A Log-Star Distributed Maximal Independent Set Algorithm for Growth-Bounded Graphs

Johannes Schneider, Roger Wattenhofer
{jschneid, wattenhofer}@tik.ee.ethz.ch
Computer Engineering and Networks Laboratory
ETH Zurich, 8092 Zurich, Switzerland

ABSTRACT

We present a novel distributed algorithm for the maximal independent set (MIS) problem. On growth-bounded graphs (GBG) 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. Our algorithm answers prominent open problems in the ad hoc/sensor network domain. For instance, 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 in $O(\log^* n)$ time, where δ is the maximum degree of the graph.

Categories and Subject Descriptors
F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems – *computations on discrete structures*;
G.2.2 [**Discrete Mathematics**]: Graph Theory – *graph algorithms*;
G.2.2 [**Discrete Mathematics**]: Graph Theory – *network problems*

General Terms
Algorithms, Theory

Keywords
Ad Hoc Networks, Sensor Networks, Radio Networks, Unit Disk Graphs, Growth Bounded Graphs, Local Algorithms, Parallel Algorithms, Maximal Independent Sets, Dominating Sets, Connected Dominating Sets, Coloring, Symmetry Breaking

1. INTRODUCTION

Minimum dominating sets (MDS) and connected dominating sets (CDS) are most likely the best 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 Section 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 Section 3 we adopt the so-called growth-bounded graph (GBG) model [11]; the GBG model only restricts the number of independent nodes in each neighborhood and is therefore a generalization of the UDG, QUDG and UBG models.

In Section 4 we present a novel distributed algorithm for the maximal independent set (MIS) problem. On growth-bounded graphs our algorithm finishes in $O(\log^* n)$ time, n being the number of nodes. As we will discuss in more detail in Section 2, our algorithm beats all existing algorithms for geometric models such as UDG or GBG by an exponential factor. Indeed, thanks to a lower bound argument by Linial [15], our algorithm is asymptotically optimal. In the GBG model a MIS is a constant approximation of a MDS, hence our algorithm gives the fastest constant MDS approximation. In Section 5 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 δ+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 distinct between nodes, i.e. IDs. In their pioneering work

PODC'08, August 18–21, 2008, Toronto, Ontario, Canada.

Cole and Vishkin [6] 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 [8, 3, 12, 11, 10]. However, it seems that the technique is limited to simple graphs such as rings or other constant-degree graphs. In contrast to Cole/Vishkin our technique extends to unbounded degree.

The MIS problem has also been studied in graphs beyond the ring (and other constant degree graphs). In general graphs the simple and elegant randomized algorithm by Luby [16] with running time of $O(\log n)$ (see as well [1, 9]) outperforms the fastest known deterministic distributed algorithm [18] which is in $O(n^{\sqrt{c/\log n}})$ 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 $\Omega(\sqrt{\log n/\log\log n})$ or $\Omega(\log\Delta/\log\log\Delta)$ communication rounds for computing a MIS [13].

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 [15]. 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 GBG) have been discovered. The currently fastest randomized algorithm by Gfeller and Vicari [7] 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 [10] with time complexity $O(\log\Delta \cdot \log^* n)$ is used, which computes an $O(\log\Delta)$ independent ruling set in the first phase. This 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 [7], it is also deterministic, and last not least simpler. Thanks to Linial's lower bound we know that it is asymptotically optimal. So far the most general descendant of [6] is [12] which achieves asymptotic optimality in a geometric model where all distances to the neighbors are known. It is rather surprising that we can match the bound of [12] 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 $S \subseteq V$ is a maximal independent set (MIS), if it is independent and there exists no independent superset $T \supset S$. A MIS S of maximum cardinality, i.e. $|S| \geq \max_{\text{MIS } T} |T|$, is called a maximum independent set (MaxIS). We consider growth-bounded graphs, which are defined as:

DEFINITION 1. *A graph $G = (V, E)$ is growth-bounded if there is a polynomial 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 $f(r)$, $\forall r \geq 0$.*

In particular, this means that for a constant c the value $f(c)$ is also a constant. A subclass of growth-bounded graphs are quasi unit disk graphs and unit disk graphs, which are often used to model wireless communication networks and have $f(r) \in O(r^2)$.

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 $O(\log n)$ 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 an 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: $x^l, x^{l-1}, ..., x^1$, 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 MIS nodes remain. For all nodes not in the MIS or not adjacent to a MIS node, 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 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. In other words, competitions are recursive.

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, it does not compete any more for at most $\log^* n + 2$ recursive competitions (we call this a phase; lines 4 to 14 in Algorithm MIS). In a phase every competing node must have changed its state (as will

be shown in Section 6). The subsequent competition (of the next phase) starts over again by using IDs. A neighbor of a ruler gets ruled (if not dominated) and stays quiet until all neighbors are ruled (or dominated). After a node v has executed a constant number of phases (we call this a stage; lines 2 to 15), it must become ruled. It starts a new phase and becomes a competitor again if all neighbors $u \in N(v)$ are ruled as well. We will give no bound on the time when a node starts a new stage. However, as long as node v waits, some neighbor $u \in N(v)$ is not ruled and thus executing a stage. For some more examples including updates of states consider Figure 1. For an overview of all possible state transitions during the execution of the algorithm see Figure 2.

Next, we give a more formal definition of a competition to clarify our notation. Let r_v^j denote the result of the j^{th} (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_u^{j-1} consists of t bits ($t \leq \log^{(j)} n$ as shown in Lemma 4) and has the form $r_u^{j-1} = y_u^t, y_u^{t-1}, ..., y_u^1$. 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 j^{th} (recursive) competition with $j \geq 1$ node v chooses a competitor $u \in N(v)$, s.t. $r_u^{j-1} = \min_{w \in N(v)} r_w^{j-1}$. 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. $r_v^{j-1} \leq r_u^{j-1}$), 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 $r_v^j < i \leq \log^{(j)} n$) if $r_v^{j-1} \geq r_u^{j-1}$.

Termination of Algorithm MIS will be shown in Section 6 for growth-bounded graphs. However, the algorithm is robust in the sense that it is correct for general graphs as well.

A node executes phases, stages and competitions in a synchronous manner with its neighbors. 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 stages ahead of others, however, locally all nodes always are within one step.

5. APPLICATIONS OF MIS

Our MIS algorithm serves as a key building block to tackle many other problems for growth-bounded graphs. We give three well-known examples.

Algorithm MIS

For each node $v \in V$
1: **repeat**
2: {Stage start} state $s_v :=$ competitor
3: **repeat**
4: {Phase start} $r_v^0 := ID_v$
5: **if** $s_v =$ ruler **then** $s_v :=$ competitor **end if**
6: j:=0
7: **repeat**
8: {Competition start} j:= j+1
9: **if** $s_v =$ competitor **then**
10: Select competitor $u \in N(v)$ with $r_u^{j-1} = \min_{w \in N(v)} r_w^{j-1}$
11: $r_v^j := \max(\{k | (y_v^k > y_u^k) \wedge (r_v^{j-1} > r_u^{j-1}) \cup \{0\})$
12: **end if**
13: Update state s_v {Competition end}
14: **until** $\nexists u \in (N(v) \cup v)$ with $s_u =$ competitor {Phase end}
15: **until** $\nexists u \in (N(v) \cup v)$ with $s_u =$ ruler {Stage end}
16: **until** $s_v \in$ {dominator,dominated}

Algorithm Update State

1: **if** $s_v =$ competitor **then**
2: Exchange r^j with competing neighbors $t \in T \subseteq N(v)$
3: **if** $\forall t \in T$ holds $r_t^j > r_v^j$ **then**
4: $s_v :=$ dominator
5: **else if** $\forall t \in T$ holds $r_t^j \geq r_v^j$ **then**
6: $s_v :=$ ruler
7: **end if**
8: **end if**
9: Exchange state s with all neighbors $t \in N(v)$
10: **if** $\exists\ t \in N(v)$ with ($s_t =$ dominator) **then**
11: $s_v :=$ dominated
12: **else if** ($s_v \neq$ ruler) $\wedge$ ($\exists\ t \in N(v)$ with (s_t=ruler)) **then**
13: $s_v :=$ ruled
14: **end if**

5.1 CDS and MDS

In order to obtain a CDS given a MIS S, each node $v \in S$ chooses a shortest path for 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 growth bounded graph, we get a constant approximation of the Minimum CDS (MCDS) in $O(\log^* n)$ time. See also [2]. Due to a lower bound [14] of $\Omega(\log^* n)$ to get a constant approximation of an MDS, our algorithm 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 growth-bounded graph.)

5.2 PTAS for MDS and MaxIS

By using the clustering technique from [11] 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.

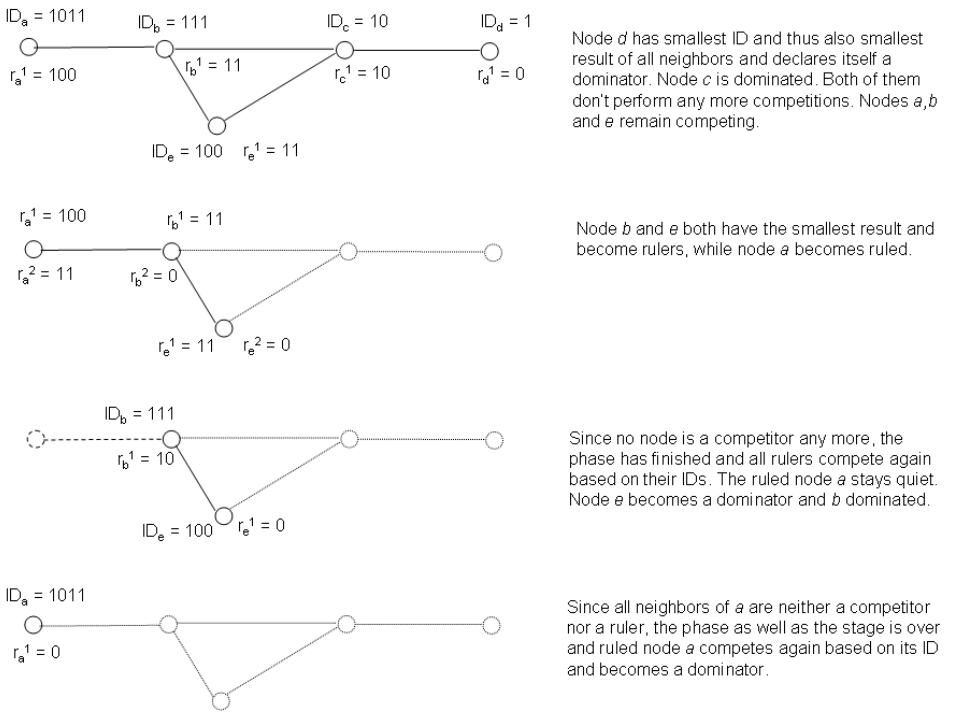

Figure 1: Graph showing a complete execution of Algorithm MIS. Dominators and dominated nodes are shown with a dotted line. Ruled nodes with a dashed line.

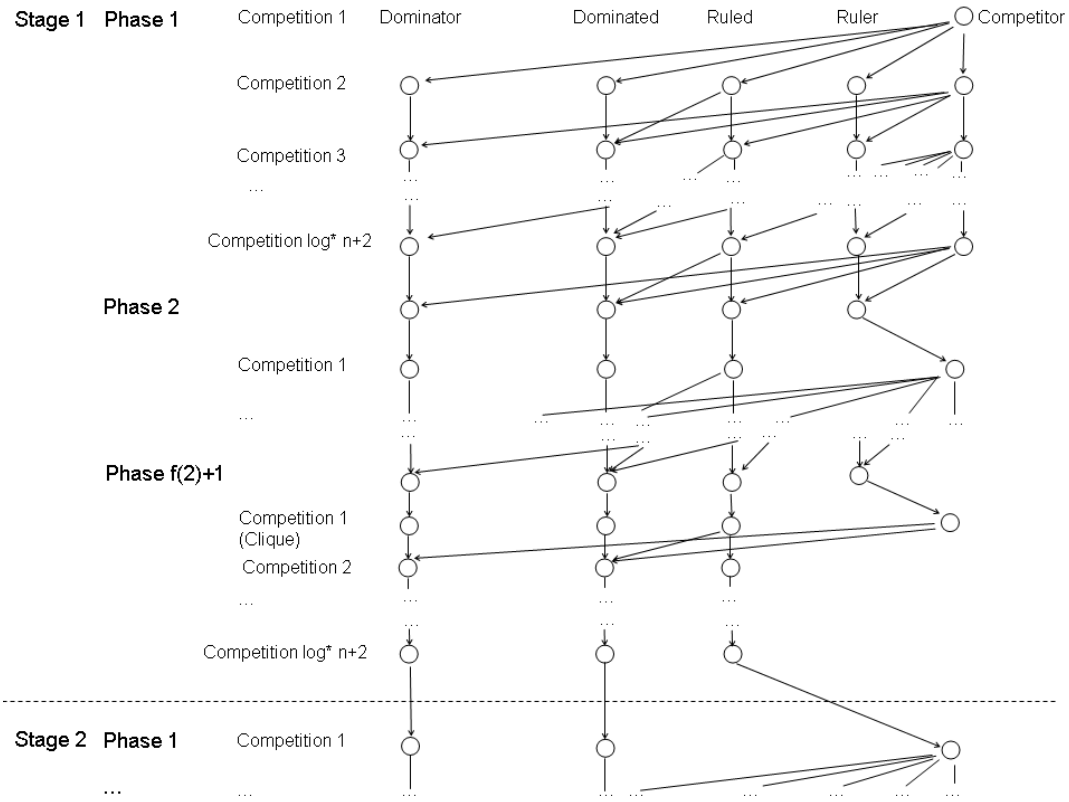

Figure 2: Illustration of stages, phases and competitions.

5.3 δ+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 Figure 3.

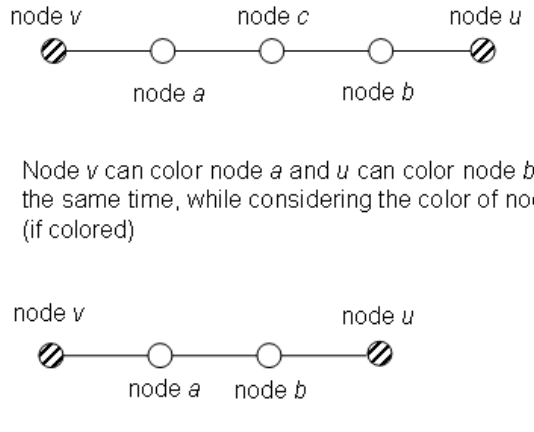

Figure 3: Illustration showing that the distance between two nodes concurrently coloring their neighbors must be at least 4

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 an 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 growth-bounded with $f'(r) \leq f(3 \cdot r)$, yielding an overall running time of $O(\log^* n)$. Due to Linial's $\Omega(\log^* n)$ [15] 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 $E' = \{(u,v)|u,v \in S \wedge \big((\exists s \in (V \setminus S)(\{(u,s),(s,v)\} \subseteq E) \vee (\exists s,t \in (V \setminus S)(\{(u,s),(s,t),(t,v)\} \subseteq E)\big)\}$. 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. ANALYSIS

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 at the end. Then we focus on termination and give evidence that a phase ends after at most $\log^* n + 2$ competitions. In other words no node can be a competitor for more than $\log^* n + 2$ competitions. In addition we prove that after every phase some nodes near a competitor v stop competing with v for the rest of the stage. As a next step, we prove that in the $f(2)^{st}$ phase every competing node must end up in exactly one clique of rulers and thus in the next phase the node in the clique with smallest ID becomes a dominator. Putting all things together yields that after a stage every non-dominated node and non-dominator has another dominator within hop distance $f(2)+3$. Since dominators are independent and the number of independent nodes within distance $f(2)+3$ is constant, it follows that after a constant number of stages the MIS is computed.

LEMMA 3. *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 after competition j, no neighbor can become a dominator or a ruler later. This follows from the fact that dominators don't alter their states and due to Algorithm Update State (line 12) for all neighbors $u \in N(v)$ holds that s_u = dominated and 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 next lemma bounds the number of competitions per phase and furthermore says that all competitors must change their states during a phase (as illustrated in Figure 2).

LEMMA 4. *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 requiring 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 $(r_v^{\log^* n+1} = 0) \vee (\forall u \in N(v)$ holds $r_u^{\log^* n+1} = 1)$ 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 to 14) no node turns from a non-competitor into a competitor, the lemma follows. □

The upcoming Lemma 5 essentially guarantees that phases and stages are locally synchronous, i.e. for a node v no passive (i.e. ruled) neighbor $u \in N(v)$ can compete in a phase or start a new stage, as long as node v is still active in the current phase (i.e. it is a competitor) or stage (i.e. it is a ruler or competitor).

LEMMA 5. *As long as v is active (i.e. it is a competitor or ruler) in some stage, neither a neighbor can start a new stage nor can a passive (i.e. ruled) neighbor become active (i.e. compete in some phase) during that stage.*

PROOF. Consider the first competition $j > 1$ of some phase, where node v, which has not taken part in competition $j-1$, competes with a node u, which has participated in competition $j-1$. Assume v took part in a prior competition i with $i < j$ in the same phase. Then v could not have passed the condition in line 14 in Algorithm MIS as long as u stays a competitor. Assume v did not participate in an earlier competition i with $i < j$. Due to the local synchronizer (line 9 in Algorithm Update State), i.e. all non-dominated and non-dominators wait until they have received the state of all non-dominated and non-dominators neighbors, v knows that u is a competitor after the first competition of the phase. Thus it cannot pass the condition in line 14 in Algorithm MIS and is not able to start a new phase.

Once a node v becomes a ruler, no neighbor can remain a competitor in Algorithm Update State. A dominated node does not change its state any more. All neighboring rulers pass the condition in line 14 in Algorithm MIS simultaneously with v but fail the condition in line 15 and change their states in line 5 to competitor. Thus they will not start a new stage. Any ruled node $w \in N(v)$, which also passes the condition in line 14, must fail in line 15, since neighbor v is a ruler. Thus it cannot start a new stage. Additionally, it cannot become a competitor (line 2) and be active in a later phase. Any ruled node $w \in N(v)$ that did not pass the condition in line 14 will get to know that neighbor v became competitor again and cannot pass the condition in line 14 as long as neighbor v is a competitor. Thus node $w \in N(v)$ can only start a new stage once all neighbors $N(w)$, including node v, have state ruled. □

DEFINITION 6. *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 $v \in U$. Let U_v^i be such a set in the beginning of phase i with $v \in U_v^i$.*

Initially, node v starts with all nodes at the same time, i.e. $U_v^0 = V$. For $i > 0$ Lemma 7 shows that the set U_v^i must have been a connected set of rulers (of maximal size) at the end of phase $i-1$. Furthermore, they must must have become rulers in the same competition of the same phase. Lemma 9 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 11 shows that in each phase some nodes near every ruler stop competing with it.

LEMMA 7. *All nodes from U_v^i for every $v \in V$ with $i > 0$ must have become rulers in the same competition during phase $i-1$ of a stage. Furthermore, no ruler $w \notin U_v^i$ is a neighbor of a ruler $u \in U_v^i$, i.e. U_v^i must be of maximal size.*

PROOF. In case there exist two nodes from U_v^i that became rulers in a different competition of some phase or in the same competition but in a different phase, then since U_v^i is connected there must exist two neighbors $u, w \in U_v^i$ with $u \in N(w)$ for which this is also true.

Due to Lemma 5 adjacent nodes execute phases and stages in a synchronous fashion, i.e. neighbors u, w can only start a new phase or stage together or one of them has to wait while the other is executing. Thus nodes u, w must execute the same phase. Assume node u became a ruler in the same phase as $u \in N(w)$ but in an earlier competition. Then node w would have become ruled and could not alter its state to competitor in that stage.

Observe that when a node u becomes a ruler, all neighbors either become ruled, dominated or rulers themselves in phase $i-1$. All rulers will pass the condition in line 14 Algorithm MIS but fail the condition in line 15 and thus start a new phase together. Therefore there cannot be a ruler $w \notin U_v^i$ with a neighbor $u \in U_v^i$, since w would have become a competitor in phase i and due to the maximality of U_v^i it would have to hold that $w \in U_v^i$. Since all ruled and dominated nodes will not be active (i.e. compete) in the following phases of that stage, no others node can be added to U_v^i. □

DEFINITION 8. *A node $u \in V$ can be reached by a path p of rulers from v, if $\exists p = (v = t_0, t_1, ..., u = t_q)$ s.t. $\forall (0 \leq i < q)$ holds $s_{t_i} = ruler\}$*

LEMMA 9. *If nodes U_v^i with $i > 0$ became rulers in competition j in phase $i-1$ then the results r_v^j and r_u^j for a node $u \in U_v^i$ as well as the prefixes of r_v^{j-1} and r_u^{j-1} are the same, i.e. $y_v^i = y_u^i$ for $r_v^j < i \leq \log n$.*

PROOF. Assume u was reachable by the path $p = (v = t_0, t_1, ..., u = t_q)$ of rulers from v and $r_v^j \neq r_u^j$. Due to maximality of U_v^i (Lemma 7) all rulers t_i are in U_v^i, i.e. $t_i \in U_v^i$ for $0 \leq i < q$. By assumption there would have to exist two neighboring rulers $t_l, t_{l+1} \in U_v^i$ with $r_{t_l}^j \neq r_{t_{l+1}}^j$. 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 contradicting Lemma 7. Assume their prefixes differed, i.e. $y_v^i \neq y_u^i$ for $r_v^j < i \leq \log n$. 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 for the rest of the stage. Since there are at most $f(2)$ of such nodes w (see Lemma 14) 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 15). This implies that no node can become a ruler in the last phase of a stage (see Figure 2).

DEFINITION 10. *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. $W_v^i := (N^2(v) \setminus N(v)) \cap U_v^i$.*

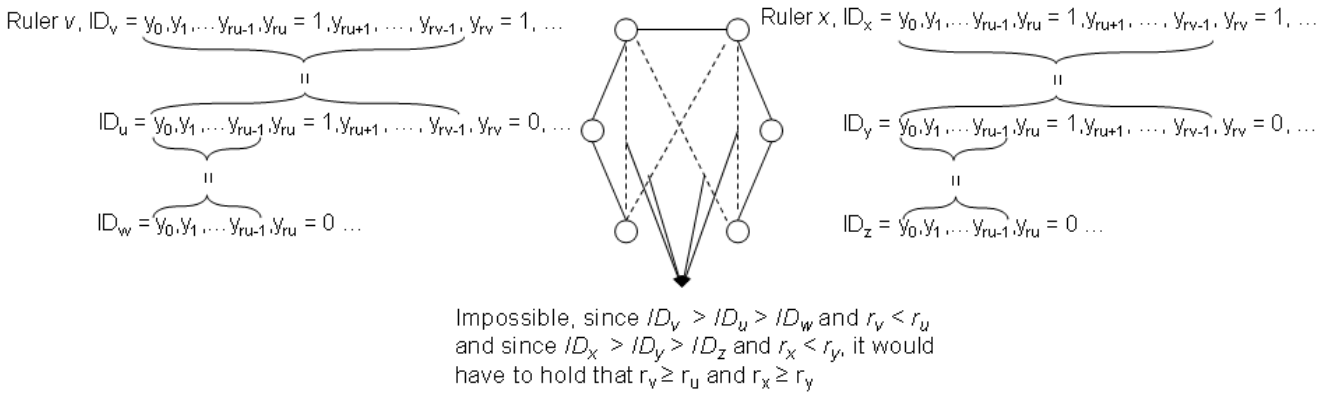

Figure 4: Graph of some nodes that participated with node v in the first competition

LEMMA 11. *Consider a node v, which has become a ruler in the j^{th} competition in phase i. If $|W_v^i| > 0$ then $\exists w \in W_v^i$, which cannot be reached by a path of rulers from v.*

PROOF. The proof will be done by induction. Let us investigate the first competition, which is based on IDs.

Consider the value of r_v^1. 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 r_v^1 cannot be a ruler. So assume $r_v^1 \in [1, \log n - 1]$.

Due to Lemma 9 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 $r_v^j < i \leq \log n$ 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 $r_u^1 \neq r_v^1$. 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 $r_u^1 < i \leq \log n$ 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 with value $r_x^1 = r_v^1$, since otherwise $r_v^1 \geq r_u^1$ 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 Figure 4.

Let us look at the j^{th} competition. 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 $0 < r_v^j \leq \log^{(j)} n$ 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 round $j-1$ and v in round j, then due to Lemma 7 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. See Figure 5. □

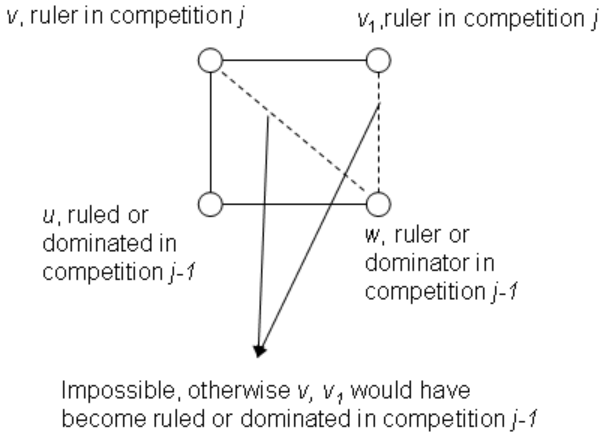

Figure 5: Graph of some nodes that participated with node v in competition $j-1$ and j

Lemma 12 shows that every connected set of competitors in phase j must be a subset of a previous connected set of competitors from phase i with $i < j$. This will be used by Lemma 13 to show that if a set of arbitrary nodes does not have a common node with a set of connected competitors in some phase, then this will hold for all later phases of the stage.

LEMMA 12. *If $U^i \cap U^j \neq \{\}$ with $i < j$ for some stage, then $U^j \subseteq U^i$.*

PROOF. For $i = 0$, we have that $U^i = V$ and the lemma follows. Once the nodes from U^i with $i > 0$ became rulers in phase $i-1$, all nodes $v \in N(u)$ with $u \in U^i$ became ruled or dominated and thus cannot become a competitor in that stage and therefore cannot be in any set U^j for phase j with $i < j$ of that stage. Thus a competitor $v \in U^i \cap U^j$ cannot reach a competitor $t \notin U^i$ by a path of competitors, since

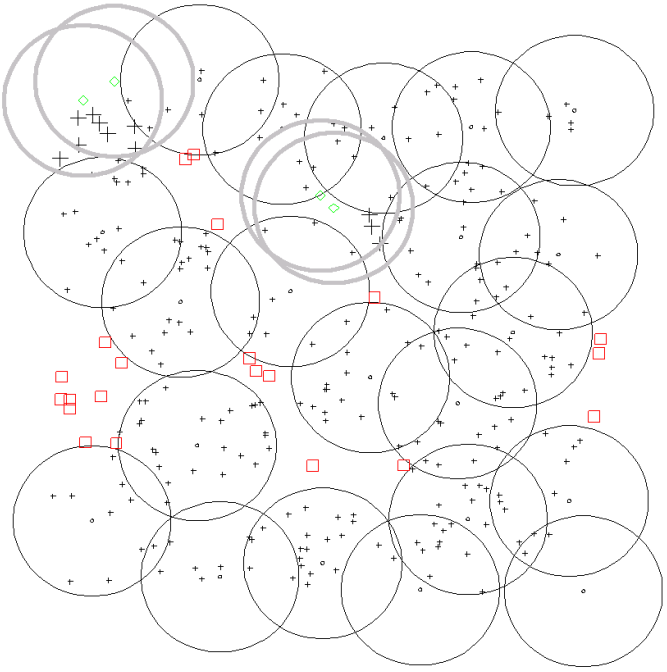

Figure 6: Algorithm MIS on an instance of a UDG. It shows the state of each node after the very first competition. For dominators and rulers a circle indicating the transmission range is shown. Rulers are depicted by green diamonds, ruled nodes by big crosses, dominated nodes by small crosses, dominators by small circles and competitors by red 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 second stage the algorithm is done.

the path would contain a ruled or dominated node $v \in N(u)$ with $u \in U^i$. Thus if $U^i \cap U^j \neq \{\}$, then $U^j \subseteq U^i$. □

LEMMA 13. *For a set $T \subset V$, s.t. $U^i \cap T = \{\}$ holds that $U^j \cap T = \{\}$ with $i < j$.*

PROOF. Due to Lemma 12, we have that $U^j \subseteq U^i$. Due to the independence of U^i and T, U^j and T must also be independent. □

The next two lemmas together give an upper bound of the number of phases of a stage.

LEMMA 14. *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 11, for phase i be denoted by $w_i \in W_v^i$. Lemma 11 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 13, no node $t \in N(w_i) \cup w_i$ will be reachable by a path of competitors from s with $s \in U_v^i$ for the rest of the stage. Since $W_v^i \subseteq U_v^i$, this implies that $W_v^{i+1} \subseteq (W_v^i \setminus (N(w_i) \cup w_i))$. 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. the nodes $U_v^{f(2)+1}$ form a clique. □

LEMMA 15. *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.*

PROOF. Using Lemma 14, 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. □

Lemma 15 implies that after a stage no node can be a ruler or competitor. Because of this and since a node cannot become ruled (or dominated) without having a ruler (or dominator) in its neighborhood (see Algorithm Update State), it follows that after a stage at least one competing node must become a dominator. Next we show that for every non-dominated node v a dominator is chosen within constant distance from v. Since dominators are independent (see Lemma 3) and in a growth bounded graph the number of independent nodes within constant distance is also a constant, only a constant number of stages are needed (see Lemma 17).

LEMMA 16. *After a stage, a node v is either a dominator or there exists a node that has become a dominator in that stage within hop distance $f(2)+2$.*

PROOF. We will show that the distance between a ruler in phase i and node v is at most i. The proof will be done by induction: 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 i^{th} phase only rulers become competitors again

(line 5) and participate in the competitions. Thus after the i^{th} 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 15 every competitor or one of its neighbors must become a dominator in the $(f(2)+1)^{st}$ phase. □

LEMMA 17. *After at most $f(f(2)+3)$ stages, i.e. executions of the repeat loop (lines 1 to 16), the computation of the MIS has finished.*

PROOF. Assume node v is ruled. Then due to Lemma 16 after at most $f(2)+2$ phases node v or some neighbor(s) $u \in N(v)$ start a new stage, i.e. all neighbors of v or u (or of both) are ruled. Due to Lemma 16 a dominator is chosen for any node that has competed in a stage within distance $f(2)+2$, implying a distance of up to $f(2)+3$ for v. Since dominators are independent (Lemma 3), 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 at most $f(f(2)+3)$ stages are needed. □

THEOREM 18. *The total time to compute a MIS is in $O(\log^* n)$ and each message is of size $O(\log n)$.*

PROOF. 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. This yields together with Lemma 4 that during one phase $O(\log^* n)$ rounds of communication have to take place because $\sum_{i=1}^{\log^* n+2} \log^{(i)} n \leq \log n + \log\log n + ... + 2^{2^2} + 2^2 + 2 + 1 \leq O(\log n)$. Since a stage consists of $f(2)+1$ phases (see Lemma 15) and after $f(f(2)+3)$ stages (see Lemma 17) the algorithm is done, the lemma follows. □

7. 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 growth bounded 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 Figure 6 was completed after 3 rounds of communication only. Our algorithm is applicable in various settings beyond the MIS problem, for instance for computing a δ+1 coloring or an asymptotically optimal CDS in $O(\log^* n)$.

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 Section 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 nonexistent!) 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 with-collision-detection model [19] uses the local algorithm presented in this paper as a building block. For a MIS algorithm in a wireless model without collision detection see, e.g. [17].

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 stages/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!

8. REFERENCES

[1] N. Alon, L. Babai, and A. Itai. A fast and simple randomized parallel algorithm for the maximal independent set problem. *J. Algorithms*, 7(4):567–583, 1986.

[2] K. Alzoubi, X. Li, Y. Wang, P. Wan, and O. Frieder. Geometric Spanners for Wireless Ad Hoc Networks. *IEEE Transactions on Parallel and Distributed Systems*, 14(5), 2003.

[3] A. Andersson, T. Hagerup, S. Nilsson, and R. Raman. Sorting in Linear Time. *Journal of Computer and System Sciences*, 57:74–93, 1998.

[4] B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proceedings of the 30 th Symposium on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.

[5] G. V. B Awerbuch. Distributed program checking: a paradigm for buildingself-stabilizing distributed protocols. In *Proceedings of the 32nd Annual Symposium on Foundations of Computer Science (FOCS)*, 1991.

[6] R. Cole and U. Vishkin. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Inf. Control*, 70(1):32–54, 1986.

[7] B. Gfeller and E. Vicari. A Randomized Distributed Algorithm for the Maximal Independent Set Problem in Growth-Bounded Graphs. In *Proc. of the 26^{th} ACM Symposium on Principles of Distributed Computing (PODC)*, 2007.

[8] A. Goldberg, S. Plotkin, and G. Shannon. Parallel Symmetry-Breaking in Sparse Graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 1(4):434–446, 1988.

[9] A. Israeli and A. Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Information Processing Letters*, 22:77–80, 1986.

[10] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast Deterministic Distributed Maximal Independent Set Computation on Growth-Bounded Graphs. In *Proc. of the 19^{th} Int. Symposium on Distributed Computing (DISC)*, 2005.

[11] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Local Approximation Schemes for Ad Hoc and Sensor Networks. In *Proc. of the 3^{rd} ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2005.

[12] F. Kuhn, T. Moscibroda, and R. Wattenhofer. On the Locality of Bounded Growth. In *Proceedings of the 24^{th} ACM Symp. on Principles of Distributed Computing (PODC)*, pages 60–68, 2005.

[13] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! In *Proc. of the 23rd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 300–309, 2005.

[14] C. Lenzen and R. Wattenhofer. Leveraging lineal's locality limit. In Submission.

[15] N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[16] M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[17] T. Moscibroda and R. Wattenhofer. Maximal Independent Sets in Radio Networks. In *Proceedings of the 24^{th} ACM Symp. on Principles of Distributed Computing (PODC)*, pages 148–157, 2005.

[18] A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. of the 24th annual ACM symposium on Theory of computing (STOC)*, pages 581–592. ACM Press, 1992.

[19] J. Schneider and R. Wattenhofer. An Asymptotically Optimal Distributed Maximal Independent Set Algorithm for Wireless Networks with Collision Detection. In Submission.