Fast Deterministic Distributed Maximal Independent Set Computation on Growth-Bounded Graphs

Fabian Kuhn[1], Thomas Moscibroda[1], Tim Nieberg[2,⋆], and Roger Wattenhofer[1]

[1] Computer Engineering and Networks Laboratory,
ETH Zurich,
8092 Zurich, Switzerland
{kuhn, moscitho, wattenhofer}@tik.ee.ethz.ch
[2] Department of Applied Mathematics,
University of Twente,
Postbus 217, 7500 AE Enschede, The Netherlands
t.nieberg@utwente.nl

Abstract. The distributed complexity of computing a maximal independent set in a graph is of both practical and theoretical importance. While there exists an elegant $O(\log n)$ time randomized algorithm for general graphs [20], no deterministic polylogarithmic algorithm is known. In this paper, we study the problem in graphs with bounded growth, an important family of graphs which includes the well-known unit disk graph and many variants thereof. Particularly, we propose a deterministic algorithm that computes a maximal independent set in time $O(\log \Delta \cdot \log^* n)$ in graphs with bounded growth, where n and Δ denote the number of nodes and the maximal degree in G, respectively.

1 Introduction

The distributed complexity of computing a maximal independent set (MIS) in a graph has been one of the tantalizing problems in distributed computing. While there are well-known and elegant randomized algorithms that compute a MIS in expected time $O(\log n)$ [20], the open problem formulated by Linial [19], whether there exists a *deterministic* distributed algorithm that finds a MIS in a graph in *polylogarithmic* time, is open.

If every node has a unique identifier, there is a trivial distributed MIS algorithm which works as follows. Every node joins the MIS if it has the smallest ID among its neighbors and if none of its neighbors has already joined the MIS. Unfortunately, this algorithm can result in an entirely sequential execution and linear running time because there may be only a single point of activity at any time. While there exist algorithms that greatly outperform the trivial sequential

⋆ This work is partially supported by the European research project Embedded WiSeNts (FP6-004400) and by the Dutch project Smart Surroundings (BSIK-03060).

P. Fraigniaud (Ed.): DISC 2005, LNCS 3724, pp. 273–287, 2005.

algorithm, e.g., [22], the quest for a polylogarithmic algorithm has so far been in vain.

The particular interest in the MIS problem stems on the one hand from its practical importance in various application settings. Specifically, in a network graph consisting of nodes representing processors, a MIS defines a set of processors which can operate in parallel without interference. On the other hand, the maximal independent set problem is also of outstanding theoretical interest, because it prototypically captures the notion of *symmetry breaking*, one of the central aspects in distributed computing, in a simple, well-defined way. While it is easily conceivable that the symmetry between nodes can be broken using randomization, doing so deterministically appears to be intrinsically difficult.

Recently, maximal independent sets have been granted particular attention by the wireless networking community. In wireless ad hoc and sensor networks, clustering is one of the foremost network organization techniques to enable efficient routing and to cope with failures and mobility. The clustering induced by a MIS has been shown to exhibit particularly desirable properties [2]. Yet, in spite of the multiplicity of papers written in the wireless networking literature, not much is known about the fundamental principles governing the computation of maximal independent sets in the kind of network graphs that have typically been used to model wireless networks.

In wireless networking, the usually studied graphs are unit disk graphs (UDG) in which all nodes are assumed to be located in the Euclidean plane and there exists a communication edge between two nodes u and v iff the Euclidean distance $d(u, v)$ is at most 1. In this paper, we study the distributed complexity of computing a MIS in unit disk graphs. Specifically, we show that for unit disk graphs, Linial's question can be answered in the affirmative. In particular, we present a novel deterministic distributed algorithm that computes a MIS in unit disk graphs in time $O(\log \Delta \cdot \log^* n)$, where Δ denotes the maximal degree in the network graph.

Obtaining a MIS deterministically is easy if each node knows it exact location, and the location of its neighbors. Moreover, if nodes can sense the distance to their neighbors, a MIS can be computed deterministically in time $O(\log^* n)$ as shown in [16]. Hence, it is an important aspect about our result that the nodes do not require *any position or distance* information. That is, the only information available at a node is the *connectivity information* to its neighbors. In graph theoretical terms, this corresponds to a problem definition in which the geometric representation of the graph is not given. Notice that even in the case of centralized algorithms, the absence of a geometric representation of a given unit disk graph renders many problems much harder. Particularly, given a unit-disk graph, it is NP-hard to find its graphical representation or even an approximation thereof [6,14].

The absence of position or distance information implies that our algorithm works for arbitrary graphs, even though it achieves the claimed $O(\log \Delta \cdot \log^* n)$ running time only in graphs with bounded growth, such as the unit disk graph. However, we establish our result not only for unit disk graphs, but for the more

general notion of *growth-bounded graphs*. This more general family of graphs captures the intuitive notion that if many nodes are located close from each other, many of them must be within mutual transmission range. In graph theoretical terms, a graph is growth-bounded if the number of independent nodes in a node's r-neighborhood is bounded.

As we show in this paper, the notion of growth-bounded graphs is closely related to *unit ball graphs* (UBG) introduced in [16]. As opposed to the two-dimensional Euclidean metric in UDGs, we assume the points to be located in an arbitrary metric space. Two nodes can communicate with one another if their mutual distance is at most 1. Our deterministic algorithm works for any metric space, but its running time depends on the *doubling dimension* of the underlying metric. A metric's doubling dimension is the smallest ρ such that every ball can be covered by at most 2^ρ balls of half the radius. Specifically, our algorithm terminates in time $O(\log \Delta \cdot \log^* n)$ if the underlying metric is *doubling*, that is, its doubling dimension ρ is constant. In this case, the resulting unit ball graph is polynomially growth-bounded.

Clearly, our claim for unit disk graphs then follows directly as a special case of this more general result, because a UDG is growth-bounded and a UBG with constant doubling dimension. Intriguingly, this result shows that in graphs with bounded growth, our deterministic algorithm outperforms Luby's randomized algorithm which requires $O(\log n)$ expected time.

Besides being of theoretical interest of its own, the reason for studying more general growth-bounded graphs instead of merely unit disk graphs are twofold. First, it is well known that unit disk graphs, while generally being respected as a first modelling step, do in many ways not capture the reality experienced in wireless networks closely. Secondly, it has been argued that the distance metric induced by Internet latencies is growth-bounded and incudes a doubling metric.

The remainder of the paper is organized as follows. We give an overview over related work in Section 2. Section 3 formally introduces our model of computation and necessary notation. The algorithm is then presented and analyzed in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

The distributed (randomized and deterministic) complexity of computing a MIS has been of fundamental interest to the distributed computing community for various reasons [20,7,4,18]. A major breakthrough in the understanding of the distributed computational complexity of MIS was the elegant randomized algorithm by Luby [20] that has a running time of $O(\log n)$ (see also [1,11]). The distributed computation of maximal independent sets have also been studied in the context of backbone construction in wireless networks [2,8] and in radio network models [21]. However, all these algorithms either have linear running time [2] or are probabilistic [8,21].

The fastest known *deterministic* distributed algorithms are based on the notion of network decompositions introduced in [4]. A $(d(n), c(n))$-*network de-*

composition of a graph $G = (V, E)$ is a partition of V into disjoint *clusters*, such that the subgraph induced by each cluster is connected, the diameter of each cluster is in $d(n)$, and the chromatic number of the resulting cluster graph is in $c(n)$, where the cluster graph is obtained by contracting each cluster into a single node. Given a $(d(n), c(n))$-decomposition a MIS can easily be computed in time $d(n) \cdot c(n)$. First all clusters of the first color compute a MIS in parallel in time $d(n)$. Subsequently, clusters of the next colors iteratively add their contributions to the MIS. In [4] authors present a deterministic $O(f(n)^c)$ time algorithm for computing an $(f(n)^c, f(n)^c)$-decomposition, where c is a constant, and $f(n) = n^{\sqrt{\log \log n / \log n}}$, yielding a deterministic $O(f(n)^c)$ MIS algorithm. The fastest currently known deterministic MIS algorithm was given in [22]. Based on a $(g(n)^d, g(n))$-decomposition in time $O(g(n)^d)$, where $g(n) = n^{\sqrt{1/\log n}}$ and d is a constant, the authors obtain a deterministic $O(g(n)^d)$ MIS algorithm.

On the other hand, the first lower bound given for the distributed computation of maximal independent sets has been given by Linial in [18]. This lower bound says that even on a ring topology, at least time $\Omega(\log^* n)$ is required to compute a maximal independent set. Subsequently, it was shown in [15] that in general graphs, every (possibly randomized) algorithm requires at least $\Omega(\sqrt{\log n / \log \log n})$ or $\Omega(\log \Delta / \log \log \Delta)$ communication rounds for computing a MIS.

While all of the above results hold for general graphs, much less is known about the complexity in unit disk graphs or growth-bounded graphs. In fact, if distances are unknown, the fastest deterministic algorithm to compute a MIS in unit disk graphs is the algorithm given in [22] for general graphs.

Bounded growth metrics in general and doubling metrics in particular have found quite a lot of attention recently [10,12,13,24,25]. It is proposed that latencies of many real networks such as peer-to-peer networks or the Internet are doubling. The network-related problems which are solved include metric embeddings [10,12], distance labeling and compact routing [25], and nearest neighbor search [13]. The doubling dimension has been introduced in [10], however, a similar notion has already been used in [3].

The notion of a unit ball graph (UBG) has been introduced in [16] in which the authors prove that if the underlying metric space exhibits constant doubling dimension, it is possible to construct a $(O(1), O(1))$-network decomposition in $O(\log^* n)$ communication rounds, which is asymptotically optimal. As shown above, this implies a $O(\log^* n)$ time algorithm for computing a MIS in UBGs. However, in contrast to our paper, [16] assumes that nodes can sense the *distances* to their neighboring nodes. That is, nodes know about the underlying metric space, whereas in this paper, nodes must base their decision merely on the connectivity information induced by the UBG.

3 Model and Definitions

For our algorithms, we use the standard message passing model. The network is modelled as an undirected graph $G = (V, E)$. Two nodes $u, v \in V$ of the

network are connected by an edge $(u, v) \in E$ whenever there is a direct bidirectional communication channel connecting u and v. For simplicity, we assume a synchronous communication model where time is divided in rounds. In each round, every node can send a message of size $O(\log n)$ to each of its neighbors in G. The time complexity of an algorithm is the number of rounds it needs to complete. Note that at the cost of higher message complexity, every synchronous message passing algorithm can turned into an asynchronous algorithm with the same time complexity.

As discussed, finding a fast deterministic algorithm for computing a MIS on a general network graph is a hard problem. We therefore restrict our attention to an important class of graphs which we call *growth-bounded* graphs and which are defined as follows.

Definition 1. ***(Growth-Bounded Graph)*** *We call a graph G f-*growth-bounded *if there is a function $f(r)$ such that every r-neighborhood in G contains at most $f(r)$ independent nodes.*

Note that $f(r)$ does not depend on the number of nodes n or any other property of G. Hence, for constant r, the number of independent nodes in an r-neighborhood is constant.

The class of growth-bounded graphs seems to cover the class of graphs which can occur in wireless networks quite well. In wireless networks, nodes have some geographic position. Nearby nodes tend to hear each other, far-away nodes cannot communicate because with the available power, radio signals can only be transmitted up to some distance. In particular the class of growth-bounded graphs covers the widely used unit disk graphs as well as UDG variants (e.g. [5,17]). In [16], the unit disk graph model has been extended to general metric spaces resulting in so-called unit ball graphs (UBG). The nodes of a UBG are the points of a metric space; two nodes are connected if and only if their distance is at most 1. The following lemma shows that if the underlying metric space of a UBG G has constant doubling dimension, G is polynomially growth-bounded

Lemma 1. *Let G be a unit ball graph and let ρ be the doubling dimension of the underlying metric space. Every r-neighborhood of G contains at most $(2r)^\rho$ independent nodes.*

Proof. By the definition of a UBG, the r-neighborhood of node v in G is completely covered by the ball $B_r(v)$ with radius r around v. By the definition of the doubling dimenstion ρ, $B_r(v)$ can be covered by at most $2^{\rho(1+\log_2 r)}$ balls of radius $1/2$. By the triangle inequality, two nodes inside a ball of radius $1/2$ have distance at most 1, that is, the nodes inside a ball of radius $1/2$ form a clique in G. The number of independent nodes in the r-neighborhood of v, therefore is at most $2^{\rho(1+\log_2 r)} = 2^\rho r^\rho$.

For the description and analysis of our MIS algorithm, we need a formal notion for the density of an independent set. To do so, we use the following definition from [23]. Let $S \subseteq V$ be a subset of the nodes of a graph $G = (V, E)$.

S is called *r-ruling* if for each node $u \in V \setminus S$, the distance to the closest node in S is at most r.

If the set S in the above definition is an independent set, we speak of an r-ruling independent set. Note that a MIS is a 1-ruling independent set.

Throughout the paper, we will make use of node neighborhoods of particular radii. By $\Gamma_r(v)$ we denote the set of nodes $u \neq v$ which have distance at most r from v. We further define

$$\Gamma_r^+(v) := \Gamma_r(v) \cup \{v\} \text{ and } \Gamma(v) := \Gamma_1(v).$$

4 Distibuted MIS Construction

In this section, we describe and analyze our distributed deterministic MIS construction for growth-bounded graphs. The algorithm consists of three phases. In the first phase, in time $O(\log \Delta \cdot \log^* n)$, an $O(\log \Delta)$-ruling independent set S of the network graph G is computed. The second phase transforms the sparse set S in to a dense independent set S' such that each node v of G has a node $u \in S'$ at distance at most 3, that is, S' is 3-ruling. For growth-bounded graphs, such a dense independent set induces an $(O(1), O(1))$-decomposition which can be used to finally extend S' to a maximal independent set in the third phase.

4.1 Constructing a Sparse Independent Set

The first phase of our MIS construction is a distributed algorithm which locally computes an $O(\log \Delta)$-ruling independent set S for a given undirected growth-bounded graph $G = (V, E)$ in time $O(\log \Delta \log^* n)$. A detailed description of the first phase is given by Algorithm 1. Before analyzing the algorithm, we give an informal description of the code.

At the beginning, S is empty and all nodes are active (denoted by the variables $b(v)$ for $v \in V$). Nodes are active as long as they have not decided whether to join the independent set S. As soon as a node becomes passive, it has either joined S in line 20 or it has decided not to join S. From a general perspective, Algorithm 1 tries to eliminate active vertices from the network until single, locally independent nodes are left. It does so with the help of edge-induced subgraphs of bounded degree. In each iteration of the while loop, a constant-degree graph $\overline{G}$ consisting of active nodes and edges of G is computed. On $\overline{G}$, we can construct a MIS in time $O(\log^* n)$ [7,9,18]. Only the nodes of the MIS of $\overline{G}$ stay active after the iteration of the while loop. This way, the number of active nodes is reduced by at least a constant factor in every while loop iteration. As soon as an active node v has no active neighbors, v joins the independent set S (line 20). The graph $\overline{G}$ is constructed as follows. First, each active node v chooses an active neighbor $d(v)$. Then, each active node u which has been chosen by at least one neighbor v, selectes a neighbor $p(u)$ for which $d(p(u)) = u$. The edge set of $\overline{V}$ consists of all edges of the form $(u, p(u))$. Because a node u can only be connected to $d(u)$ and $p(u)$, $\overline{G}$ has at most degree 2. Now, consider a single execution of the while loop (lines 3–22, Figure 1).

Algorithm 1 Computing an IS (code for vertex v)

```
 1: S := ∅;
 2: b(v) := act;
 3: while b(v) = act do
 4:   if ∃u ∈ Γ(v) | b(u) = act then
 5:     d(v) := min{u ∈ Γ(v) | b(u) = act};
 6:     inform neighbor d(v);
 7:     A_v := {u ∈ Γ(v) | d(u) = v};
 8:     if A_v ≠ ∅ then
 9:       p(v) := min A_v;
10:       inform neighbor p(v)
11:     fi;
12:     B_v := {u ∈ Γ(v) | p(u) = v};
13:     if (A_v = ∅) ∧ (B_v = ∅) then
14:       b(v) := pass
15:     else
16:       construct MIS I on graph Ḡ = (V̄, Ē) with V̄ := {u ∈ V | b(u) = act} and
          Ē := {(u, p(u)) | u ∈ V̄ ∧ A_u ≠ ∅};
17:       if v ∉ I then b(v) := pass fi
18:     fi
19:   else
20:     S := S ∪ {v}; b(v) := pass
21:   fi
22: od
```

Lemma 2. *In the graph $\overline{G} = (\overline{V}, \overline{E})$, every vertex has degree at most* 2.

Proof. Consider $v \in \overline{V}$, then there are at most two vertices adjacent to v by an edge in $\overline{E}$, namely $d(v)$ if $p(d(v)) = v$, and $p(v)$.

Note that due to this lemma, line 16 of the algorithm, that is, the local construction of a MIS I on $\overline{G}$, can be completed in $O(\log^* n)$ rounds using methodes described in [7,9,18].

Lemma 3. *Let V_A denote the set of active nodes. After k iterations of the while loop, $S \cup V_A$ is a $2k$-ruling set of G.*

Proof. We prove the lemma by induction over the number k of while loop iterations. Initially all nodes are active, thus the lemma is satisfied for $k = 0$. For the induction step, we show that if a node v becomes passive in an iteration of the while loop, either v joins S or there is an active node at distance at most 2 from v which remains active for until the next while loop iteration. Node v can become passive in lines 14, 17, or 20. If v becomes passive in line 20, it joins S and therefore the condition of the lemma is satisfied. In line 17, v is a node of $\overline{G}$ and has a neighbor u of v which is in the MIS I of $\overline{G}$. Thus, node u remains active.

The last remaining case is that v decides to become passive in line 14. By the condition in line 4, we can assume that v has at least one active neighbor at the

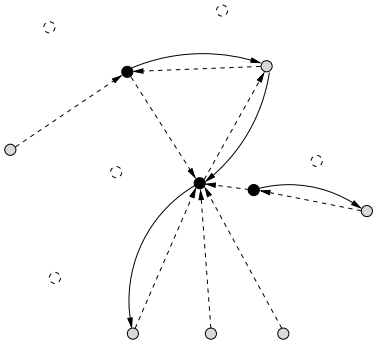

Fig. 1. One iteration of Algorithm 1. The dashed nodes are passive at the outset of the iteration. The dashed arrows between active nodes denote the links $d(v)$. The graph $\overline{G}$ is induced by the links $p(v)$ which are denoted by the solid, bended arrows. Finally, the algorithm computes a MIS on $\overline{G}$, leaving only the black nodes active for the next iteration.

beginning of the while loop iteration. Therefore, v can choose a node $u = d(v)$ in line 5. Since $A_u \neq \emptyset$, u chooses a node $p(u)$ and therefore u is a node of $\overline{G}$. Because all nodes of the MIS I of $\overline{G}$ remain active, either u or a neighbor w of u is still active after completing the while loop iteration. Since $\text{dist}_G(v, w) = 2$, this completes the proof.

The following two lemmas are used to give bounds on the number of rounds needed by Algorithm 1 to complete, and to explain the resulting structure in G for general graphs and for growth-bounded graphs, respectively.

Lemma 4. *After $O(\log n)$ consecutive executions of the while loop, Algorithm 1 terminates with a $O(\log n)$-ruling independent set S on any graph G.*

Proof. Let n_{act} be the number of active nodes at the beginning of an iteration of the while loop. We prove that in one while loop iteration, at least $n_{\text{act}}/3$ nodes become passive. The claim then follows by Lemma 3.

Let $\overline{n} \leq n_{\text{act}}$ be the number of nodes of $\overline{G}$ of some particular iteration of the while loop. All nodes which are not part of $\overline{G}$ become passive in lines 14 or 20. It therefore suffices to prove that at least one third of the nodes of $\overline{G}$ become passive. However, $\overline{G}$ is constructed such that it does not contain isolated nodes, that is, all nodes of $\overline{G}$ have at least degree 1. Note that $\overline{G}$ is an edge-induced subgraph of G. Because the maximum degree of a node in $\overline{G}$ is 2 (Lemma 2), the MIS I consists of at most $2\overline{n}/3$ nodes. Hence, at least $\overline{n}/3$ nodes become passive in line 17.

The following lemma shows that for growth bounded graphs, the run time can even be reduced to $O(\log \Delta)$ executions of the while loop where Δ denotes the maximum degree of the network graph G.

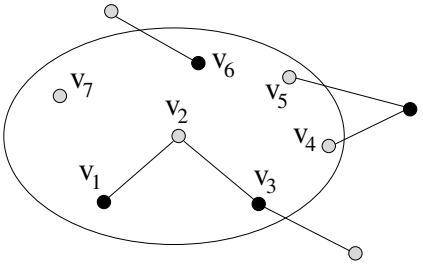

Fig. 2. The cluster with the edges in $\overline{G}$. Black nodes will remain active in the next iteration. The nodes v_1, v_2, and v_3 are in C_i. Nodes v_4, v_5, and v_6 are connected only to nodes outside of the cluster and hence, are in set C_o. Finally, $v_6 \in C_p$.

Lemma 5. *If the network graph G is growth-bounded, after $O(\log \Delta)$ consecutive execution of the while loop, Algorithm 1 terminates with an $O(\log \Delta)$-ruling independent set S.*

Proof. Let M be a maximal independent set of G. The set M defines a clustering of G as follows. We associate a cluster $\mathcal{C}_u$ with each node $u \in M$. Each node $v \notin M$ is assigned to the cluster of an adjacent node $u \in M$. Note that each cluster contains at most $\Delta + 1$ nodes. Let us define the cluster graph $\mathcal{G}_\mathcal{C}$ as follows. The nodes of $\mathcal{G}_\mathcal{C}$ are the clusters $\mathcal{C}_u$. Two nodes $\mathcal{C}_u$ and $\mathcal{C}_v$ are connected if there is an edge connecting the respective clusters. Because we assume that G is growth bounded, there is a function f such that there are at most $f(3) = O(1)$ independent nodes at distance at most 3 from a node u. Therefore, the maximum degree of $\mathcal{G}_\mathcal{C}$ is bounded by $d := f(3)$.

In the following, we show that the maximum number of active nodes per cluster is reduced by a factor 2 in a constant number of while loop iterations. For convenience, we define a unit of time to be one iteration of the while loop. Let α be the maximum number of active nodes per cluster at some time t. We will show that there is a constant k such that at time $t + k$ each cluster contains at most $\alpha/2$ active nodes. Note that this implies the lemma because we have $\alpha \leq \Delta + 1$ at time $t = 0$. Let $\mathcal{C}_u$ be a cluster with $c > \alpha/2$ active nodes. Let us look at a single iteration of the while loop of Algorithm 1. We partition the c active nodes of $\mathcal{C}_u$ into three groups according to their neighbors in $\overline{G}$ (Figure 2). We denote the set of nodes v which become passive in line 6 because there is no node w for which $d(w) = u$ by C_p. The set of nodes which have a neighbor inside $\mathcal{C}_u$ and which are only connected to nodes outside $\mathcal{C}_u$ are called C_i and C_o, respectively. Clearly, we have $|C_p| + |C_i| + |C_o| = c$. Because the maximum degree of $\overline{G}$ is 2, during the construction of the MIS in line 10 at least one third of the nodes in C_i become passive. The nodes in C_o can be divided into the nodes C_o^p which become passive and the nodes C_o^a which stay active. Each node outside $\mathcal{C}_u$ is connected to at most 2 nodes in C_o^a. Therefore, at least $|C_o^a|/2$ nodes outside $\mathcal{C}_u$ become passive. Let $c_i := |C_p| + |C_i| + |C_o^p|$ and $c_o := |C_o^a|$. We have $c_i + c_o = c$. In each iteration of the while loop at least $c_i/3$ nodes in

Algorithm 2 Computes a dense IS

Input: t-ruling independent set S
Output: 3-ruling independent set S
1: $S' := S$;
2: **while** S' is not 3-ruling **do**
3: **for each** $u \in S'$ **do**
4: compute $\hat{S}_u \subset \Gamma_4^+(u)$ such that $S' \cup \hat{S}_u$ is an IS and $\forall v \in \Gamma_3^+(u), \exists w \in S' \cup \hat{S}_u$: $\{v, w\} \in E$;
5: $\mathcal{G}$ is the graph induced by $\bigcup_{u \in S'} \hat{S}_u$;
6: $S' := S' \cup \mathrm{MIS}(\mathcal{G})$;
7: **od**;
8: **od**

$\mathcal{C}_u$ and at least $c_o/2$ nodes of clusters which are adjacent to $\mathcal{C}_u$ become passive. Assume that after k iterations of the while loop, there are still $\alpha/2$ active nodes in $\mathcal{C}_u$. Let $c^{(j)}$, $c_i^{(j)}$, and $c_o^{(j)}$ be the values of c, c_i, and c_o of the j^{th} iteration, respectively. Because there are at most α nodes at the beginning, we have

$$\frac{1}{3} \cdot \sum_{j=1}^{k} c_i^{(j)} \leq \frac{\alpha}{2} \tag{1}$$

because otherwise at least $\alpha/2$ nodes of $\mathcal{C}_u$ would have become passive. Therefore, the number of nodes in the neighbor clusters of $\mathcal{C}_u$ which have become passive is at least

$$\frac{1}{2} \cdot \sum_{j=1}^{k} c_o^{(j)} = \frac{1}{2} \cdot \sum_{j=1}^{k} c^{(j)} - c_i^{(j)} \geq \frac{k\alpha}{4} - \frac{1}{2} \cdot \sum_{j=1}^{k} c_i^{(j)}.$$

Because of Equation (1), this is at least $(k-3)\alpha/4$. Because there are at most $d\alpha$ active nodes in neighbor clusters of $\mathcal{C}_u$ at the beginning, after $O(d) = O(1)$ iterations of the while loop there are no active nodes in the neighborhood of $\mathcal{C}_u$ left. From then on, at least one third of the nodes in $\mathcal{C}_u$ becomes passive in every further iteration.

Summarizing the Lemmas 2–5, we obtain the following theorem.

Theorem 1. *Algorithm 1 is a local, distributed algorithm which computes an* $\mathrm{O}(\log \Delta)$*-ruling independent set in* $O(\log \Delta \cdot \log^* n)$ *rounds for any growth-bounded graph* $G = (V, E)$.

For general graphs, the Algorithm terminates in $O(\log n \cdot \log^* n)$ *rounds producing an* $O(\log n)$*-ruling independent set. All messages are of size* $O(\log n)$.

4.2 Making the Independent Set Dense

In the following, we show how the relatively sparse independent set which we constructed so far can be made dense enough to obtain an $(O(1), O(1))$-decomposition for growth-bounded graphs. Specifically, we show how on growth-bounded graphs,

a t-ruling independent set can be transformed into a 3-ruling independent set in $O(t \log^* n)$ rounds using messages of size $O(\log n)$. Algorithm 2 describes the basic method to achieve this. The idea is to enlarge the independent set in small steps such that it gets denser in each step. Before coming to a detailed analysis, we give a rough overview. In line 3, each node of the independent set adds new nodes to the independent set such that each neighbor in distance at most 3 has a neighbor in the extended set. Because every independent set node adds new nodes, it is not guaranteed that the additional nodes generated by different independent set nodes are independent. Therefore, in lines 4 and 5, the independence of the extended independent set is restored by computing a MIS on the new nodes (see Lemma 7). The following lemma shows that in each iteration of the while loop, the maximum distance of any node to the next node of S' decreases by at least 1.

Lemma 6. *Let S' be a t-ruling independent set for $t > 3$. After one iteration of the while loop of Algorithm 2, S' is a $(t-1)$-ruling independent set.*

Proof. We first prove that S' remains an independent set throughout the algorithm. The sets $\hat{S}_u$ are constructed such that nodes in S' and nodes in $\hat{S}_u$ are independent. We therefore only have to prove that all the new nodes form an independent set. However, this is clearly guaranteed because in line 6, a maximal independent set of the graph induced by all the new nodes is computed.

To prove that the maximum distance from a node to the next independent set node decreases, we consider a node $v \in V$ for which the distance to the nearest node $u \in S'$ is $t > 3$. We prove that after an iteration of the while loop, the distance between v and the closest node in S' is at most $t-1$. The set $\hat{S}_u$ is constructed such that every node w in the 3-neighborhood $\Gamma_3^+(u)$ has a neighbor in $S' \cup \hat{S}_u$. On a shortest path (of length t) connecting u and v, let x be the node which is at distance exactly 3 from u. There must be a neighbor y of x for which $y \in \hat{S}_u$. After computing the MIS in line 6, either y or a neighbor z of y join the independent set S'. The distance between v and y is $t-1$ and the distance between v and z is $t-1$ which concludes the proof.

It remains to show that Algorithm 2 can indeed be implemented by an efficient distributed algorithm. Lemma 7 gives exact bounds on the distributed complexity of the Algorithm 2.

Lemma 7. *Let G be a growth bounded graph. On G, Algorithm 2 can be executed by a distributed algorithm with time complexity $O(t \log^* n)$ using messages of size $O(\log n)$.*

Proof. By Lemma 6, Algorithm 2 terminates after at most t iterations of the while loop. We therefore have to prove that each while loop iteration can be exectuted in time $O(\log^* n)$ using messages of size $O(\log n)$. Let us first look at the construction of $\hat{S}_u$ for some node $u \in S'$. A node $v \in \Gamma_4^+(u)$ can potentially join $\hat{S}_u$ if it has neighbor in $S' \cup \hat{S}_u$ and if it has an uncovered neighbor $w \in \Gamma_3^+(u)$, that is, w has no neighbor in $S' \cup \hat{S}_u$. We call such a node v candidate. We add a candidate v to $\hat{S}_u$ if it has a lower ID than all adjacent candidates. Finding

out whether a node is a candidate and whether it has the lowest ID among its neighbor candidates can be done in 3 rounds. First, all nodes of $S' \cup \hat{S}_u$ inform their neighbors that they are in the independent set. Then, all covered nodes in $\Gamma_3^+(u)$ inform their neighbors which can now decide whether they are candidates. Finally, the candidates exchange their IDs. We call those 3 rounds a step. In each step, at least the candidate with the highest ID joins $\hat{S}_u$. Because we assume that G is a growth bounded graph, there can be at most $f(4) = O(1)$ independent nodes in $\Gamma_u^+(u)$ for some function f. Hence, the number of nodes in $\hat{S}_u$ and therefore the number of steps needed to construct $\hat{S}_u$ is constant. Note that if there was no restriction on the message size, u could collect the complete 4-neighborhood, locally compute $\hat{S}_u$, and inform the nodes in $\hat{S}_u$ in 8 rounds.

It now remains to prove that the construction of the MIS in line 6 of Algorithm 2 can be computed in $O(\log^* n)$ rounds. Let us therefore have a look at the structure of the graph $\mathcal{G}$ which is induced by the union of the sets $\hat{S}_u$ for all $u \in S'$. Consider a node v of $\mathcal{G}$, that is, $v \in \hat{S}_u$ for some $u \in S'$. Further, let w be a neighbor of v in $\mathcal{G}$. The node w is in $\hat{S}_{u'}$ for some node $u' \in S' \setminus \{u\}$. Because $\hat{S}_{u'}$ consists of nodes of $\Gamma_4^+(u')$, the distance between v and u' is at most 5. Since G is a growth bounded graph, there exists a function f such that there are at most $f(5)$ independent nodes at distance at most 5 from v. Thus, there are at most $f(5)$ possible nodes $u' \in S'$ which can cause neighbors w for v. Because all nodes in $\hat{S}_{u'}$ are independent, the number of neighbors of w in $\hat{S}_{u'}$ is at most $f(1)$. Therefore, the maximum degree of the graph $\mathcal{G}$ can be upper bounded by $f(5) \cdot f(1) = O(1)$. It is well-known that on a constant-degree graph, a MIS can be constructed in $O(\log^* n)$ rounds using messages of size $O(\log n)$ [7,9,18].

Combining Lemmas 6 and 7 we obtain the next theorem.

Theorem 2. *On a growth-bounded graph, a t-ruling independent set can be transformed into a 3-ruling independent set in $O(t \log^* n)$ rounds using messages of size $O(\log n)$.*

4.3 Computing the MIS

We will now describe the last phase of our algorithm, turning the 3-ruling independent set S' from Algorithm 2 into a MIS. Set S' induces a natural clustering of the nodes of G. For each node $u \in S'$, we define the cluster $\mathcal{C}_u$ to be the set of all nodes $v \in V$ for which u is the nearest node of S', ties are broken arbitrarily. The cluster graph $\mathcal{G}_{S'}$ induced by S' is then defined as follows. The node set of $\mathcal{G}_{S'}$ is the set of clusters $\{\mathcal{C}_u | u \in S'\}$. The clusters $\mathcal{C}_u$ and $\mathcal{C}_v$ are connected by an edge in $\mathcal{G}_{S'}$ if and only if there are nodes $u' \in \mathcal{C}_u$ and $v' \in \mathcal{C}_v$ which are neighbors in the network graph G. Because S' is a 3-ruling set, the distance between the centers u and v of two neighboring clusters $\mathcal{C}_u$ and $\mathcal{C}_v$ can be at most 7. The degree of $\mathcal{G}_{S'}$ is therefore bounded by $f(7) = O(1)$ if G is f-growth-bounded. The first step of the third phase of our MIS algorithm is to compute $\mathcal{G}_{S'}$ and to color $\mathcal{G}_{S'}$ with $f(7) + 1$ colors, resulting in a $(O(1), O(1))$-decomposition of G. Applying algorithms from [7,9,18], this can be achieved in $O(\log^* n)$ rounds using messages of size $O(\log n)$.

Having computed this decomposition, we can now compute a MIS M of G by sequentially computing the contributions from each color of the coloring of $\mathcal{G}_{S'}$. For each node v, let x_v be the color of v's cluster. Using the cluster colors and the node identifiers, we define a lexicographic order $\prec$ on the set V such that for $u, v \in V$, $u \prec v$ if and only if $x_u < x_v$ or if $x_u = x_v \wedge \mathrm{ID}(u) < \mathrm{ID}(v)$. Each node now proceeds as follows. Initially, we set $M = S'$. All nodes v of S' inform their neighbors about the joining of M by sending a JOIN(v) message. If a node u receives a JOIN(v) message from a neighbor v, it cannot join the MIS any more and therefore sends a COVERED(u) message to all neighbors. If a node v has not received a JOIN(u) message but has received a COVERED(u) from all $u \in \Gamma(v)$ for which $u \prec v$, it can safely join M. Note that all neigbors $w \in \Gamma(v)$ with $w \succ v$, would need to receive a COVERED(v) message from v before joining M. If a node v joins M, it informs its neighbors by sending a JOIN(v) message. As shown by the next lemma, the described algorithm computes a MIS M in time $O(1)$.

Lemma 8. *On f-growth-bounded graphs the above algorithm computes a MIS M in time $2f(7)f(3)$.*

Proof. We first show that M indeed is an independent set of G. For the sake of contradiction, assume that there are two adjacent nodes u and v which both join M. W.l.o.g., we assume that $u \prec v$. Assuming that v joins M means that v must have received a COVERED(u) message from u. However, this is a contradition to the assumption that u joins M. To see that M is a MIS, observe that as long as M is not maximal, there is a smallest node u (with respect to $\prec$) which is not covered.

It remains to prove the time complexity of the above algorithm. First note that because the radius of each cluster is at most 3, there can be at most $f(3)$ MIS nodes per cluster. Let us now look at a single cluster $\mathcal{C}_u$ of the smallest color 1. Because with respect to the order $\prec$, the nodes of $\mathcal{C}_u$ are smaller than all nodes of neighboring clusters, the smallest uncovered node of $\mathcal{C}_u$ is always free to join M. When a node v joins M, it takes two rounds until the neighbors of v have forwarded the information that they have been covered. Because there are at most $f(3)$ nodes of $\mathcal{C}_u$ which join M, it takes at most $2f(3)$ rounds until all nodes of color 1 are covered or have joined M. As soon as there is no uncovered node of a color i, the above argument holds for color $i+1$. Therefore, after at most $f(7) \cdot 2f(3)$ rounds, all nodes are either covered or have joined M.

Combining Theorems 1, and 2 and Lemma 8, we obtain the main theorem of this paper.

Theorem 3. *Let G be a growth-bounded network graph. There is a deterministic distributed algorithm which constructs a maximal independent set on G in time $O(\log \Delta \cdot \log^* n)$.*

5 Conclusions

In this paper, we have given a deterministic distributed algorithm which computes a maximal independent set in time $O(\log \Delta \cdot \log^* n)$ in unit ball graphs if the underlying metric has constant doubling dimension. This includes the practically important special case of unit disk graphs. Our algorithms does not rely on any representation of the underlying metric space, i.e., nodes do not need to know about the distances to their neighbors.

The MIS problem being of fundamental nature, our result sheds new light into the intriguing question of the possibilities and limitations of different models in distributed computing. It is therefore interesting to compare our solution with MIS algorithms in other models. In the radio network model, where collisions between neighboring senders may occur, the fastest known algorithm for unit disk graph runs in time $O(\log^2 n)$ and requires randomization [21]. In the message passing model, the fastest known algorithm for general graphs is randomized and runs in time $O(\log n)$ [20]. Finally, [16] showed that if nodes know the distances to their neighbors, there exists a deterministic $O(\log^* n)$ time algorithm in graphs with bounded growth. The last comparison particularly highlights the importance of *distance information*, which appears to render the MIS problem much simpler.

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, P.-J. Wan, and O. Frieder. Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks. In *Proceedings of the 3^{rd} ACM Int. Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC)*, pages 157–164, EPFL Lausanne, Switzerland, 2002.
3. P. Assouad. Plongements lipschitziens dans $\mathbf{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.
4. B. Awerbuch, A. V. Goldberg, M. Luby, and S. A. Plotkin. Network decomposition and locality in distributed computation. In *Proc. of the 30^{th} Symp. on Foundations of Computer Science (FOCS)*, pages 364–369, 1989.
5. L. Barrière, P. Fraigniaud, and L. Narayanan. Robust Position-Based Routing in Wireless Ad Hoc Networks with Unstable Transmission Ranges. In *Proc. of the 5^{th} International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M)*, pages 19–27. ACM Press, 2001.
6. H. Breu and D. G. Kirkpatrick. Unit Disk Graph Recognition is NP-hard. *Computational Geometry. Theory and Applications*, 9(1-2):3–24, 1998.
7. R. Cole and U. Vishkin. Deterministic Coin Tossing with Applications to Optimal Parallel List Ranking. *Information and Control*, 70(1):32–53, 1986.
8. R. Gandhi and S. Parthasarathy. Distributed Algorithms for Coloring and Connected Domination in Wireless Ad Hoc Networks. In *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2004.
9. A. Goldberg, S. Plotkin, and G. Shannon. Parallel Symmetry-Breaking in Sparse Graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 1(4):434–446, 1988.

10. A. Gupta, R. Krauthgamer, and J. Lee. Bounded Geometries, Fractals, and Low-Distortion Embeddings. In *Proc. of 44th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2003.
11. A. Israeli and A. Itai. A Fast and Simple Randomized Parallel Algorithm for Maximal Matching. *Information Processing Letters*, 22:77–80, 1986.
12. J. Kleinberg, A. Slivkins, and T. Wexler. Triangulation and Embedding using Small Sets of Beacons. In *Proc. of 45th IEEE Symp. on Foundations of Computer Science (FOCS)*, 2004.
13. R. Krauthgamer and J. Lee. Navigating Nets: Simple Algorithms for Proximity Search. In *Proc. of 15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2004.
14. F. Kuhn, T. Moscibroda, and R. Wattenhofer. Unit Disk Graph Approximation. In *Proceedings of the 2004 Joint Workshop on Foundations of Mobile Computing (DIALM)*, pages 17–23, New York, NY, USA, 2004. ACM Press.
15. F. Kuhn, T. Moscibroda, and R. Wattenhofer. What Cannot Be Computed Locally! In *Proc. of the* 23^{rd} *ACM Symp. on Principles of Distributed Computing (PODC)*, pages 300–309, 2004.
16. F. Kuhn, T. Moscibroda, and R. Wattenhofer. The Locality of Bounded Growth. In *Proc. of the* 24^{th} *ACM Symp. on Principles of Distributed Computing (PODC)*, 2005.
17. F. Kuhn, R. Wattenhofer, and A. Zollinger. Ad-Hoc Networks Beyond Unit Disk Graphs. In *Proceedings of* 1^{st} *Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 69–78. ACM Press, 2003.
18. N. Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, February 1992.
19. N. Linial. Local-Global Phenomena in Graphs. *Combinatorics Probability and Computing*, 2:491–503, 1993.
20. M. Luby. A Simple Parallel Algorithm for the Maximal Independent Set Problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.
21. T. Moscibroda and R. Wattenhofer. Maximal Independent Sets in Radio Networks. In *Proc. of the* 23^{rd} *ACM Symp. on Principles of Distributed Computing (PODC)*, 2005.
22. A. Panconesi and A. Srinivasan. Improved distributed algorithms for coloring and network decomposition problems. In *Proc. of the* 24^{th} *annual ACM symposium on Theory of computing (STOC)*, pages 581–592. ACM Press, 1992.
23. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, 2000.
24. C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA)*, pages 311–320, 1997.
25. K. Talwar. Bypassing the embedding: Approximation schemes and compact representations for low dimensional metrics. In *Proc. of 36th ACM Symp. on Theory of Computing (STOC)*, 2004.