A randomized distributed algorithm for the maximal independent set problem in growth-bounded graphs

A Randomized Distributed Algorithm for the Maximal Independent Set Problem in Growth-Bounded Graphs

ETH Technical Report 551

Beat Gfeller Elias Vicari

ETH Zurich,
Institute of Theoretical Computer Science,
8092 Zurich, Switzerland.

{gfeller,vicariel}@inf.ethz.ch.

February 2007

Abstract

The efficient distributed construction of a maximal independent set (MIS) of a graph is of fundamental importance. We study the problem in the class of Growth-Bounded Graphs, which includes for example the well-known unit disk graphs. In contrast to the fastest (time-optimal) existing approach [11], we assume that no geometric information (e.g., distances in the graph's embedding) is given. Instead, nodes employ randomization for their decisions. Our algorithm computes a MIS in $O(\log\log n \cdot \log^* n)$ rounds with very high probability for graphs with bounded growth, where n denotes the number of nodes in the graph. In view of Linial's $\Omega(\log^* n)$ lower bound for computing a MIS in ring networks [12], which was extended to randomized algorithms independently by Naor [18] and Linial [13], our solution is close to optimal.

In a nutshell, our algorithm shows that for computing a MIS, randomization is a viable alternative to distance information.

1 Introduction and Related Work

The question of how to compute a maximal independent set (MIS) in a distributed setting has been studied for many years [4, 14, 2, 13]. This problem asks for a subset S of the vertices of a graph with the property that no two vertices of S share an edge and such that there is no strict superset of S with the same property. Its importance stems from the fact that it serves as a basic building block for many distributed algorithms, and that it captures the essence of symmetry breaking. Furthermore it is a nice example of a graph-theoretical problem that admits a trivial greedy solution in a centralized setting, but is nevertheless a challenging research topic for the distributed computation community.

With the advent of wireless ad hoc and sensor networks, the study of distributed MIS computation has received further attention [17, 21]. This is because algorithms for topology control and routing often construct a MIS in an initial phase [6, 5, 1, 22]. Typically, the time required for this phase dominates the total running time for the algorithms. For example, constructing a $(1+\varepsilon)$-approximate minimum dominating set is possible in time $O(T_{\mathsf{MIS}} + \log^* n/\varepsilon^{O(1)})$ in Growth-Bounded

Graphs, where T_{MIS} is the time for MIS computation [9]. Furthermore, a $(1+\varepsilon)$-approximate minimum connected dominating set, which can be used as a "virtual backbone" for routing, can be obtained in the same time [7], again in Growth-Bounded Graphs.

The energy constraints of wireless ad hoc networks ask for minimizing the computation time (and thus "active time" of the nodes) required to complete the desired tasks. This is further encouraged by the fact that ad hoc networks may frequently change their structure. To model the particular connectivity structure inherent to wireless networks, usually a restricted class of graphs (as opposed to general graphs) is considered. By far the most prominent such class are Unit Disk Graphs (UDG). However, for various reasons UDGs are a very idealized model of real wireless networks, and many generalizations have been proposed [19, 23]. For this paper, we adopt the class of Growth-Bounded Graphs, which seems relatively general since it contains UDGs and many other models as special cases.

Various approaches for distributively computing a MIS in UDGs is available. However, many of them have a worst case running time of $O(n)$, e.g. [1]. As an interesting variant, [11] considers the distributed computation of a MIS in a setting where nodes are embedded in a metric space, and some geometric information about the embedding is available to the nodes. It is shown that if each node knows the distance to each of its neighbors, one can compute a MIS in $O(\log^* n)$ rounds. If nodes are located in the Euclidian plane, and each node knows its position, a MIS can even be computed in constant time.

Thus, very fast running times can be achieved under the above assumptions. However, it seems debatable whether these features are implementable in a practical scenario. It may be that the energy they require is unacceptable for the wireless devices at hand, or these features (e.g. measuring distance) may simply not be available, due to hardware constraints or the type of deployment of the sensors. In any case, a distance measuring device might increase the hardware costs considerably.

In this respect, one might want to compute a MIS without using these features, yet being similarly fast. We therefore propose to use randomization instead of distance measuring. This seems like a natural choice, as intuitively, randomization should ease symmetry-breaking. Moreover, randomization has proved successful in computing a MIS in general graphs: While the fastest known deterministic algorithm for general graphs due to Panconesi and Srinivasan [20] runs in $O(n^{d\sqrt{1/\log n}})$ (where d is a constant), Luby proposed an almost exponentially faster $O(\log n)$ randomized algorithm [14]. This is close to the $\Omega(\sqrt{\log n/\log\log n})$ lower bound given in [10]. The question whether a polylogarithmic deterministic algorithm for the MIS problem exists is still open.

In contrast, a deterministic MIS algorithm with running time $O(\log\Delta\log^* n)$ exists for Growth-Bounded Graphs [8], while the best known lower bound for this class is $\Omega(\log^* n)$ (even for randomized algorithms) [18, 13]. This raises the question whether randomization enables a running time improvement of the same magnitude as in general graphs. Our results, summarized in the next section, show that using randomization, one can indeed compute a MIS in Growth-Bounded Graphs almost as quickly as with the help of distance information.

Contribution

Our main contribution is a synchronous randomized distributed algorithm for computing a MIS in Growth-Bounded Graphs with n vertices running in $O(\log\log n\log^* n)$ rounds with high probability. Specifically, for any fixed $k \geq 1$ the probability that the algorithm requires more than $O(\log\log n\log^* n)$ rounds is at most $1/n^k$ (k affects the constant factor hidden in the asymptotic notation). The nodes require only *connectivity* information about the graph, and all messages are of size $O(\log n)$.

This running time compares on one side with the $O(\log \Delta \log^* n)$ running time of the best known deterministic algorithm for Growth-Bounded Graphs [8], and on the other side with the $O(\log^* n)$ running time of the deterministic algorithm which assumes that the graph is embedded in a metric space, and requires the nodes to know the geometric distance to their neighbors [11]. In view of the $\Omega(\log^* n)$ lower bound for randomized algorithms [18, 13], our solution is close to optimal.

It is worth noting that our algorithm achieves the fastest known constant-factor approximation for the Dominating Set Problem in Growth-Bounded Graphs, since any MIS is at most five times larger than an optimal dominating set in this class of graphs [15].

The main technical novelty introduced by this paper is a new randomized algorithm to find a $2t$-ruling set with low induced maximum degree in $O(t)$ rounds. Even in general graphs, our approach finds a $O(\log \log \Delta)$-ruling set with induced degree $O(\log^5 n)$ in $O(\log \log \Delta)$ rounds (with high probability), which might be of independent interest. In each step of this algorithm, a subset of the nodes is selected. This subset consists of three different sets, which are designed such that their combination guarantees the desired ruling-property and a rapidly decreasing maximum degree. Two of these sets are chosen deterministically, and the third set is chosen randomly using an approach resembling Luby's algorithm, but with a different choice of probabilities.

2 Model and Definitions

2.1 Terminology

A network is modeled as an undirected simple graph $G = (V, E)$ on n nodes. For $V' \subseteq V$, we denote by $G[V']$ the subgraph of G *induced* by V': the vertex-set of $G[V']$ is V' and the edge-set consists of the edges of G with both endpoints in V'. The *distance* between two nodes $u, v \in V$ is the number of edges of a shortest path connecting u and v in G and is denoted by $d(u, v)$. This definition extends naturally to the distance between a node and a set, or between two sets. Let the *neighborhood* of a node $u \in V$ be $N(u) := \{u\} \cup \{v \in V \mid (u, v) \in E\}$. The *neighborhood* of a set $S \in V$ is defined as $N(S) := \{u \in V \mid d(u, S) \leq 1\}$. Note that this includes S itself. We define the *i-neighborhood* $N_i(S)$ of S recursively as $N_0 := S$ and $N_i(S) := N(N_{i-1}(S))$ for $i > 0$. The *size* d_v of a node $v \in V$ is the size of $N(v)$, that is, the degree of v plus one[1]. Δ denotes the maximum degree of any node in G. For convenience, we define $\overline{\Delta} := \Delta + 1$.

A set $T \subseteq V$ is said to be *independent* in G if no two nodes $u, v \in T$ are neighbors in G. An independent set T is a *Maximal Independent Set* of G (MIS), if no superset $T' \supset T$ is independent in G. Further, T is a *k-ruling set* if every node of G is within distance k from some node of T. Note that a MIS is a 1-ruling set.

A graph G is a *Growth-Bounded Graph* if there is a class of graphs $\mathcal{C}$ and a function $f_{\mathcal{C}}$, such that every r-neighborhood $N_r(u)$ of any node u in G has at most $f_{\mathcal{C}}(r)$ independent nodes. Note that the function $f_{\mathcal{C}}$ must solely depend on r and $\mathcal{C}$. While discussing the asymptotic running time of the algorithms, we assume that each input graph G belongs to the same class $\mathcal{C}$ (for example all Unit Disk graphs).

Throughout the paper, we use $\log x$ to denote the binary logarithm of x, and $\ln x$ for the natural logarithm of x.

[1] This $+1$ increment leads to simpler terms in our analysis, but is not of particular importance.

2.2 Message-Passing Model

In this paper we employ the synchronous *message-passing* model as a framework for our algorithm. The wireless ad hoc network is modeled as a Growth-Bounded Graph. Two nodes can directly communicate if an edge is present between the corresponding nodes in the graph. In each round, each node can send a message of size $O(\log n)$ bits to each of its neighbors. Messages between distant nodes must be forwarded via intermediate nodes. The nodes are solely provided with local *connectivity* information, i.e. any node knows only about its direct neighbors. Consequently, they may not know the full network topology (nor the number of nodes) and cannot sense the (local) embedding of the network. For instance, they cannot measure distances between nodes.

The complexity of an algorithm is the number of rounds required from its start until its completion in the worst-case (with respect to the network topology and to the assignment of the identifiers to the nodes). Assuming an already established physical layer, we do not consider collision and interference issues.

3 Distributed MIS algorithm

In this section, we present Algorithm MaxIndepSet, which computes a MIS in Growth-Bounded Graphs in $O(\log\log n \log^* n)$ rounds, with very high probability. First, we outline the structure of the algorithm.

In the first phase of Algorithm MaxIndepSet, a subset $T \subseteq V$ of G's nodes is selected, such that

- T is a $O(\log\log n)$-ruling set of G, and
- the subgraph induced by T has a maximum degree of $O(\log^5 n)$.

This set is obtained by repeatedly applying the Algorithm RandStep (see below), using V as the initial set. Then in a second phase, the set T is further thinned out, such that the remaining set of nodes T' is an *independent* $O(\log\log n)$-ruling set of G. As in the first phase, this is achieved by repeatedly selecting particular subsets of the previous set (starting with T). Finally, in a third phase, this sparse independent set T' is extended into a maximal independent set.

The main novelty of our approach lies in the first phase of the algorithm, where the *logarithm* of the maximum degree induced by the remaining nodes decreases *geometrically* in each step, using randomization. For the second and the third phase, we use two deterministic algorithms from [8].

Note that the transition from the first phase to the second is triggered by a threshold of $O(\log^5 n)$ for the maximum degree of the remaining graph. Not knowing n, the nodes cannot know within reasonable time (i.e., locally) when the maximum degree has been reduced enough. This is why we "interleave" the executions of the algorithms for the first two phases. We will argue that this does not harm the effectiveness of either phase.

The next sections describe the building blocks of our algorithm in more detail. We will show that each phase requires at most $O(\log\log n \log^* n)$ rounds (Phase 1 uses $O(\log\log\Delta)$ rounds with high probability), leading to the following result.

Theorem 1. *There exists a randomized algorithm that computes a MIS for any Growth-Bounded Graph G within $O(\log\log n \log^* n)$ rounds (with high probability) of synchronous distributed computation in the message-passing model. All messages are of size $O(\log n)$.*

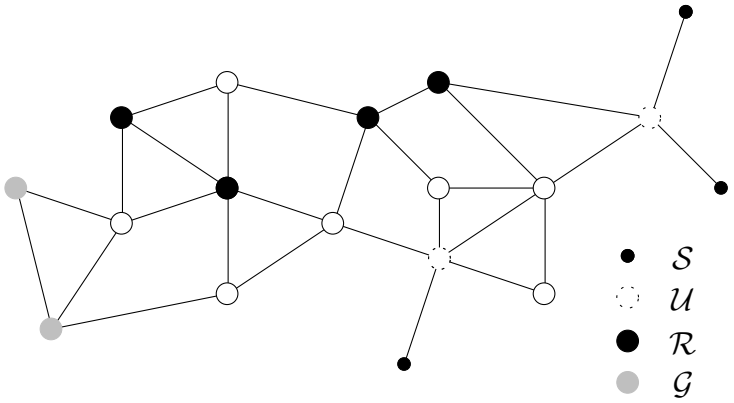

Figure 1: Illustration of Algorithm RandStep. The sets $\mathcal{S}$, $\mathcal{B}$ and $\mathcal{G}$, which are drawn as filled disks, remain in the graph. Their union $\mathcal{S} \cup \mathcal{B} \cup \mathcal{G}$ is a 2-ruling set of G.

3.1 Phase 1: A $O(\log\log n)$-ruling set with small induced degree

This phase consists of repeated applications of Algorithm RandStep: Let T_1 be the set of nodes returned by the first execution of RandStep. This set induces a graph $G[T_1]$, on which RandStep is applied again, yielding T_2 and a new graph $G[T_2]$. In this way, we obtain sets T_i which are increasingly sparse, while at the same time remaining $O(\log\log n)$-ruling.

The central idea behind Algorithm RandStep is to choose a random subset of all nodes, such that the maximum degree of the induced subgraph decreases rapidly, i.e., from Δ to Δ^c with $c < 1$, but assuring that the remaining set of nodes is a 2-ruling set of the previous graph. The key question then is how to choose the probabilities such that nodes with large degree will have small degree afterwards: If all nodes have the same degree d, and each node decides independently with probability $p = 1/d^{1/4}$ whether to stay in the set, then the expected degree is $d^{3/4}$. This achieves that the maximum degree (or, correspondingly, size) decreases from Δ to Δ^c with high probability, and furthermore the probability that a node *and all* of its neighbors leave the set is rather low. Clearly, when nodes have different degrees, they will use different probabilities, so the above reasoning does not work. As we will see however, if the degrees of neighboring nodes do not differ too much, then one can still obtain a similarly fast decreasing maximum degree (with high probability).

Algorithm 1: RandStep

Input: A Growth-Bounded Graph $G = (V, E)$.
Output: A 2-ruling set $T \subseteq V$ of V.

1 $\mathcal{R} := \emptyset$
2 $\mathcal{S} := \{u \in V \mid d_u^2 < d_v \text{ for some } v \in N(u)\}$
3 $\mathcal{U} := N(\mathcal{S}) \backslash \mathcal{S}$
4 $\mathcal{B} := V \backslash \mathcal{U}$ ($\mathcal{B}$ is the set of black nodes)
5 Each $u \in \mathcal{B}$ independently joins $\mathcal{R}$ with probability $p = \frac{1}{d_u^{1/4}}$. ($\mathcal{R}$ is the set of red nodes)
6 $\mathcal{G} := \mathcal{B} \backslash (N(\mathcal{R}) \cup N(\mathcal{U}))$ (set of green nodes)
7 **return** $T := \mathcal{S} \cup \mathcal{R} \cup \mathcal{G}$

Algorithm RandStep works as follows (see Figure 1): First, nodes which have a neighbor with size much larger than their own know that they have a relatively low size, so they can simply stay in the graph for the next round (this is the set $\mathcal{S}$ in the algorithm). Additionally, their neighbors

(set $\mathcal{U}$ in the algorithm) can safely leave the graph because they are sure to have a neighbor who stays. All other nodes then have only neighbors with similar sizes. We refer to these nodes as black nodes and denote them by the set $\mathcal{B}$. A black node u becomes red (and stays in the graph) with probability $p = 1/d_u^{1/4}$. The respective set is called $\mathcal{R}$ in Algorithm RandStep. Finally, with low probability, there may be black nodes for which no node within distance two stays in the graph. In order to guarantee the 2-ruling property (in contrast to merely knowing a high probability bound for it), we add these nodes to the set of green nodes $\mathcal{G}$ which also stays in the graph[2].

For convenience, the algorithm is formulated in a global fashion, but a distributed version, where each node can execute these steps by communicating only with its direct neighbors, is immediate.

Analysis

In the following, we prove that with high probability, iterating Algorithm RandStep for $O(\log\log n)$ times yields a $O(\log\log n)$-ruling set whose induced subgraph has a maximum degree of $O(\log^5 n)$.

First, we examine the ruling property of Algorithm RandStep when applied repeatedly.

Lemma 2. *After iterating Algorithm* RandStep *t times, the obtained set T_t is a $2t$-ruling set.*

Proof. Consider an iteration from T_i to T_{i+1}. Note that after every execution i of Algorithm RandStep, the sets $\mathcal{S}$, $\mathcal{U}$ and $\mathcal{B}$ form a partition of T_i. Any node in $\mathcal{S}$ stays in the set. Any node in $\mathcal{U}$ is dominated by some node in $\mathcal{S}$. $\mathcal{R} \cup \mathcal{G}$ is by construction a dominating set of $\mathcal{B}\backslash N(\mathcal{U})$. Thus, the only nodes in T_i that may not be dominated by $\mathcal{S} \cup \mathcal{R} \cup \mathcal{G}$ are those in $N(\mathcal{U})\backslash\mathcal{U}$. These nodes by definition have a 2-hop neighbor in $\mathcal{S}$. Thus, after each iteration, $\mathcal{S} \cup \mathcal{R} \cup \mathcal{G}$ is a 2-ruling set of T_i. The claim now follows by induction over i (as initially, $T_0 = V$ is a 0-ruling set). □

For the further analysis, we consider only one execution of Algorithm RandStep on the graph $G = (V, E)$. During the exposition and the analysis of the algorithms, the sizes and neighbors of a node are to be understood with respect to the current graph. In the analysis we choose the constants quite arbitrarily to yield simple terms. This only affects constant factors in the running time.

The first lemma states that the sizes of neighboring nodes in $\mathcal{B}$ do not differ too much.

Lemma 3. *For any $u \in \mathcal{B}$, and for each $v \in N(u)$, we have*

$$(d_v)^{1/2} \leq d_u \leq (d_v)^2.$$

Proof. By the lemma's assumption, $u \notin \mathcal{S}$ and thus $v \notin \mathcal{S}$. As $u \notin \mathcal{S}$, we have $d_u^2 \geq \max_{q \in N(u)} d_q \geq d_v$, so the first inequality follows. Similarly, the second inequality holds by exchanging u and v. □

Lemma 4. *The probability that a node $u \in \mathcal{B}$ with $d := d_u \geq k^2 \ln^2 n$ becomes green is at most $\frac{1}{n^k}$, for $k > 1$.*

Proof. Let $u \in \mathcal{B}$ be a node with the property that $N(u)$ consists of only black nodes. Only such nodes may potentially become green. If, after the algorithm's execution, one of the nodes in $N(u)$ becomes red then u does not become green.

[2]For ensuring the 2-ruling property, it would suffice to define $\mathcal{G} := \mathcal{B}\backslash(N_2(\mathcal{R}) \cup N(\mathcal{U}))$, but replacing $N_2(\mathcal{R})$ by $N(\mathcal{R})$ simplifies the analysis.

As the $d-1$ neighbors are all black, their sizes are all at most d^2 by Lemma 3. Thus, each of them becomes red with probability at least $\frac{1}{d^{1/2}}$. The probability that neither u nor any of its $d-1$ neighbors become red is therefore at most

$$\left(1-d^{-1/2}\right)^d \leq e^{-d^{1/2}},$$

using the basic inequality $1-x \leq e^{-x}$. So for $d \geq k^2 \ln^2 n$, we have $P[u \text{ becomes green}] \leq \frac{1}{n^k}$. □

As we show in the following Lemmas, as long as the maximum degree Δ of G is at least $\Omega(\log^5 n)$ one iteration of RandStep almost surely decreases $\overline{\Delta}$ of the graph to $2\overline{\Delta}^{7/8}$ or below.

Lemma 5. *For any $k > 1$, the probability that a black node $u \in \mathcal{B}$ with $d \geq 9k^2 \ln^2 n$ has more than $2d^{7/8}$ red neighbors (including itself) is at most $\frac{1}{n^k}$.*

Proof. Recall that each black node v becomes red with probability $\frac{1}{d_v^{1/4}}$, independent of the choices of its neighbors.

Let X be the number of red neighbors of u, plus one if u is red, i.e., $X := |N(u) \cap \mathcal{R}|$ (this possibly includes u itself). X is a random variable which is the number of successes in d independent *Poisson trials* (see e.g. [16]), where a "success" means that some neighbor of u (or u itself) joins $\mathcal{R}$. Each of these trials (of joining $\mathcal{R}$) may have a different success probability, but by Lemma 3 all neighbors of u (and u itself) have size at least $d^{1/2}$, so each neighbor of u (and u too) joins with probability at most $\frac{1}{d^{1/8}}$. Clearly, $E[X] \leq d \cdot \frac{1}{d^{1/8}} = d^{7/8}$. Using the Chernoff bound [3]

$$P[X \geq (1+\delta)E[X]] \leq e^{-E[X]\delta^2/3} \quad \text{for} \quad 0 < \delta \leq 1,$$

with $\delta = 1$ and $d \geq 9k^2 \ln^2 n$, we obtain

$$P\left[X \geq 2d^{7/8}\right] \leq e^{-\frac{1}{3}d^{7/8}} \leq e^{-(k \ln n)} \leq \frac{1}{n^k}.$$

□

Lemma 6. *The probability that a node $u \in \mathcal{B}$ with $d := d_u \geq 9k^4 \ln^4 n$ has more than $2d^{7/8}$ neighbors (including itself) in $\mathcal{R} \cup \mathcal{G}$ is at most $\frac{2}{n^{k-1}}$, $k > 1$.*

Proof. Let A be the event that u has more than $2d^{7/8}$ red neighbors, and let B be the event that u has any green neighbor. By Lemma 3, all neighbors of u have at least size $3k^2 \ln^2 n$, so by Lemma 4 every neighbor of u becomes green with probability at most $\frac{1}{n^k}$. From the union bound[3], it follows that the probability of u having any green neighbor is at most $\frac{1}{n^{k-1}}$, because u can have at most n neighbors.

Using this fact and Lemma 5 for the second inequality, we have

$$P[A \cup B] \leq P[A] + P[B] \leq \frac{1}{n^k} + \frac{1}{n^{k-1}} \leq \frac{2}{n^{k-1}}.$$

□

[3]The *union bound* upper bounds the probability that any of the events $E_1, E_2, \ldots, E_n$ occurs: $P\left[\bigcup_{i=1}^n E_i\right] \leq \sum_{i=1}^n P[E_i]$, even if these events are not independent (see e.g. [16]).

Lemma 7. *Let Δ be the maximum degree of the subgraph induced by T_i after $i \geq 0$ iterations of Algorithm* RandStep *(recall $\overline{\Delta} = \Delta + 1$). Assume that $\overline{\Delta} \geq 6k^5 \ln^5 n$, $k > 2$. After one more iteration, the maximum size of the subgraph induced by T_{i+1} is at most $2\overline{\Delta}^{7/8}$ with probability at least $1 - \frac{2}{n^{k-2}}$.*

Proof. Consider any node $u \in T_i$ with $d_u \geq 2\overline{\Delta}^{7/8}$. This node will only be in T_{i+1} after the next iteration if it is in $\mathcal{S} \cup \mathcal{R} \cup \mathcal{G}$. However, clearly u cannot be in $\mathcal{S}$ (because otherwise there would exist a node v with $d_v > d_u^2 \geq 4\overline{\Delta}^{14/8} \geq \overline{\Delta}$, a contradiction.). So u is either in $\mathcal{R}$ or in $\mathcal{G}$. Moreover, by construction any node in $\mathcal{B}$ has no neighbors in $\mathcal{S}$, so its size is the number of neighbors in $\mathcal{R} \cup \mathcal{G}$. As $d_u \geq 2\overline{\Delta}^{7/8} \geq 2(6k^5 \ln^5 n)^{7/8} \geq 9k^4 \ln^4 n$, Lemma 6 can be applied, so the size d'_u of u after the iteration satisfies $d'_u \leq 2\overline{\Delta}^{7/8}$ with probability at least $1 - \frac{2}{n^{k-1}}$. Hence the probability that any of the nodes with size $d \geq 6k^5 \ln^5 n$ has a size $\geq 2\overline{\Delta}^{7/8}$ is at most $\frac{2}{n^{k-2}}$, again using the union bound. □

The above bounds together yield a high-probability bound for the number of RandStep iterations required to reduce the maximum degree to polylogarithmic size. Note that this result holds for arbitrary graphs, not only for Growth-Bounded Graphs.

Theorem 8. *For any constant $k > 3$, and some suitable constant $c = c(k)$, when* RandStep *is run on any graph $G = (V, E)$ with maximum degree Δ, then with probability at least $1 - 2c/n^{k-3}$ the maximum degree in the subgraph of G induced by T_i is at most $6k^5 \ln^5 n$ after no more than $c \ln \ln \Delta$ iterations.*

Proof. We call an iteration (of RandStep) *successful* if the current maximum size of any node in the subgraph induced by T_i is reduced from $\overline{\Delta}$ to $2\overline{\Delta}^{7/8}$. Thus, after m successful iterations, the degree is at most

$$2^{1+7/8+(7/8)^2+\ldots+(7/8)^{m-1}} \cdot \overline{\Delta}^{(7/8)^m} = 2^8 2^{1-(7/8)^m} \overline{\Delta}^{(7/8)^m} \leq 512\overline{\Delta}^{(7/8)^m}.$$

In order to reduce the degree to below $6k^5 \ln^5 n$, m must satisfy

$$512\overline{\Delta}^{(7/8)^m} \leq 6k^5 \ln^5 n,$$

which holds for $m \geq c \ln \ln \overline{\Delta}$ for some constant $c(k)$.

By Lemma 7, each iteration is successful with probability at least $1 - \frac{2}{n^{k-2}}$ as long as the degree is at least $6k^5 \ln^5 n$. Since we require at most $c \ln \ln \overline{\Delta}$ successes and each iteration succeeds with probability at least $1 - \frac{2}{n^{k-2}}$, as long as the maximum degree is big enough (Lemma 7), the probability that *all* rounds are successful is high. Formally, let A_i be the event that round i is not successful. There are at most $c \ln \ln \overline{\Delta}$ iterations that could potentially fail. Hence, the probability that at least one of these iterations fails is at most

$$P\left[\bigcup A_i\right] \leq \sum_i P[A_i] \leq \frac{2c \ln \ln \overline{\Delta}}{n^{k-2}} \leq \frac{2c}{n^{k-3}}.$$

By taking into account that $\overline{\Delta}$ might reach the threshold $6k^5 \ln^5 n$ earlier in the algorithm's execution, the probability that it ends within $c \ln \ln \overline{\Delta}$ iterations only grows. Thus, after $c \ln \ln \overline{\Delta}$ iterations, $\overline{\Delta} \leq 6k^5 \ln^5 n$ and hence the maximum degree of the graph is at most $6k^5 \ln^5 n$ with high probability. □

3.2 Phase 2: A deterministic Algorithm for a $O(\log \log n)$-ruling set

As the maximum degree of the remaining graph becomes smaller, RandStep becomes less effective in reducing it. This is because a node u in $\mathcal{B}$ with small d_u will only be removed from the graph with a small probability (and its neighbors, too). On the other hand, the deterministic algorithm Sparsify from [8] (called Algorithm 1 in their paper) guarantees to halve the maximum degree of a Growth-Bounded Graph in a constant number of steps. Thus, the latter algorithm is slower than ours while Δ is large, but is faster than RandStep once Δ is only polylogarithmic.

Algorithm 2: SparsifyStep

Input: A Growth-Bounded Graph $G = (V, E)$.
Output: The node-set M of a subgraph of G

1 Send an invitation to one arbitrarily chosen neighbor.
2 Accept one invitation, if any was received.
3 $E' := \{(u, v) \in E \mid u \text{ accepts the invitation of } v \text{ or vice versa}\}$
4 Compute a MIS M on the edge-induced subgraph $\overline{G} := G(V, E')$.
5 **return** M

The algorithm Sparsify is roughly described as follows: In each iteration, a subgraph $\bar{G}$ of G is selected such that every node in $\bar{G}$ has degree 1 or 2, and each node of G has a neighbor in $\bar{G}$. Then a MIS of $\bar{G}$ is computed in $O(\log^* n)$ time using the algorithm from [4]. Only nodes in this MIS stay in the graph for the next iteration. This subset is a 2-ruling set of the nodes in G.

It can be seen from the proof of Lemma 5 in [8] that this algorithm halves the degree of the graph after a constant number of iterations. Furthermore, inspection of this proof also shows that adding interleaved iterations of RandStep does not harm the analysis. Hence:

Lemma 9. *For a Growth-Bounded Graph, Algorithm* Sparsify *reduces the maximum degree of the graph from Δ to $\Delta/2$ in $h = O(1)$ iterations. Each iteration needs $O(\log^* n)$ rounds.*

3.3 Combining Phase 1 and Phase 2: The independent ruling-set Algorithm

The key observation for obtaining Algorithm RulingSet which is fast in both of these phases is that the two algorithms can be "interleaved": after executing one call of RandStep, we execute one call of SparsifyStep. This is possible because for both algorithms, one iteration takes a Growth-Bounded Graph as input, and returns as output a subset of its nodes which is a 2-ruling set of the input graph. Hence, after t iterations of the combined algorithm, the remaining set of nodes is a $4t$-ruling set of the original graph. Furthermore, Theorem 8 still holds for the combined algorithm, as the inserted iterations of SparsifyStep never increase the degree of any node. Once the degree of the remaining subgraph is at most $\Delta \leq 6k^5 \ln^5 n$, by Lemma 9 the deterministic steps guarantee that the degree is quickly decreased to zero (i.e., all remaining nodes are independent) within $O(\log(6k^5 \ln^5 n)) = O(\log \log n)$ iterations of the combined algorithm, as also the iterations of the RandStep algorithm never increase any node's degree.

Of course, interleaving one single step of the deterministic algorithm (and not the required h steps to guarantee the bisection of the degree) deteriorates the constant factor of the running time, but allows on the other hand the execution of the algorithm without knowledge of h. Summarizing, the following theorem holds.

Algorithm 3: RULINGSET

Input: A Growth-Bounded Graph $G = (V, E)$
Output: An independent ruling-set I of G

```
1 I := V; i := 0
2 while I_i is not independent do
3     Call RANDSTEP for I → I
4     if I independent then exit while-loop
5     Call SPARSIFYSTEP for I → I
6 end
7 return I
```

Theorem 10. *For any Growth-Bounded Graph G, Algorithm* RULINGSET *computes a $O(\log\log n)$-ruling independent set of G in $O(\log\log n \log^* n)$ rounds (with high probability) of synchronous distributed computation.*

3.4 Phase 3: Obtaining the Maximal Independent Set

At this point the $O(\log\log n)$-ruling independent set computed by Algorithm RULINGSET is *condensed* to yield a Maximal Independent Set. We invoke the condensing algorithm from [8] (Algorithm 2), which extends any t-ruling independent set to a MIS in $O(t \cdot \log^* n)$ rounds in Growth-Bounded Graphs. Thus, we obtain a MIS in $O(\log\log n \log^* n)$ rounds. For the sake of completeness we describe shortly how this algorithm works. It is, in turn, split into two phases: the first phase condenses the existing ruling set until the resulting set is 3-ruling. This is achieved by letting every active node (i.e., a node in current independent set) compute an independent set in its 4-neighborhood under the constraint that every node in the 3-neighborhood is dominated by an active node. The resulting new set might be no longer independent, but the growth-bounded property of the graph permits to efficiently compute an independent set out of this denser set of active nodes. This phase is repeated until the final set is 3-ruling and takes $O(\log\log n \log^* n)$ rounds.

The final phase, i.e. the task of producing the desired MIS from the 3-ruling set, is achieved by defining a cluster graph that enables the local completion of the independent set achieved so far. A coloring of the cluster graph defines an order of precedence on the active nodes that, in turn, ensures that the final set is independent. Again, the efficiency of the algorithm relies on the bounded growth property of the original graph, which leads to the final MIS in O(1) rounds.

Combined, Theorem 2 and Lemma 8 from [8] imply the following:

Theorem 11. *For a Growth-Bounded Graph G, Phase 3 transforms a $O(\log\log n)$-ruling independent set of G into a MIS of G in $O(\log\log n \log^* n)$ rounds of synchronous computation.*

References

[1] Khaled M. Alzoubi, Peng-Jun Wan, and Ophir Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 157–164, New York, NY, USA, 2002. ACM Press.

[2] Baruch Awerbuch, Andrew V. Goldberg, Michael Luby, and Serge A. Plotkin. Network decomposition and locality in distributed computation. In *IEEE Symposium on Foundations of Computer Science*, pages 364–369, 1989.

[3] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, December 1952.

[4] Richard Cole and Uzi Vishkin. Deterministic coin tossing with applications to optimal parallel list ranking. *Information and Control*, 70(1):32–53, 1986.

[5] Andrzej Czygrinow and Michał Hańćkowiak. Distributed approximation algorithms in unit-disk graphs. In *Proceedings of 20th International Symposium on Distributed Computing (DISC), Stockholm, Sweden, September 18-20, 2006*, volume 4167 of *Lecture Notes in Computer Science*, pages 385–398. Springer, 2006.

[6] Ece Gelal, Gentian Jakllari, Srikanth V. Krishnamurthy, and Neal E. Young. Topology control to simultaneously achieve near-optimal node degree and low path stretch in ad hoc networks. In *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON06)*, volume 2, pages 431–439, 2006.

[7] Beat Gfeller and Elias Vicari. A faster distributed approximation scheme for the connected dominating set problem for growth-bounded graphs. Technical Report 540, Department of Computer Science, ETH Zurich, 2006.
Available at http://www.inf.ethz.ch/research/disstechreps/techreports.

[8] Fabian Kuhn, Thomas Moscibroda, Tim Nieberg, and Roger Wattenhofer. Fast Deterministic Distributed Maximal Independent Set Computation on Growth-Bounded Graphs. In *19th International Symposium on Distributed Computing (DISC), Cracow, Poland*, September 2005.

[9] Fabian Kuhn, Thomas Moscibroda, Tim Nieberg, and Roger Wattenhofer. Local Approximation Schemes for Ad Hoc and Sensor Networks. In *3rd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Cologne, Germany*, September 2005.

[10] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. What cannot be computed locally! In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 300–309, New York, NY, USA, 2004. ACM Press.

[11] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. On the Locality of Bounded Growth. In *24th ACM Symposium on the Principles of Distributed Computing (PODC), Las Vegas, Nevada, USA*, July 2005.

[12] Nathan Linial. Distributive graph algorithms — global solutions from local data. In *28th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 331–335, 1987.

[13] Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

[14] Michael Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM J. Comput.*, 15(4):1036–1055, 1986.

[15] Madhav V. Marathe, Heinz Breu, Harry B. Hunt III, S. S. Ravi, and Daniel J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25:59–68, 1995.

[16] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, January 2005.

[17] Thomas Moscibroda and Roger Wattenhofer. Maximal independent sets in radio networks. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 148–157, New York, NY, USA, 2005. ACM Press.

[18] Moni Naor. A lower bound on probabilistic algorithms for distributive ring coloring. *SIAM Journal on Discrete Mathematics*, 4(3):409–412, 1991.

[19] Tim Nieberg and Johann L. Hurink. Wireless Communication Graphs. In *Proceedings of DEST International Workshop on Signal Processing for Sensor Networks, ISSNIP'04, Melbourne, Australia, 367–372*, 2004.

[20] Alessandro Panconesi and Aravind Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, pages 581–592, New York, NY, USA, 1992. ACM Press.

[21] Srinivasan Parthasarathy and Rajiv Gandhi. Distributed algorithms for coloring and domination in wireless ad hoc networks. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 447–459, 2004.

[22] Sriram V. Pemmaraju and Imran A. Pirwani. Energy conservation via domatic partitions. In *MobiHoc '06: Proceedings of the seventh ACM international symposium on Mobile ad hoc networking and computing*, pages 143–154, New York, NY, USA, 2006. ACM Press.

[23] Stefan Schmid and Roger Wattenhofer. Algorithmic Models for Sensor Networks. In *14th International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS), Island of Rhodes, Greece*, April 2006.