giving an informal description of our deterministic MIS algorithm. Each node performs a series of competitions against neighbors, such that more and more nodes drop out until only nodes joining the MIS remain. For all nodes not in the MIS or not adjacent to a node in the MIS, the process is repeated.

To get a deeper understanding of a competition we take a closer look at the very first competition. A node v competes against the neighbor u with minimum ID. If ID_u is larger than ID_v, i.e. node v has the smallest ID among all neighbors, the result is 0. If ID_v is not the smallest of all neighbors, the result of the competition is the maximum position for which v's ID has a bit equal to 1 and u's ID has a bit equal to 0. For ID_v being 11101 and ID_u being 10001, the two differing positions are 3 and 4 and thus the result of the competition for v is 4, i.e. $r_v = 4$.

The result r_v of the first competition forms the basis for the next competition, the result of that competition in turn is used for the following competition and so forth.

A node can be in one of five states, which it might alter after each competition (see Algorithm Update State). Initially each node is a competitor. If the result of the competition for node v is (strictly) smaller than that of all its competing neighbors, node v becomes a dominator and joins the MIS. All adjacent nodes of a dominator become dominated. Both dominators and dominated nodes are not involved in further competitions. In case the result of a node is as small as that of all its neighbors and at least one neighbor has the same result, the node becomes a ruler. A neighbor of a ruler gets ruled (if not dominated). A ruler immediately ends the current phase of becoming a dominator (lines 4 to 14 in Algorithm MIS), becomes a competitor again and proceeds to the next phase. After a node has advanced to the cth phase (where c is some number depending on the function $f()$—see Definition 1) by changing back and forth between competitor and ruler only, i.e. without ever being ruled, it must be dominated or a dominator. A ruled node, ends the phase and stays quiet until all neighbors are ruled (or dominated). Then it starts the algorithm again as competitor, i.e. it is again in its first phase (of becoming a dominator). During a phase a node executes at most $\log^* n + 2$ recursive competitions (as will be shown in Sect. 5.1) and every competing node must change its state. In the subsequent competition (the first one of a new phase) all competitors compete again by using IDs. For some more examples including updates of states consider Fig. 1.

Next, we give a more formal definition of a competition to clarify our notation. Let r_v^j denote the result of the jth (recursive) competition for node v. The first competition is always based on the IDs. Thus we define $r_v^0 := ID_v$. Any number r_v^{j-1} consists of l bits ($l \leq \log^{(j)} n$ as shown in Lemma 2) and has the form $r_v^{j-1} = y_v^t, y_v^{t-1}, \ldots, y_v^1$. A competitor only competes against nodes that are also competitors, i.e. the results of ruled or dominated nodes are not considered. In order to perform the jth (recursive) competition with $j \geq 1$ node v chooses a competitor $u \in N(v)$, s.t.

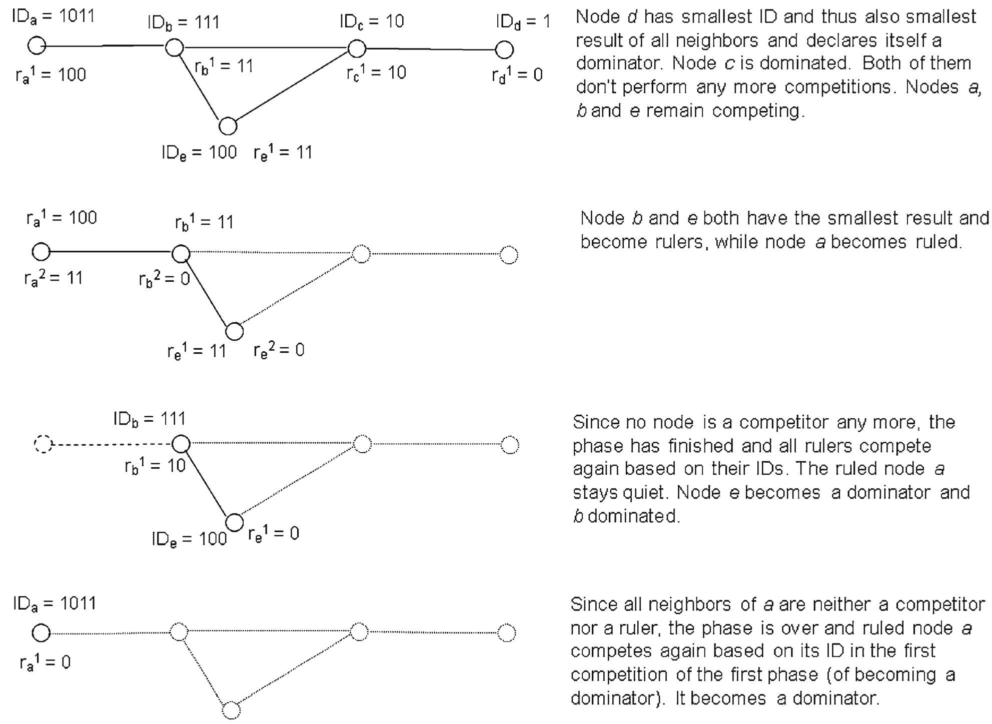

Fig. 1 Graph showing a complete execution of Algorithm MIS. Dominators and dominated nodes are shown with a *dotted line*. Ruled nodes with a *dashed line*

$r_u^{j-1} = \min_{w \in N(v)} r_w^{j-1}$. In case the length of r_u^{j-1} and r_v^{j-1} differ, we make them equal by prepending zeros to the smaller number r^{j-1}. The result r_v^j for node v gives the maximum position, s.t. the (r_v^j)th bit of number r_v^{j-1} is 1 (i.e. $y_v^{r_v^j} = 1$) and the (r_v^j)th bit of r_u^{j-1} is 0 (i.e. $y_u^{r_v^j} = 0$). If r_v^{j-1} is a minimum of all r^{j-1} (i.e. $r_v^{j-1} \le r_u^{j-1}$), then we set $r_v^j = 0$. Taking into account both cases yields: $r_v^j := \max(\{k | (y_v^k > y_u^k) \wedge (r_v^{j-1} > r_u^{j-1})\} \cup \{0\})$. Observe that by definition all bits higher than the (r_v^j)th bit are the same (i.e. $y_u^i = y_v^i$ for $r_v^j < i \le \log^{(j)} n$) if $r_v^{j-1} \ge r_u^{j-1}$.

Fast termination of Algorithm MIS will be shown in Sect. 5.1 for bounded-independence graphs. However, the algorithm is robust in the sense that it is correct for general graphs as well (see Sect. 5.2).

A node executes phases in a synchronous manner with its neighbors (see also Lemma 3). For a reader not familiar with distributed computing this might seem a too strong assumption. A simple way to solve the problem is that we let all nodes know an upper bound of n. With that all nodes can execute all steps of the algorithm in lock-step, even if some of the nodes are not participating in some of the steps (because they are not competing anymore, for instance). On the one hand this guarantees global synchronization, on the other hand our algorithm is not uniform anymore.

A better solution is to use a local synchronizer (i.e. synchronizer α). With that, all messages can be exchanged completely asynchronously; the only constraint is that nodes need to wait until their neighbors have signalled that they are okay with executing the next step of the algorithm. Using a synchronizer it may happen that some nodes already are two rounds ahead of others, however, locally all nodes are always within one step.

Algorithm MIS

For each node $v \in V$
repeat
  state s_v := competitor
  repeat
    {Phase start} $r_v^0 := ID_v$
    if s_v = ruler **then** s_v := competitor **end if**
    j:=0
    repeat
      {Competition start} j:= j+1
      if s_v = competitor **then**
        Select competitor $u \in N(v)$ with $r_u^{j-1} = \min_{w \in N(v)} r_w^{j-1}$
        $r_v^j := \max(\{k | (y_v^k > y_u^k) \wedge (r_v^{j-1} > r_u^{j-1})\} \cup \{0\})$
      end if
      Execute **Update State** s_v {Competition end}
    until $\nexists u \in (N(v) \cup v)$ with s_u = competitor {Phase end}
  until $\nexists u \in (N(v) \cup v)$ with s_u = ruler
until $s_v \in$ {dominator,dominated}

Algorithm Update State

if s_v = competitor **then**
  Exchange r^j with competing neighbors $T \subseteq N(v)$
  if $\forall t \in T$ holds $r_t^j > r_v^j$ **then**
    s_v := dominator
  else if $\forall t \in T$ holds $r_t^j \geq r_v^j$ **then**
    s_v := ruler
  end if
end if
Exchange state s with all neighbors $t \in N(v)$
if $\exists\, t \in N(v)$ with s_t = dominator **then**
  s_v := dominated
else if ($s_v \neq$ ruler) $\wedge$ ($\exists\, t \in N(v)$ with s_t=ruler) **then**
  s_v := ruled
end if

5 Analysis

We analyze Algorithm MIS for bounded-independence graphs as well as for general graphs.

5.1 Bounded-independence graphs

The proof for Algorithm MIS is done by showing correctness of the computed MIS first, i.e. dominators are independent and every node has a dominator as a neighbor in case the algorithm finishes. Then we focus on termination and give evidence that a node ends a phase after at most $\log^* n + 2$ competitions. In other words no node can be a competitor for more than $\log^* n + 2$ consecutive competitions without ever changing its state. In addition we prove that after every phase some nodes near a competitor v stop competing with v (at least) until v becomes ruled. As a next step, we prove that after the $f(2)$th phase every competing node must end up in exactly one clique of rulers and thus in the $(f(2)+1)$st phase the node in the clique with smallest ID becomes a dominator. Then we show that every non-dominated node and non-dominator has another dominator within hop distance $f(2) + 3$ after $O(\log^* n)$ rounds of communication. Since dominators are independent and the number of independent nodes within distance $f(2) + 3$ is constant, it follows that after $O(\log^* n)$ the MIS is computed.

Lemma 1 *No dominators can be adjacent. On termination of Algorithm MIS every node is either a dominator or must have at least one dominator as a neighbor.*

Proof When a node v becomes a dominator, no neighbor $u \in N(v)$ turns into a dominator in the same competition, since result r_v is smallest for all neighbors.

When a node v becomes a dominator, no neighbor can become a dominator or a ruler in a later competition. This follows from the facts that a dominator does not alter its state and that all neighbors $u \in N(v)$ have s_u = dominated after executing Algorithm Update State. They will remain in that state, as long as they are adjacent to a dominator.

The property that every node gets dominated or is a dominator after the execution of Algorithm MIS follows directly from the condition in line 16.

The upcoming lemma bounds the number of competitions per phase and furthermore says that all competitors must change their states during a phase. Since nodes that have become rulers immediately start the next one, the following lemma also bounds the time until rulers progress to the next phase.

Lemma 2 *After a phase, i.e. after at most* $\log^* n + 2$ *recursive competitions, no node is a competitor.*

Proof The first competition is based on the IDs, which have at most $\log n$ bits. The result of the first competition r^1 gives an index of a bit of the ID and thus requires at most $\lceil \log \log n \rceil$ bits. The result r^2 of the second competition is a number less than $\lceil \log \log n \rceil$ and uses at most $\lceil \log \log \log n \rceil$ bits etc. In general r^j needs up to $\lceil \log^{(j+1)} n \rceil$ bits. After $\log^* n+1$ (see Definition 2) competitions the result will be a single bit, i.e. 0 or 1. If $(r_v^{\log^* n+1} = 0) \vee \big(\forall u \in N(v)$ holds $r_u^{\log^* n+1} = 1\big)$ then $s_v \in$ {dominator, ruler} else $s_v \in$ {dominated, ruled}. Thus every node becomes a non-competitor once. Since within the loop (lines 7–14) no node turns from a non-competitor into a competitor, the lemma follows.

The next Lemma 3 essentially guarantees that phases are started and executed locally synchronously. Recall that once a node has become ruled, it stays quiet until all its neighbors are ruled or dominated and then starts the algorithm again as competitor in the first phase (of becoming a dominator).

Lemma 3 *If node v is a competitor in the ith phase (of becoming a dominator) all competing neighbors must also be in the ith phase. Moreover, for a node v executing the jth competition, all competing neighbors must also execute the jth competition.*

Proof Since we assume synchronous wake-up the very first competition and phase is started by all nodes in parallel. So assume all nodes execute the same phase and the same competition. As long as node v is a competitor all neighbors must be competing in the same competition or be dominated. If node v has been ruled and starts again with the first competition of the first phase (of becoming a ruler), then all neighbors must be ruled or dominated as well and thus if a neighbor starts a new phase it must also be the first phase and the first competition. If a node becomes a ruler in the jth competition of phase i, then all neighbors $u \in N(v)$ that have become rulers in the same competition, will start phase $i + 1$ concurrently. All other neighbors must be ruled or dominated and thus cannot start a new phase until all neighbors become ruled or dominated.

Definition 3 Let the set $U \subseteq V$ be a connected set of competitors of maximal size, s.t. no competitor $w \notin U$ is a neighbor of a node $v \in U$. For any vertex v, let U_v^i denote a connected set of competitors of maximal size that contains v in the beginning of the ith phase of node v.

Lemma 4 ensures that all nodes of a set of connected rulers have the same result r (and the results in the previous competition have the same prefix). Afterwards, Lemma 5 shows that in each phase some nodes near every ruler stop competing with it.

Definition 4 A node $u \in V$ can be reached by a path p of rulers from v in competition j of phase i, if $\exists p = (v = t_0, t_1, \ldots, u = t_q)$, s.t. $\forall (0 \le k < q)$ holds that node t_k has become a ruler in competition j of phase i.

Lemma 4 *If nodes U_v^i with $i > 0$ became rulers in competition j in phase $i-1$ of node v then for any node $u \in U_v^i$, r_u^j is the same as r_v^j, and the prefixes of r_v^{j-1} are the same as those of r_u^{j-1}, i.e. $y_v^i = y_u^i$ for $r_j^v < i \le \log n$.*

Proof Assume u was reachable by the path $p = (v = t_0, t_1, \ldots, u = t_q)$ of rulers from v and $r_v^j \ne r_u^j$. Due to the maximality of U_v^i (see Definition 3) all rulers t_i are in U_v^i, i.e. $t_i \in U_v^i$ for $0 \le i < q$. By assumption there would have to exist two neighboring rulers $t_l, t_{l+1} \in U_v^i$ with $r_{t_l}^j \ne r_{t_{l+1}}^j$. Since either $r_{t_l}^j > r_{t_{l+1}}^j$ or the other way round, they could not both have become rulers in the same competition (This would contradict Lemma 3). Assume their prefixes differed, i.e. $y_v^i \ne y_u^i$ for $r_v^j < i \le \log n$. Then r_v^j could not be equal to r_u^j.

The next lemma gives evidence that for a ruler v after every phase one node w at hop distance two and all its neighbors $N(w)$ will not compete with v (at least) until it gets ruled. Since there are at most $f(2)$ of such nodes w (see Lemma 8) only $f(2)+1$ phases are needed until a ruler has no two hop neighbors and thus must be in a clique. After the first competition of the next phase a dominator is chosen in every clique (see Lemma 9) and all other nodes in the clique are dominated.

Definition 5 Let the set $W_v^i \subseteq U_v^i$ be the set of nodes at distance two from node v that compete with v in phase i, i.e. $W_v^i := (N^2(v) \backslash N(v)) \cap U_v^i$.

Lemma 5 *Consider a node v in phase i, which has become a ruler in the jth competition for any $j \ge 0$. If $|W_v^i| > 0$ then $\exists w \in W_v^i$, which cannot be reached by a path of rulers from v.*

Proof Case $j = 0$, i.e. let us investigate the first competition, which is based on IDs. Consider the value r_v^1 of node v. If $r_v^1 = 0$, then v has a smaller ID than all its neighbors and thus is a dominator. If $r_v^1 = \log n$, then by definition $y_v^{\log n} = 1$ and node v must have had a neighbor u with $y_u^{\log n} = 0$. This neighbor u must have $r_u^1 < \log n$ and thus v cannot be a ruler. So assume $r_v^1 \in [1, \log n - 1]$.

Due to Lemma 4 all rulers s reachable by a path of rulers from v must have $r_s^j = r_v^j$ and their values r^{j-1} must have the same prefix as v. By definition of r_v^j there must exist a node $u \in N(v)$, s.t. $y_v^i = y_u^i$ for $r_v^j < i \le \log n$ and $1 = y_v^{r_v^j} > y_u^{r_v^j} = 0$. Thus $ID_v > ID_u$. Since nodes u and v differ in position r_v^1, we have $r_u^1 \ne r_v^1$. Since v is a ruler, $r_v^1 < r_u^1$. Because $r_v^1 < r_u^1$ and $ID_v > ID_u$, this neighbor u must itself have a neighbor w with $ID_w < ID_u$ and $y_w^i = y_u^i$ for $r_u^1 < i \le \log n$ and $1 = y_u^{r_u^1} > y_w^{r_u^1} = 0$. Apart from that, v and u have an ID with the same prefix from bit (at most) $\log n$ down to bit $r_v^1 + 1$. The node w cannot be a neighbor of any ruler $x \in U_v^i$ with value $r_x^1 = r_v^1$, since otherwise $r_v^1 \ge r_u^1$ because $1 = y_v^{r_u^1} = y_u^{r_u^1} > y_w^{r_u^1} = 0$, i.e. the prefix of w is smaller than that of v. See Fig. 2.

Case $j > 0$, i.e. let us look at the jth competition for $j > 0$. Assume node v was competing in all previous competitions and was in particular not a ruler after the $(j-1)$st one. The arguments are similar to those of the first competition.

Assume $0 < r_v^j \le \log^{(j)} n$ then the same reasoning applies as for the first competition – only the value for r_v^1 has to be substituted by r_v^j, ID_v by r_v^{j-1} and $\log n$ by $\log^{(j)} n$.

Assume $r_v^j = 0$. Since v was not a ruler (or dominator) in competition $j-1$, there exists a neighbor $u \in N(v)$ with $r_u^{j-1} < r_v^{j-1}$. Neighbor u cannot participate in competition j, since otherwise by definition $r_v^j > 0$. Since v is a ruler, u must have become dominated or ruled in competition $j-1$ by a neighbor $w \in N(u)$. If w became a dominator in round $j-1$, all neighbors $s \in N(w)$ became dominated in round $j-1$ as well. Thus w cannot be reached by a path of rulers from v. If w turned into a ruler in competition $j-1$ and v in competition j, then due to Lemma 3 nodes w, v cannot be in the same connected set of rulers and thus node w cannot be reached by a path of rulers from v.

Figure 3 illustrates that no edge can exist between a node w having been a ruler in competition $j-1$ and a ruler v in current competition j, since otherwise node v would have been already ruled (or dominated).

Lemma 6 shows that every connected set of competitors, containing node v in phase j, must be a subset of a previous connected set of competitors, containing node v in phase i with $i < j$. This will be used by Lemma 7 to show that if a set of arbitrary nodes does not have a common node with

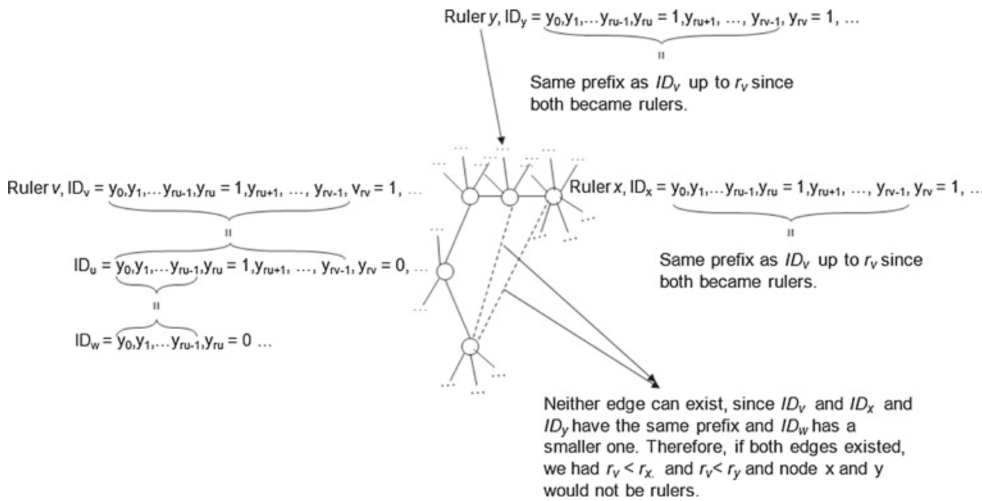

Fig. 2 Graph of some nodes that participated with node v in the first competition. Assume nodes v, x, y became rulers and furthermore $ID_v > ID_u > ID_w$. In this case w cannot be reachable by a path P of rulers, such as $P = (v, y, w)$ or $P = (v, y, x, w)$

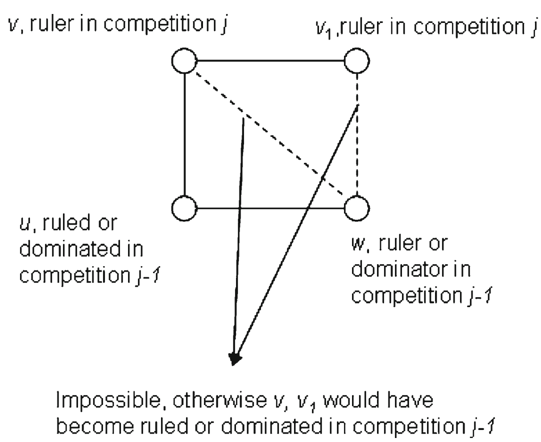

Fig. 3 Graph of some nodes that participated with node v in competition $j-1$ and j

a set of connected competitors in some phase, then this will hold for all proceeding phases.

Lemma 6 *If node v has been either a ruler or a competitor during j phases, then $U_v^j \subseteq U_v^i$ for $i < j$.*

Proof Let the neighborhood $N(T)$ of a set of nodes $T \subseteq V$ be the set $\{s \mid \exists u \in T : s \in N(u) \backslash T\}$.

Let $w \in U_v^i$ be the (or one of the) first node(s) that become a ruler or dominator in phase i (say in competition k). By definition all nodes $N(U_v^i)$ have been ruled or dominated in competition k and thus cannot become rulers. Thus $U_w^{i+1} \subseteq U_v^i$ and all neighboring nodes $N(U_w^{i+1})$ of U_w^{i+1} are ruled or dominated as well. Consider the next competition l with $l > k$, where some node $t \in (U_v^i \backslash (U_w^{i+1} \cup N(U_w^{i+1}))$ becomes a dominator or ruler. Ten we have that $U_v^{i+1} \subseteq (U_v^i \backslash (U_w^{i+1} \cup N(U_w^{i+1})) \subseteq U_v^i$. The argument proceeds in the same manner. Thus we have that $U_v^{i+1} \subseteq U_v^i$. Analogously, it follows that $U_v^{i+2} \subseteq U_v^{i+1} \subseteq U_v^i$ a.s.o.

Lemma 7 *For a set $T \subset V$, s.t. $U^i \cap T = \emptyset$ holds that $U_v^j \cap T = \emptyset$ with $i < j$.*

Proof Due to Lemma 6, we have that $U_v^j \subseteq U_v^i$. Due to the disjointedness of U_v^i and T, U_v^j and T must also be disjoint.

The next two lemmas together give an upper bound of the number of (consecutive) phases until a node becomes a dominator.

Lemma 8 *If a node v has become a ruler in the $f(2)$th phase, then it is in a clique of competitors in phase $f(2)+1$.*

Proof Let a node w, as defined in Lemma 5, for phase i be denoted by $w_i \in W_v^i$. Lemma 5 implies that no neighbor $t \in N(w_i)$ can be a ruler reachable by a path of rulers from v. Thus by definition $U_v^{i+1} \cap N(w_i) = \{\}$. Due to Lemma 7, no node $t \in N(w_i) \cup w_i$ will be reachable by a path of competitors from v until (at least) v has become ruled. Since by definition $W_v^i \subseteq U_v^i$, this implies that $W_v^{i+1} \subseteq (W_v^i \backslash (N(w_i) \cup w_i))$. As a consequence nodes $w_i \in W_v^i$ and $w_k \in W_v^k$ with $i \neq k$ (i.e. from different phases) are independent. The size of a maximum independent set in $N^2(v)$ is upper bounded by $f(2)$. In every phase i, at least one node $w_i \in W_v^i$ at distance two from v is removed. Thus after at most $f(2)$ phases, node v cannot reach any competitor at hop distance at least 2 by a path of competitors, i.e. $W_v^{f(2)+1} = \{\}$ and the nodes $U_v^{f(2)+1}$ form a clique.

Lemma 9 *If a node v is still competing in the $(f(2)+1)$st phase then either v or a neighbor of v will become a dominator in that phase.*

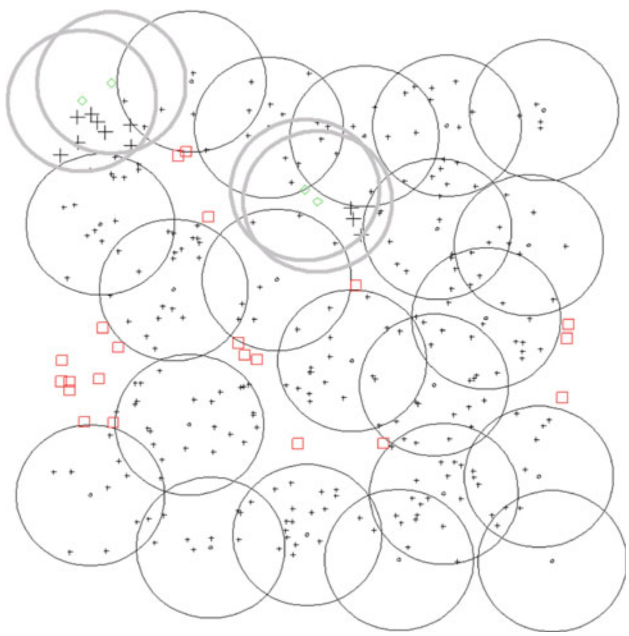

Fig. 4 Algorithm MIS on an instance of a UDG. It shows the state of each node after the very first competition. For dominators and rulers a *circle* indicating the transmission range is shown. Rulers are depicted by *diamonds*, ruled nodes by *big crosses*, dominated nodes by *small crosses*, dominators by *small circles* and competitors by *boxes*. As can easily be seen, most nodes are already dominated after the first competition. After the second there are no competitors left and after the first competition of the next phase the algorithm is done

Proof Using Lemma 8, we have that each competitor v is in a clique in the beginning of the $(f(2)+1)$st phase. Thus in the first competition the node with the smallest ID of the clique will become a dominator.

Next we show that for every non-dominated node v a dominator is chosen within constant distance from v. Essentially this is because a node cannot end a phase as a ruled node without having a ruler advancing to the next phase in its neighborhood (see Algorithm Update State). Due to Lemma 9 after a constant number of phases a node must become a dominator or dominated (See Fig. 4).

Since dominators are independent (see Lemma 1) and in a bounded-independence graph the number of independent nodes within constant distance is also a constant, only $O(\log^* n)$ rounds of communication are needed (see Theorem 1).

Lemma 10 *After $O(\log^* n)$ rounds of communication each node v either becomes a dominator or there exists an additional node that has become a dominator within hop distance $f(2)+3$.*

Proof We will show that the distance between a ruler in phase i and node v is at most i. After the first phase, every node v is a ruler itself or adjacent to a ruler.

Assume the distance was at most $i-1$ after the $(i-1)$st phase. In the ith phase only rulers become competitors again (line 5) and participate in the competitions. Thus after the ith phase, every competitor will become a ruler or a dominator or have at least one of the two in its neighborhood. Thus the distance between a ruler and a ruled node grows at most by 1 per phase.

Due to Lemma 9 every competitor or one of its neighbors must become a dominator in the $(f(2)+1)$st phase. Assume node $u \in U_v^0$ started competing with v. Since a phase has at most $\log^* n + 2$ competitions (see Lemma 2) and to perform a competition the algorithm requires three communication rounds, after at most $3 \cdot (f(2)+1) \cdot (\log^* n + 2) \in O(\log^* n)$ rounds, node u must have got a dominator within distance $f(2)+2$. Assume node u is ruled, then at least one of its neighbor must be competing or it starts competing again itself. Thus after at most $O(\log^* n)$ rounds of communication it must have got an additional dominator within hop distance $f(2)+3$.

Theorem 1 *The total time to compute a MIS is in $O(f(f(2)+3)\log^* n)$ and each message is of size $O(\log n)$.*

Proof Due to Lemma 10 within $O(\log^* n)$ every node gets a dominator within distance $f(2)+3$. Since dominators are independent (Lemma 1), the number of dominators within distance $f(2)+3$ is upper bounded by the size of a maximum independent set in $N^{f(2)+3}(v)$, which is $f(f(2)+3)$. This yields that $f(f(2)+3) \cdot O(\log^* n)$ rounds of communication are needed.

For initialization all IDs (of maximum length $\log n$) have to be exchanged among neighbors requiring one communication round for the first competition to execute. The following update of the state needs every message to be of size $O(\log\log n)$ and takes three rounds of communication. Namely, exchanging the result of the prior competition which also serves as input for the next. Additionally, a request and possibly delayed reply of the current state of all neighbors. Apart from that no communication has to take place for initialization and the first competition. In an analogous derivation the second competition requires only messages of size $O(\log\log\log n)$ etc. and three communication rounds. Thus each message is of size at most $O(\log n)$.

5.2 General graphs

For a general graph the size of a MaxIS in the neighborhood $N^r(v)$ of a node v can be bounded by a function $g(|N^r(v)|)$, since the size of a MaxIS including nodes up to distance r cannot be larger than the size of the neighborhood $N^r(v)$.

Theorem 2 *Let the function $g(\Delta^r)$ be such that for each node $v \in V$, the size of a MaxIS in the neighborhood $N^r(v)$ is at most $g(\Delta^r)$, $\forall r \geq 0$. Then algorithm MIS needs at most $O(\min(n, g(g(\Delta^2)+3))$ time.*

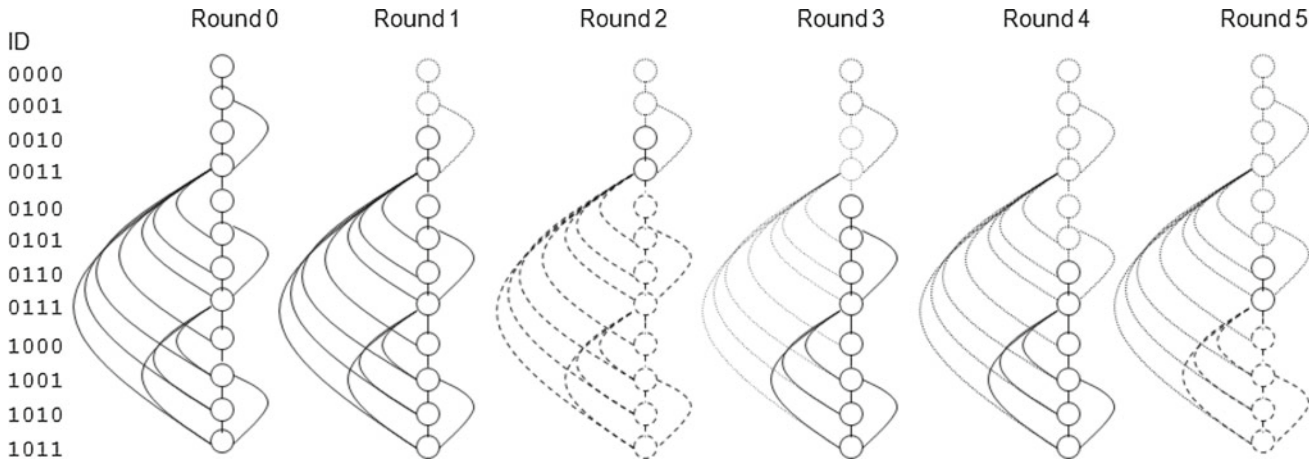

Fig. 5 Algorithm MIS running on a unbounded-independence graph, taking time O(n). The *solid lines* show competitors. The *dotted lines* are dominators and dominated nodes. The *dashed lines* indicate ruled nodes

Proof To see that the running time is in $O(g(g(\Delta^2)+3))$ the same analysis as in Sect. 5.1 can be used. Thus let us focus on the case that the number of rounds is also in O(n).

While not all nodes are dominated or dominators, there always exists at least one set U of connected competitors (or rulers that become competitors immediately). Consider the connected set of competitors U_v^i (see Definition 3) for phase i. For a competition we can distinguish two outcomes: First, all nodes $u \in U_v^i$ become rulers, ruled, dominated or dominators. Second, at least one node $w \in U_v^i$ remains a competitor and at least one node $u \in U_v^i$ becomes a ruler or a dominator. (Observe, that it is not possible that all nodes remain competing, since the node with minimum result will become a ruler or dominator.) In case, some nodes $W \subset U_v^i$ became rulers, say node $s \in U_v^i$ for instance, then due to Lemma 5 in phase $i+1$ there exists a node $w \in W \backslash U_s^{i+1}$. Therefore, there exist two independent non-empty sets U_s^{i+1} and $U_v^i \backslash U_s^{i+1}$ (at least $w \in U_v^i \backslash U_s^{i+1}$). Apart from that the set $U_v^i \backslash U_s^{i+1}$ either contains a dominator or a ruler. In the next competition (the first of phase $i+1$) the node with minimum ID in U_s^{i+1} becomes a dominator and if the set $U_v^i \backslash U_s^{i+1}$ contained a ruler also the node with minimum ID in this set. Assume for phase i it takes $j > 1$ competitions until all nodes in U_v^i become non-competitors. The number of dominators $u \in U_v^i$ obtained during the last $j-1$ competitions in phase i plus the number of dominators in the first competition of phase $i+1$ of nodes $u \in U_v^i$, is at least $\max\{j-1, 1\}$. Thus, in the worst case all nodes $u \in U_v^i$ become non-competitors after two competitions and only one dominator can be accredited to these two competitions which yields at most $2 \cdot n$ competitions. To perform a competition the algorithm requires three communication rounds.

The running time can indeed be O(n) as shown by the following graph: Each node v with $0 < ID_v < n-1$ has edges to nodes u, w with $ID_u = ID_v - 1$ and $ID_w = ID_v + 1$. A node v having its last two bits equal to 1 (i.e. ID_v mod 100 = 11) is additionally connected to all nodes with higher ID and to the node u with $ID_u = ID_v - 2$ (see Fig. 5). The states of the nodes during the execution of the algorithm follow a pattern which repeats every three rounds of communication. In the first round the node with smallest ID (with the last two bits being 00) becomes a dominator and its neighbor becomes dominated (its ID ends with 01). All other nodes remain competitors. In the second round the two nodes with smallest IDs (having the last two bits 10 and 11) change to rulers and all others become ruled. In the third round the node with smallest ID (ending with 10) turns into a dominator and its neighbor becomes dominated. All ruled nodes switch back to competitors again.

6 Applications of MIS

Our MIS algorithm serves as a key building block to tackle many other problems for bounded-independence graphs.

6.1 CDS and MDS

In order to obtain a CDS given a MIS S, each node $v \in S$ chooses a shortest path to every node $u \in (N^3(v) \cap S)$ with $ID_u < ID_v$ and adds all nodes from the path to the CDS. Because the size of the set $N^3(v) \cap S$ is at most $f(3)$ and the length of any chosen path is at most 3, at most $3 \cdot f(3) \cdot |S| + |S|$ nodes form the CDS. Since S is a constant approximation of the MDS in a bounded-independence graph, we get a constant approximation of the Minimum CDS (MCDS) in O($\log^* n$) time. See also [2]. Due to a lower bound [9,17] of $\Omega(\log^* n)$ to get a constant approximation of an MDS, our algorithm

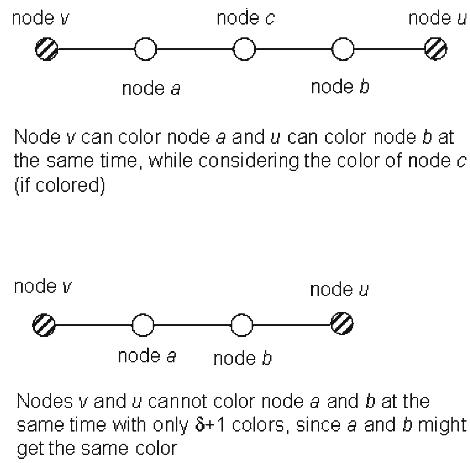

Fig. 6 Illustration showing that the distance between two nodes concurrently coloring their neighbors must be at least 4

has asymptotically optimal time complexity for the MDS and MCDS problem. (Observe that the lower bound is also valid for the MCDS problem, since a MCDS is a constant approximation of an MDS in a bounded-independence graph.)

6.2 PTAS for MDS and MaxIS

By using the clustering technique from [14] together with our MIS algorithm, a $(1+\epsilon)$-approximation for the MDS and MaxIS problem is computed in $O(\log^* n/\epsilon^{O(1)})$ time. More precisely, we use Theorem 5.8 from [14]:

Theorem 3 (Theorem 5.8 [14]) *Let $G = (V, E)$ be a polynomially bounded-independence graph. Then, there exist local, distributed $(1+\epsilon)$—approximation algorithms, $\epsilon > 0$, for the MaxIS and MDS problems on G. The number of communication rounds needed for the respective construction of the subsets is $O\big(T_{MIS} + \log^* n/\epsilon^{O(1)}\big)$, where T_{MIS} is the time to compute a MIS in G.*

6.3 $\delta + 1$ Coloring

We state two methods for computing a $\delta + 1$ coloring, both relying on the same observation that a node v can color all its neighbors if no other node $u \in N^3(v)$ does so at the same time. See Fig. 6.

In the first procedure a node v competes against a neighbor $u \in N^3(v)$, i.e. we compute a MIS S on the graph $G' = (V, E')$ with $E' = E \cup \{(u, v)|\{(u, s), (s, v)\} \subseteq E\} \cup (\{(u, v)|\{(u, s), (s, t), (t, v)\} \subseteq E\})$. All MIS nodes $v \in S$ color all their neighbors in G and themselves, taking into account already used colors of colored nodes $w \in N^2(v)$. This procedure is repeated for all uncolored nodes. In each iteration i a MIS S_i is computed and an uncolored node $u \in V$ either gets colored or has distance at most three in G to a node $v \in S_i$. The union of MIS S_i and S_j with $i \neq j$ forms an independent set in G. The number of independent nodes for node v at distance at most three is bounded by $f(3)$. Thus after at most $f(3)$ computations of a MIS in G' node v gets colored. The graphs G' is also bounded-independence with $f'(r) \leq f(3 \cdot r)$, yielding an overall running time of $O(\log^* n)$. Due to Linial's $\Omega(\log^* n)$ [18] lower bound our algorithm is asymptotically optimal. Observe that in order to compete against all nodes in $N^k(v)$ messages of size $O(\log n)$ are sufficient, since a node only needs to know the minimum result of a competition. At first a node broadcasts its own result to all neighbors and from then on forwards the smallest result it received so far for $k - 1$ rounds. A distance two coloring with the same message and time complexity can be obtained, when every node v competes against all nodes $u \in N^6(v)$ instead of $N^3(v)$ and colors its two hop neighborhood.

Alternatively to the above method a MIS S on $G = (V, E)$ can be computed first. Next we consider the graph G' defined by nodes S and edges between two nodes $u, v \in S$, if they can reach each other by a path of length at most three, i.e. $G' = (S, E')$ with $E' = \{(u, v)|u, v \in S \wedge \big((\exists s \in (V \backslash S)(\{(u, s), (s, v)\} \subseteq E) \vee (\exists s, t \in (V \backslash S)(\{(u, s), (s, t), (t, v)\} \subseteq E)\big)\}$. We compute a MIS $S' \subseteq S$ on G'. Every node $v \in S'$ colors all its neighbors, respecting already used colors. The process is repeated for all uncolored nodes. Thus in an iteration a node v either gets colored or a neighbor $u \in N^4(v) \cap S'$ colors itself and all its neighbors. Since colored nodes are not considered any more, for a node v there are at most $f(4)$ such neighbors $u \in N^4(v)$. Therefore in total at most $2 \cdot f(4)$ MIS computations are required, giving an overall running time of $O(\log^* n)$.

6.4 Maximal matching

We can use the same idea as for the coloring in Sect. 6.3. We compute a MIS S such that all nodes in the MIS have distance at least 6. This allows a node v in the MIS S to compute a maximal matching for all nodes $u \in N(v) \cup v$, i.e. it can pick any edge $e = (u, w)$ with $w \in N^2(v)$ and add it to the matching. All nodes adjacent to an edge in the matching and those that have no unmatched neighbors are removed and the process repeats, i.e. again a MIS S' is computed for the remaining nodes, such that all nodes in the MIS have distance at least 6.

Whenever a MIS S is computed, any node is either matched and stops or a node within distance 6 matches all its neighbors. Since the number of nodes that can match all their neighbors within distance 6 corresponds to the size of a maximum independent set, we need to compute at most $O(f(6))$ MIS S, giving an overall running time of $O(\log^* n)$.

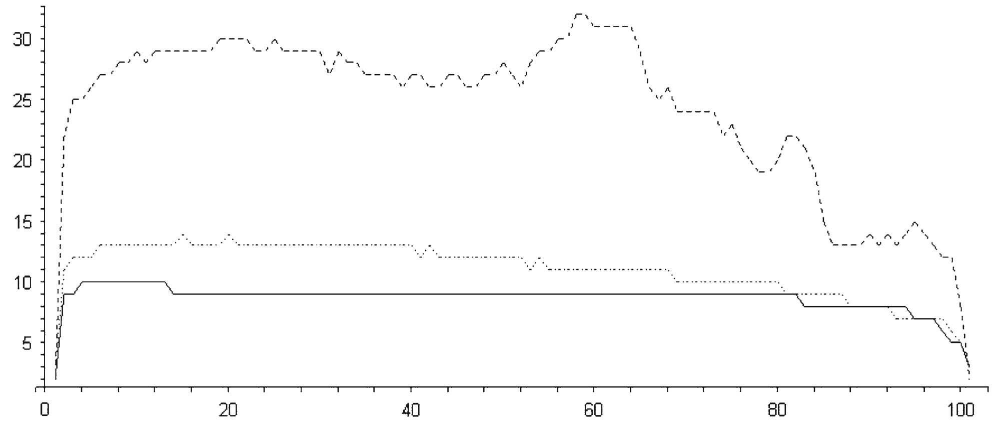

Fig. 7 Simulations on Erdős Rényi graphs of 1500 nodes; The probability of an edge between two nodes is indicated on the x-axis. The y-axis shows the number of communication rounds. The *dashed line* shows the rounds needed for Luby's algorithm, the *dotted* one for Algorithm MAX and the *solid* one for our Algorithm MIS

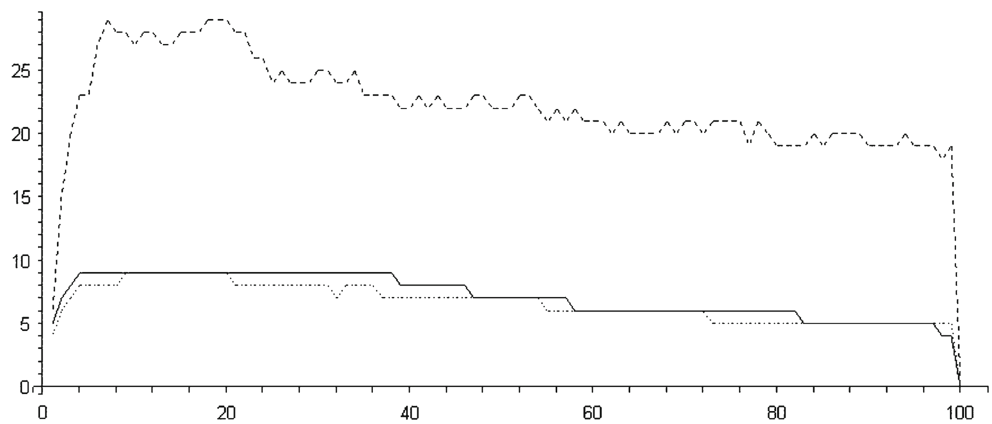

Fig. 8 Simulations on UDGs of 1500 nodes. The probability of an edge between two nodes is indicated on the x-axis. The y-axis shows the number of communication rounds. The *dashed line* shows the rounds needed for Luby's algorithm, the *dotted* one for Algorithm MAX and the *solid* one for our Algorithm MIS

6.5 Others

Our algorithm MIS has been used in the area of self-assembling systems [25], to compute a PTAS for minimum clique partition in the UDG model [22], to approximate the robot assignment [7] and the facility location problem [21] just to name a few.

7 Experimental results

We compared three algorithms namely Algorithm MIS, Luby's [19] and Algorithm MAX, where a node joins the MIS if its ID is maximum among all neighbors. In fact, we slightly adapted algorithm MIS, such that before it performs the first competition, it updates the state based on the IDs, i.e., all nodes become dominators that have minimum ID among their neighbors and dominated nodes inform their neighbors. For a competition our algorithm needs three communication rounds: One to exchange the result of a competition and two more to let all neighbors know about the new state. Algorithm MAX needs two rounds to inform all neighbors about the new states. Both algorithms MIS and MAX need an initial round to exchange the IDs among the neighboring nodes. For Luby's algorithm the update of the states takes two rounds and we assumed that one initial round is needed to get to know the degree of the nodes.

We considered random graphs, where an edge between two nodes was added with probability p. Our experiments were conducted for UDG and random graphs with 1500 nodes. They indicate that Luby's algorithm performs worst in general. Algorithm MIS and MAX perform relatively similar (see Figs. 7 and 8), however Algorithm MAX lacks good worst case guarantees for UDG graphs. In particular for loosely connected networks, the chance that there are long chains of nodes with increasing IDs increases with the network size and thus algorithm MAX performs worse in such scenarios.

Due to the random arrangement of nodes the first selection of dominators in Algorithm MIS and MAX is similar to the one of Luby, except for the fact that Algorithm MIS and MAX do not face the problems of multiple neighbors joining the MIS at the same time and that no nodes join at all. If there are many small (unconnected) subgraphs of nodes not covered by the MIS, Luby's algorithm has a high chance that at least for some of them, no node decides to join the MIS, whereas Algorithm MIS and MAX both will select at least one.

8 Conclusions

In this paper we have presented a novel deterministic coin tossing technique which enables us to achieve an asymptotically optimal distributed algorithm for computing a MIS in bounded-independence graphs. Although the hidden constants in the analysis are significant, our simulations indicate that the constants will be small in practice. Indeed, quite surprisingly, the example of Fig. 4 was completed after 3 rounds of communication only. Our algorithm has been applied in various settings beyond the MIS problem, for instance for computing a $\delta + 1$ coloring, an asymptotically optimal CDS in O($\log^* n$) or for solving the facility location problem [21].

We believe that our paper strikes a chord with symmetry breaking. Randomized symmetry breaking has the problem of only producing results "in expectation". Often however symmetry breaking algorithms are used as building blocks, and then they need to work "with high probability" which regularly causes a logarithmic overhead. In other words, when it comes to "ultra-fast" distributed algorithms, determinism may have an advantage over randomization

Finally, as promised in Sect. 3, let us discuss the communication model in more detail. We presented our algorithm in the classic local model, where each node can talk to each neighbor in each round. In some application domains, in particular in wireless networks, this assumption is too demanding, as it asks for a perfect (and hence non-existent!) media access control (MAC) protocol. In reality MAC protocols are quite unreliable, with messages not being received because of wireless channel fluctuations, or message collisions. One way to address this is to study the problem in a model that includes the MAC layer, in the sense that the algorithm designer has to exactly specify at what time the nodes transmit or receive. This is a cumbersome job, especially since the result is often related to the algorithm in the clean local model, e.g., the without collision detection model [24] uses the local algorithm presented in this paper as a building block.

What about situations where the software engineer has no control over the MAC layer? How should our algorithm be implemented? We argue that one should simply simulate our algorithm in the "rollback compiler" self-stabilization framework [5]. Note that our algorithm does not use all the flexibility provided by the local model, instead the result messages r_v^i can be broadcast in the neighborhood. In the self-stabilization framework our protocol then boils down to repeatedly transmitting the same single message, containing information about the relevant result values computed in the individual phases/competitions. This message is of logarithmic size, and resilient to a whole array of failures and dynamics, e.g., message collisions, message failures, even nodes rebooting and topology changes due to mobility or late node wake-ups. Nodes will always have a correct MIS at most $O(t \log^* n)$ time after the last failure, assuming that each node can successfully transmit a message at least once in time t!

References

1. Alon, N., Babai, L., Itai, A.: A fast and simple randomized parallel algorithm for the maximal independent set problem. J. Algorithms **7**(4), 567–583 (1986)
2. Alzoubi, K., Li, X., Wang, Y., Wan, P., Frieder, O.: Geometric spanners for wireless ad hoc Networks. IEEE Trans. Parallel Distributed Syst. **14**(5), 408–421 (2003)
3. Andersson, A., Hagerup, T., Nilsson, S., Raman, R.: Sorting in linear time. J. Comput. Syst. Sci. **57**, 74–93 (1998)
4. Awerbuch, B., Goldberg, A.V., Luby, M., Plotkin, S.A.: Network decomposition and locality in distributed computation. In: Proceedings of the 30th Symposium on Foundations of Computer Science (FOCS), pp. 364–369 (1989)
5. Awerbuch, B., Varghese, G.: Distributed program checking: a paradigm for building self-stabilizing distributed protocols. In: Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS) (1991)
6. Barenhoim, L., Elkin, M.: Sublogarithmic Distributed MIS Algorithm for Sparse Graphs using Nash-Williams Decomposition. In: Journal of Distributed Computing Special Issue of selected papers from PODC 2008 (2010)
7. Bonorden, O., Degener, B., Kempkes, B., Pietrzyk, P.: Complexity and approximation of a geometric local robot assignment problem. In: ALGOSENSORS (2009)
8. Cole, R., Vishkin, U.: Deterministic coin tossing with applications to optimal parallel list ranking. Inf. Control **70**(1), 32–54 (1986)
9. Czygrinow, A., Hanckowiak, M., Wawrzyniak, W.: Fast distributed approximations in planar graphs. In: DISC (2008)
10. Gfeller, B., Vicari, E.: A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs. In: Proceedings of the 26th ACM Symposium on Principles of Distributed Computing (PODC) (2007)
11. Goldberg, A., Plotkin, S., Shannon, G.: Parallel symmetry-breaking in sparse graphs. SIAM J. Discrete Math.(SIDMA) **1**(4), 434–446 (1988)
12. Israeli, A., Itai, A.: A fast and simple randomized parallel algorithm for maximal matching. Inf. Process. Lett. **22**, 77–80 (1986)
13. Kuhn, F., Moscibroda, T., Nieberg, T., Wattenhofer, R.: Fast deterministic distributed maximal independent set computation on

growth-bounded graphs. In: Proceedings of the 19th International Symposium on Distributed Computing (DISC) (2005)
14. Kuhn, F., Moscibroda, T., Nieberg, T., Wattenhofer, R.: Local approximation schemes for ad hoc and sensor networks. In: Proceedings of the 3rd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC) (2005)
15. Kuhn, F., Moscibroda, T., Wattenhofer, R.: On the locality of bounded growth. In: Proceedings of the 24th ACM Symposium on Principles of Distributed Computing (PODC), pp. 60–68 (2005)
16. Kuhn, F., Moscibroda, T., Wattenhofer, R.: What cannot be computed locally! In: Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC), pp. 300–309 (2005)
17. Lenzen, C., Wattenhofer, R.: Leveraging linial's locality limit. In: 22nd International Symposium on Distributed Computing (DISC) September (2008)
18. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992)
19. Luby, M.: A simple parallel algorithm for the maximal independent set problem. SIAM J. Comput. **15**, 1036–1053 (1986)
20. Panconesi, A., Srinivasan, A.: Improved distributed algorithms for coloring and network decomposition problems. In: Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC), pp. 581–592. ACM Press (1992)
21. Pandit, S., Pemmaraju, S.: Finding facilities fast. In: Proceedings of the 10th International Conference on Distributed Computing and Networking (ICDCN) (2009)
22. Pirwani, I.A., Salavatipour, M.R.: A ptas for minimum clique partition in unit disk graphs. CoRR (2009)
23. Schneider J., Wattenhofer, R.: A log-star distributed maximal independent set algorithm for growth-bounded graphs. In: 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, August (2008)
24. Schneider, J., Wattenhofer, R.: Coloring unstructured wireless multi-hop networks. In: Proceedings of the 28th ACM Symposium on Principles of Distributed Computing (PODC) (2009)
25. Sterling, A.: Self-assembling systems are distributed systems. CoRR (2009)