On the Locality of Bounded Growth

Fabian Kuhn
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
kuhn@tik.ee.ethz.ch

Thomas Moscibroda
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
moscitho@tik.ee.ethz.ch

Roger Wattenhofer
Computer Engineering and Networks Laboratory
ETH Zurich
8092 Zurich, Switzerland
wattenhofer@tik.ee.ethz.ch

ABSTRACT

Many large-scale networks such as ad hoc and sensor networks, peer-to-peer networks, or the Internet have the property that the number of independent nodes does not grow arbitrarily when looking at neighborhoods of increasing size. Due to this bounded "volume growth," one could expect that distributed algorithms are able to solve many problems more efficiently than on general graphs. The goal of this paper is to help understanding the distributed complexity of problems on "bounded growth" graphs. We show that on the widely used unit disk graph, covering and packing linear programs can be approximated by constant factors in constant time. For a more general network model which is based on the assumption that nodes are in a metric space of constant doubling dimension, we show that in $\mathrm{O}(\log^* n)$ rounds it is possible to construct a $(\mathrm{O}(1), \mathrm{O}(1))$-network decomposition. This results in asymptotically optimal $\mathrm{O}(\log^* n)$ time algorithms for many important problems.

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

bounded growth, covering, distributed algorithms, dominating sets, doubling dimension, locality, maximal independent set, network decomposition, packing, unit disk graphs

1. INTRODUCTION

The advent of wireless multi-hop networks such as mobile ad hoc and sensor networks has brought about many new algorithmic challenges and paradigms, and it has created a flurry of research activity in this new field. Some of the specific challenges stemming from these network types are mobility and energy constraints. Yet, in spite of these new aspects, most of the algorithmic design principles have remained the same as in classical distributed systems. One important difference between wireless and classical networks is that wireless networks look different. While in classical systems, there often is no restriction on the network topology, in wireless networks, it is usually assumed that the network structure is defined by some geometric graph.

1.1 Model

In the ad hoc and sensor networks community, the most important and most widely used graph model is the *unit disk graph* (UDG). It is assumed that all nodes are in the Euclidean plane. There is an edge between two nodes if and only if their distance is at most 1. The UDG model idealizes a real scenario where the radios of all wireless nodes have equal transmission ranges (normalized to 1) such that two nodes can communicate whenever they are within each others transmission range.

While on the one hand, issues such as mobility or energy make ad hoc networks a theoretically interesting and tremendously challenging subject to study, the fact that UDGs are a lot simpler than general graphs should on the other hand make many problems easier than in the general case. In a non-distributed setting, it is indeed true that for many problems UDGs often allow constant approximations or even polynomial time approximation schemes (PTAS) whereas the same problems cannot be approximated well for general graphs. In a distributed scenario, often, the best known algorithms for UDGs are not faster than the best algorithms for general graphs. On general graphs, for many network coordination tasks such as the construction of small dominating sets, maximal independent sets, or graph colorings, strong upper and lower bounds are known (e.g. [16, 17, 20, 22]). A lot less is known about the distributed complexity of the same problems on UDGs. In fact, to the best of our knowledge, the only non-trivial lower bound that applies to UDGs is Linial's $\Omega(\log^* n)$ for coloring the ring [20]. With the exception of [10] (expected $\mathrm{O}(1)$ approximation for dominating set in $\mathrm{O}(\log\log n)$ rounds), time upper bounds are usually polylogarithmic. However, as the example of the minimum vertex cover problem shows, there are problems which are simpler on UDGs than on general graphs. On the one hand, it can be shown that all non-isolated nodes of a UDG form a 2-approximation for minimum vertex cover. On the other hand, it is shown in [17] that $\Omega(\sqrt{\log n/\log\log n})$ rounds are needed on general graphs.

The goal of this paper is to improve this situation by giving fast algorithms for all of the discussed problems for the UDG and interesting generalizations of the UDG. For our algorithms, we assume that each node can learn the distances to all its neighbors. Our main result (Section 5) is formulated for the following generalization of the UDG model which we call unit ball graph (UBG). Assume that nodes are points in some metric space. Two nodes are connected by an edge if and only if their distance is at most 1. Each node knows the distances to all its direct neighbors.[1] Our result for UBGs depend on the *doubling dimension* of the underlying metric; they are particularly strong if we have a *doubling metric* (constant doubling dimension). We define the *doubling dimension* of a metric as the smallest ρ such that every ball can be covered by at most 2^ρ balls of half the radius. Note that this definition is up to constants equivalent to alternative definitions which have been used in the literature.

Besides that the described extension of UDGs towards general metric spaces makes our results stronger, it is mainly interesting for two reasons. First, although in theory the UDG model is widely used, describing ad hoc and sensor networks as UDGs is usually far from reality. On the other hand, a realistic graph model should comprise the geometric properties of wireless network graphs. While having a distance metric which is doubling certainly keeps many of the good properties real wireless networks have, many of the too idealized assumptions are dropped. The second reason for looking at networks which are based on metric spaces of small doubling dimension is that growth restrictions are a natural assumption in many other networks such as peer-to-peer networks or the Internet. It is for example commonly assumed that the distance metric induced by Internet latencies is doubling. Hence, from that perspective, this paper proves strong upper bounds on the locality of many problems in the Internet.

All our algorithms and results apply to the standard synchronous message passing model where time is divided in rounds. In every round, each node can send a message to all of its neighbors. Although we give upper bounds on message sizes at some places, we generally assume that message size is unbounded and that there is no restriction to local computations. Note that we assume that each node can send a different message to each neighbor while in wireless networks, it is often assumed that a node can only send the same message to all neighbors (local broadcast). As long as there is no assumption on the maximum message size, the two models are equivalent because a node can pack the information for all neighbors into one large message which is then locally broadcasted.

1.2 Results

The paper has two main results. In Section 4 we show that in the described UDG model, an arbitrary covering or packing linear program (LP) can be approximated with a constant factor in a constant number of rounds. The fractional versions of many important problems such as minimum dominating set, maximum matching, or certain flow problems fall into this category. The result is especially interesting in light of recent lower bounds for fractional covering and packing problems which show that on general graphs for a constant approximation, at least $\Omega(\sqrt{\log n/\log\log n})$ rounds are required [16, 17].

In Section 5, we give a deterministic algorithm which computes an $(\mathrm{O}(1),\mathrm{O}(1))$-decomposition in $\mathrm{O}(\log^* n)$ rounds on a UBG if the underlying metric is doubling. A $(d(n),c(n))$-*network decomposition* of a graph $G=(V,E)$ is a partition of V in 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 [5]. A network decomposition is a very basic structure which can be used as the basis of distributed algorithms for a huge number of problems. For instance, given a $(d(n),c(n))$-decomposition a maximal independent set (MIS) or a $\Delta+1$-graph coloring can be computed in time $d(n)\cdot c(n)$ (Δ is the largest degree of the graph). First all clusters of the first color compute a MIS/coloring in parallel in time $d(c)$. Subsequently, the clusters of the next colors add their contributions to the MIS/coloring. In a similar way, a $c(n)$-approximation for minimum dominating set can be computed in time $d(n)$. Here, the clusters of different colors do not have to wait for each other. Another essential application of network decomposition is the synchronization of asynchronous systems as introduced by Awerbuch [4]. Note that the time complexity $\mathrm{O}(\log^* n)$ of our decomposition algorithm is asymptotically optimal due to the matching $\Omega(\log^* n)$-lower bound for computing a MIS on a ring [20].

All our algorithms are formulated for the synchronous message passing model. Time is divided in rounds. In each round, each node can send a message to each of its neighbors.

The rest of the paper is organized as follows. Section 2 discusses related work. The technical results are presented in Sections 3, 4, and 5. The paper is concluded in Section 6

2. RELATED WORK

Unit disk graphs have been used in a great number of papers on ad hoc and sensor networks. Especially interesting in the context of the present paper are local distributed approximation algorithms for problems such as the minimum dominating set problem which is used in order to cluster wireless networks [2, 10]. Also closely related to our work are distributed algorithms which locally construct sub-graphs of the UDG with certain desirable properties (spanner, planar, etc.), a task which is usually called topology control [28, 27]. In ad hoc and sensor networks, nodes are often assumed to know the distances to their neighbors or even their coordinates. In [8], distances are used to construct local coordinate systems which can then for instance be used for routing. Other applications such as geometric routing [19] or location services [1] build on the fact that nodes even know their coordinates in the plane.

Bounded growth metrics in general and doubling metrics in particular have found quite a lot of attention recently [12, 14, 15, 24, 26]. It is proposed that latencies of many real networks such as peer-to-peer networks or the Internet are

[1]In fact it is even sufficient that all nodes know the distances to their neighbors up to a constant factor and that the triangle inequality holds up to a constant factor.

1	2	3	4	1	2	3	4
7	8	5	6	7	8	5	6
3	4	1	2	3	4	1	2
5	6	7	8	5	6	7	8
1	2	3	4	1	2	3	4

Figure 1: Coloring of the grid with 8 colors

doubling. The network-related problems which are solved include metric embeddings [12, 14], distance labeling and compact routing [26], and nearest neighbor search [15]. The doubling dimension has been introduced in [12], however, a similar notion has already been used in [3].

The concept of network decomposition has been introduced in [5] in which the 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 where $f(n) = n^{\sqrt{\log\log n/\log n}}$. They also showed how their decomposition algorithm can be used in order to obtain deterministic $O(f(n)^c)$ algorithms for finding a maximal independent set or a $(\Delta + 1)$ coloring in G. Building on earlier results in [6], it was shown in [21] that every graph G admits a $(\log n, \log n)$-decomposition. Algorithm [21] gives a randomized distributed algorithm with expected running time $O(\log^2 n)$ that computes such a decomposition. The deterministic algorithm of [5] was subsequently improved in [23], yielding a $(g(n)^d, g(n))$-decomposition in time $O(g(n)^d)$, where d is a constant and $g(n) = n^{\sqrt{1/\log n}}$. For unit disk graphs, the fastest known (randomized) algorithm to compute a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition is to first compute a MIS in expected time $\mathrm{O}(\log n)$ using an algorithm by Luby [22], the nodes are then clustered around the MIS nodes which become cluster leaders.

3. GLOBAL COORDINATES

In the literature, wireless ad-hoc and sensor networks are often modeled as unit disk graphs. Additionally, many algorithms assume that nodes have access to coordinate information [1, 2, 19, 27, 28]. Nodes obtain this information from a positioning system such as GPS either directly or by running a distributed positioning algorithm. In this section, we will have a closer look at this graph model and give a simple construction of a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition. We assume that nodes know their coordinates in the Euclidean plane and that two nodes can communicate directly if and only if their distance is at most 1 (UDG model). Note that this means that nodes "see" a common global coordinate system.

Having a global coordinate system enables to compute a decomposition as described in Section 1. We use the standard trick of partitioning the plane by a grid into square cells of side length $1/\sqrt{2}$. Each square cell defines a cluster of nodes. By checking their coordinates, nodes can decide in which cell they are located and hence to which cluster they belong. Since the length of the diagonal of a single square cell is 1, the induced graph of each cluster is a clique. A proper coloring of the cluster graph is obtained by globally coloring the grid such that no two cells whose distance is less than 1 are colored with the same color. Figure 1 shows how this can be achieved using 8 colors. Hence, by assigning each cluster the respective color, we obtain a $(1, 8)$-decomposition.

It is of course not surprising that global information such as coordinates helps devising fast distributed algorithms. The possibility of computing a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition from UDG coordinates alone indicates the power of such coordinate information. It means that unit disk graph coordinates suffice to compute essentially everything which can be computed locally in a constant number of rounds. In the next section, we will see that the described simple algorithm for computing a network decomposition can even be applied in some form in absence of global information.

4. FRACTIONAL COVERING AND PACKING PROBLEMS

In most cases, it is not realistic to assume that there is a positioning system which nodes could use to obtain coordinate information. In this section, we show that the main ideas of the last section can be adapted in order to solve many interesting problems in the case where no global information is present.

We again consider the standard unit disk graph model. In addition to knowing the direct neighbors, we merely assume that nodes can sense the distances to their neighbors. By exchanging this information for a few rounds, this enables the nodes to build up a *local coordinate system*. That is, distances between nodes can be used to compute angles and to learn about the geometry of the neighborhood. It is however not possible to align all those local coordinate systems, each node has its own local view. Assume for instance that we want to compute a small dominating set. In the presence of global coordinates, we can compute a network decomposition as described in the last section. Choosing one node per cluster (e.g., the node with the largest ID) gives a dominating set which is only by a constant factor larger than an optimal dominating set. If we try to do this with the local coordinate systems, the clusters of different local system will be different. Hence, also the selected nodes (dominating set) will be different in each coordinate system. This can lead to disastruous solutions and does not yield a non-trivial approximation.

While all the local coordinate systems inherently differ from each other, the set of all possible global coordinate systems is the same at every node. Hence, if we computed all dominating sets corresponding to the clusterings of all (infinitely many) different global coordinate systems, all nodes would come up with their local part of the same (multi-)set of different global dominating sets. It is of course still not possible to globally select one of these dominating sets. However, if we assign values 0 and 1 to non-dominators and dominators, respectively, it is possible to compute the *average* over all dominating sets. This does not result in a global dominating set, however it does result in a common fractional dominating set solution, that is, we solve the natural LP relaxation of the dominating set problem.

In the following, we present an explicit and more general algorithm for the above intuitive description. We consider fractional covering and packing problems. A covering problem (PLP) and its dual packing problem (DLP) are linear programs of the form

$$\begin{array}{llcll} \min & \underline{c}^{\mathrm{T}}\underline{x} & \qquad & \max & \underline{b}^{\mathrm{T}}\underline{y} \\ \text{s.t.} & A\cdot\underline{x}\geq\underline{b} & & \text{s.t.} & A^{\mathrm{T}}\cdot\underline{y}\leq\underline{c} \\ & \underline{x}\geq\underline{0} & & & \underline{y}\geq\underline{0} \end{array}$$

where all entries of A, b, and c are non-negative. The dual LP of a covering LP is a packing LP and vice versa. We assume that all variables x_i and y_i represent some value in the graph, that is, they belong to some node or edge. We assume that the conditions of the LP are local in the sense that whenever a primal (dual) variable x_i (y_j) occurs in the inequality corresponding to a dual (primal) variable y_j (x_i), x_i and y_j are separated by at most a constant number of hops in the network graph. This locality condition is true in all natural network coordination problems such as minimum dominating set (MDS), maximum matching (MM), etc. In MDS, for each nodes v_i there is a primal variable x_i and a dual variable y_i. The primal feasibility condition demands that the sum of the x-values in the 1-neighborhood of all nodes is at least 1, the dual feasibility condition states that the sum of the y-values of each 1-neighborhood is a most 1. Hence, only variables of adjacent nodes occur together in an inequality of the MDS LP. For MM, x-variables are associated with nodes and y-variables correspond to edges. Again, only adjacent variables occur together in the same inequality.

We will now first look at a solution of such LPs based on the network decomposition of Section 3. We will then show, how to convert this into a solution which does not need global coordinates using the idea of averaging over the set of all possible solutions.

Assume that we are given a $(1, \mathrm{O}(1))$-decomposition as described in Section 3. By exchanging the IDs among direct neighbors, each cluster can select the node with the largest ID as leader. In parallel, each leader then computes a local LP such that the combined local solutions form a constant approximation for (PLP) or (DLP). Let v_0 be the leader of some cluster $\mathcal{C}_0$. Let $\mathcal{Y}_0$ be the set of all dual y-variables which belong to nodes at distance at most 1 from v_0 or to edges which are adjacent to neighbors of v_0. The set $\mathcal{Y}_0$ has a corresponding set $\mathcal{E}_0$ of primal inequalities of (PLP). Let $\mathcal{X}_0$ be the set of primal x-variables which occur in the inequalities $\mathcal{E}_0$. Let P_0 be the covering problems consisting of the objective function of (PLP) and the inequalities $\mathcal{E}_0$. P_0 is a LP on the variables $\mathcal{X}_0$. Further, let D_0 be the packing LP which is obtained by deleting all variables from (DLP) which are not in $\mathcal{Y}_0$. That is, we restrict the matrix A to the rows and columns defined by $\mathcal{Y}_0$ and $\mathcal{X}_0$, respectively. By the definition of (PLP) and (DLP), the nodes and edges of the variables in $\mathcal{X}$ and $\mathcal{Y}$ are all within constant distance from v_0. Thus, v_0 can locally solve P_0 and D_0 in a constant number of rounds. The local solutions for each cluster can be combined by summing up the values of all local LPs for each variable.

Lemma 4.1. *Summing up the described local LPs for all clusters yields solutions for (PLP) and (DLP) with the same value of the objective function. The solution of (PLP) is a feasible constant approximation, the solution of (DLP) can be made feasible by dividing each y-variable by a constant factor.*

Proof. We start by proving the feasibility of (PLP). Because all clusters have diameter 1, all dual y-variables are in the set $\mathcal{Y}_i$ of at least one cluster leader v_i. Therefore, every inequality of (PLP) occurs in some local LP P_i. Because the x-values of all local LPs are summed up, it is sufficient to make every primal covering constraint feasible once in order to obtain a globally feasible solution for (PLP).

For the almost-feasibility of (DLP), observe that we have chosen the set $\mathcal{Y}_0$ such that the solution of the local dual problem D_0 is feasible for (DLP) (set all unused y-variables to 0): Clearly all inequalities which appear in D_0 are also feasible for (DLP); because all inequalities of (DLP) containing a variable $y_i \in \mathcal{Y}_0$ also appear in D_0, all other inequalities of (DLP) are of the form $0 \leq c_j$ for some j. Because of the locality condition for our LPs, all $x_i \in \mathcal{X}_0$ are at a constant distance from v_0. Therefore, each x_i can only occur in a constant number of local covering LPs. Because there is a one-to-one correspondence between primal variables and dual inequalities, each dual inequality can as well only occur in a constant number of local LPs. Because each local LP is dual-feasible for (DLP), this means the the sum of all local LPs is dual feasible for (DLP) up to a constant factor.

If all local LPs are solved optimally, the values of the objective functions for a pair (P_0, D_0) of local LPs are equal. Therefore, clearly, when summing up the local LPs, we get the same objective function values for (PLP) and (DLP) as well. By LP duality, the approximation factor of (PLP) is at most equal to the constant factor by which the dual inequalities have to be divided in order to obtain a feasible (DLP) solution. $\square$

We will now show, how to average the described solution over all possible coordinate systems. Equivalently to averaging the x and y values for all possible coordinate systems, we can choose one coordinate system uniformly at random[2] and compute the *expected values* for the x and y variables. In the above description, we have chosen the local LPs such that they are independent of the nodes' assignments to clusters. They only depend on the choice of the cluster leaders. Hence, each node v_i can compute its local LP. Let p_i be the probability that v_i is a cluster leader if the coordinate system is chosen uniformly at random. If we assume that every node v_i can compute its p_i, a constant-factor approximation to a given covering or packing LP can be computed as follows.

1. compute local LP and p_i
2. increase all variables x_j or y_j of LP by $p_i x_j$ or $p_i y_j$, respectively
3. if LP is a packing problem, divide by appropriate constant factor

It remains to show that p_i can really be computed. We will present an elegant way to approximate p_i up to a small constant factor. By the construction of the network decomposition of Section 3, p_i is the probability that v_i is the node with the largest ID within its cell of a random square grid of cell size $1/\sqrt{2}$. Hence, p_i is the probability that v_i has the largest ID in a random square or side length $1/\sqrt{2}$ containing v_i. Because for every square of side length $1/\sqrt{2}$,

[2]In principle, this means that the origin and the direction of the x-axis are chosen uniformly at random

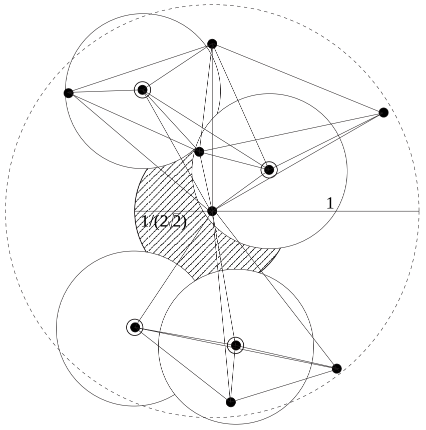

Figure 2: Computation of p'_i: Node v_i is at the center, the encircled nodes are the neighbors of v_i having a larger ID than v_i. The shaded area is proportional to p'_i.

there is a circle of diameter $1/\sqrt{2}$ which is completely inside the square, the probability p'_i of having the largest ID in a random circle of diameter $1/\sqrt{2}$ is $p'_i \geq p_i$. Using p'_i instead of p_i in the above algorithm therefore guarantees that the computed (PLP) solution is feasible.

We will now argue that the objective functions are not affected too much by using p'_i instead of p_i. Let p''_i be the probability that v_i is the node with largest ID in a square of side length 1/2 containing v_i. Taking p''_i instead of p_i corresponds to making the network decomposition with a grid of cell size 1/2 instead of $1/\sqrt{2}$. Using p''_i in the above algorithm would therefore give a solution which is at most by a factor 2 worse than the solution when using p_i because the area of each cell is smaller by a factor 2. Hence, the number of neighboring clusters in the decomposition doubles. Additionally, we have that $p''_i \geq p'_i$ because every circle of diameter $1/\sqrt{2}$ completely contains a square with side length 1/2. Thus, taking p'_i instead of p_i in the described algorithm results in a feasible solution for (PLP) which is worse than the solution using p_i by at most a factor 2.

The probability p'_i can be computed by v_i as follows.

1. exchange 1-hop distances with neighbors
2. compute angles between adjacent neighbors
3. geometrically arrange neighbors in one possible way.
4. For some node v, let $D(v)$ be the disk with radius $1/(2\sqrt{2})$ around v. Further, let $N^+(v)$ be the set of neighbors of v which have a larger ID than v. The probability p'_i can be computed as the area of

$$D(v_i) \setminus \bigcup_{u \in N^+(v_i)} D(u)$$

divided by the area of $D(v_i)$. That is, p'_i is the fraction of $D(v_i)$ which is not covered by any of the disks $D(u)$ with $\mathrm{ID}(u) > \mathrm{ID}(v_i)$.

Figure 2 illustrates step 4 of the described algorithm.

Lemma 4.2. *The above algorithm correctly computes the probability p'_i that v_i is the node with the largest ID in a random circle of diameter $1/\sqrt{2}$ containing v_i.*

Proof. We first assume that step 3 of the algorithm is unique, that is, we have a local coordinate system where v_i and its neighbors are correctly geometrically arranged. Choosing a random circle of diameter $1/\sqrt{2}$ containing v_i can be done by placing the center of the circle uniformly at random in the disk of radius $1/(2\sqrt{2})$ around v_i. For a given center p, v_i has the largest ID if there is no node v_j with $\mathrm{ID}(v_j) > \mathrm{ID}(v_i)$ at distance at most $1/(2\sqrt{2})$ from p. Therefore, v_i does have the largest ID if and only if the center p is chosen at distance more than one from all neighbors v_j of v_i with $\mathrm{ID}(v_j) > \mathrm{ID}(v_i)$. This exactly is the case if p is in the area which is computed in step 4 of the algorithm. Hence, the lemma is true if step 3 is unique.

Let $N(v_i)$ be the induced graph of v_i's neighbors (not including v_i), that is, the edges of $N(v_i)$ are all edges between neighbors of v_i. We start with the case where $N(v_i)$ consists of a single component. If we know the distances to two adjacent neighbors as well as the distance between those two neighbors, we can compute the angle at v_i between the two neighbors. If $N(v_i)$ is a single component, we can then find the angles between all neighbors of v_i. The geometry of the 1-neighborhood of v_i is therefore determined up to rotation around v_i, that is, we can compute a local coordinate system for which step 3 of the algorithm is unique.

Step 3 of the algorithm is not unique if $N(v_i)$ consists of several connected components. The geometry of each component can be determined, however the angle between differenct components can not be infered from the knowledge of the distances between neighbors alone. However, two nodes from different components of $N(v_i)$ are at distance more than 1 from each other. Therefore, the disks of radius $1/(2\sqrt{2})$ around two nodes $u, u' \in N(v_i)$ do not intersect if u and u' belong to different components in $N(v_i)$. Thus, the area which is computed in step 4 is the same for all possible geometric arrangments of the neighbors of v_i. $\square$

The results of this section are summarized in the upcoming theorem. The time bound for the fractional dominating set problem follows because in this case the given algorithm becomes particularly simple. The local LPs can be solved in this case by assigning 1 to all cluster leaders and 0 to all other nodes. Thus, the values p'_i form a constant approximation for minimum fractional dominating set.

Theorem 4.3. *In the given UDG model where distances are known, all local fractional covering and packing problems can be approximated up to a constant factor in constant time. In particular, the fractional minimum dominating set problem can be approximated in a single round.*

From a complexity theoretic point of view, looking at fractional distributed problems is extremely interesting. On the one hand, they usually still have the main properties of their corresponding integer problems, on the other hand, some of the synchronization problems occuring for the integer variants of the problems can be avoided. However, with some exceptions [7], in practice we are mostly interested in integer solutions. In the case of covering and packing problems, randomized rounding can be used in order to convert fractional solutions into reasonable approximations for the integer problems [25]. In [18, 16], an efficient distributed formulation of randomized rounding is given.

To conclude this section, we would like to highlight an intriguing comparison concerning the distributed complexity on unit disk graphs and on general graphs. In [17], we have shown that on general graphs, approximating minimum fractional dominating set up to a constant factor needs time at least $\Omega(\sqrt{\log n/\log\log n})$. The fact that in the given unit disk graph model, we can compute a constant approximation in a single communication round shows that there can be a large gap between the distributed complexity of problems on the unit disk graph and on general graphs.

5. NETWORK DECOMPOSITION

In the last section, we have seen that in the unit disk graph, knowing the distances to direct neighbors is enough to reasonably approximate important problems such as minimum dominating set in just one round or a constant number of rounds. If we want to compute more sophisticated structures such as a maximal independent set or even a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition, the methods of Section 4 cannot be used. It is in fact not hard to see that it is not even possible to construct a MIS or a decomposition in a constant number or rounds. In [20], Linial proved that computing a MIS on a ring needs at least $\Omega(\log^* n)$ rounds. Because a ring where all edges have length 1 is a unit disk graph, this lower bound applies to our model. Note that it does not help to know the edge lengths if all edges have length 1. In this section, we will show that in the model of Section 4, it is indeed possible to compute a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition in $\mathrm{O}(\log^* n)$ rounds. Our result even holds in a more general model where nodes can "live" in an arbitrary metric space instead of the Euclidean plane as in the UDG model. Analogously to the UDG model, we define the *unit ball graph* where two nodes are connected by an edge if and only if their distance is at most 1. Our result also applies in different related models where e.g. the distance between two nodes reflects the propagation delay of messages between the two nodes. The quality of the network decomposition that we achieve depends on the doubling dimension of the underlying metric.

5.1 Basic Algorithm

In this section, we will now first present a (potentially slow) deterministic distributed algorithm which computs a $(2, \mathrm{O}(1))$-decomposition. In a second step (Section 5.2), we will then show how the algorithm can be implemented such that its runtime is $\mathrm{O}(\log^* n)$. For the slow version of the algorithm, we assume that all nodes know the minimum distance $d_{\min}$ between any two nodes. This assumption would not be necessary. However, making the assumption results in a simpler, easier to understand algorithm. For the fast implementation, the assumption is not needed anymore. The computing of the decomposition is described by Algorithm 1.

Algorithm 1 starts with a small radius r which is increased by a factor 2 in every iteration of the while loop. At the beginning, the set $\mathcal{V}$ of possible cluster leaders contains all nodes. In each iteration, a subset of the nodes is selected such that the nodes selected in the subset form a maximal independent set on the graph of all edges of length $\leq r$.

LEMMA 5.1. *Algorithm 1 computes a $(2, 2^{4\rho})$-decomposition where ρ is the doubling dimension of the underlying metric. The maximum degree of the cluster graph is at most $2^{4\rho} - 1$.*

Algorithm 1 Network Decomposition: Clustering

```
1: $r := \min\{2^\lambda | \lambda \in \mathbb{Z} \wedge 2^\lambda \geq d_{\min}\}$;
2: $\mathcal{V} := V$;
3: while $r \leq 1/2$ do
4:     $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with $\mathcal{E} = \{\{u,v\} | d(u,v) < r\}$;
5:     compute MIS on $\mathcal{G}$;                                   [9, 20]
6:     $\mathcal{V} := \{v \in \mathcal{V} | v$ in MIS$\}$;
7:     $r := r \cdot 2$
8: od;
9: All nodes in $\mathcal{V}$ are cluster leaders, the other nodes belong
   to the cluster of the nearest leader.
10: Let $\Delta_C$ be the maximum degree of the cluster graph
    $G_C$. Color $G_C$ with $\Delta_C + 1$ colors.                     [9, 20]
```

PROOF. We first prove that each node has a cluster leader at distance at most 1 and that therefore, the diameter of each cluster is at most 2. The algorithm maintains a set $\mathcal{V}$ of nodes which are candidates for becoming cluster leader. In each iteration, some nodes are removed from $\mathcal{V}$. We have to prove that for all nodes u which are removed, there is a node v with $d(u,v) \leq 1$ which stays in $\mathcal{V}$ until the end, that is, v becomes cluster leader. Let $r_u = 2^{\lambda_u}$ ($\lambda_u \in \mathbb{Z}$) be the radius at which u is removed from $\mathcal{V}$. Whenever a node is removed from $\mathcal{V}$, there is a node at distance at most r which stays in $\mathcal{V}$. Otherwise, the independent set which is computed in line 5 is not maximal. Hence, after removing u, there is a node $u_0 \in \mathcal{V}$ with $d(u, u_0) \leq r_u$. If u_0 is removed in the subsequent iteration, there is a node u_1 with $d(u_0, u_1) \leq 2r_u$ which remains in $\mathcal{V}$. We hence get a sequence $u_0, u_1, \ldots, u_i, \ldots$ of nodes where $d(u_{i-1}, u_i) \leq 2^i r_u$ such that u_i remains in $\mathcal{V}$, i iterations after the removal of u. Summing up the distances results in a geometric series. For the distance between u and u_i, we therefore get

$$d(u, u_i) \;\leq\; \sum_{j=0}^{i} 2^j r_u \;<\; 2^{i+1} r_u \;=\; 2r_{u_i},$$

where r_{u_i} is the radius of the iteration where node u_i remains in $\mathcal{V}$ and where u_{i-1} is removed from $\mathcal{V}$. Let v be the last node in the sequence, that is, v is a cluster leader. Because the radius of the last iteration of Algorithm 1 is $1/2$, we have $d(u,v) < 1$. Thus, the radius of each cluster is at most 1.

It now remains to show that the maximum degree Δ_C of the cluster graph is at most $2^{4\rho} - 1$. On the one hand, from the last iteration of the algorithm ($r = 1/2$), it is guaranteed that the distance between any two cluster leaders is more than $1/2$. Otherwise, the nodes of the MIS of line 5 would not be independent. Therefore, each ball of radius $1/4$ or smaller contains at most one cluster leader. On the other hand, because the radius of each cluster is at most 1, the distance between two cluster leaders of adjacent clusters is at most 3. This means that for a cluster leader v, all leaders of adjacent clusters are in $B_3(v)$, the ball with radius 3 around v. By the definition of ρ, $B_3(v)$ can be covered by at most $2^{4\rho}$ balls of radius $3/16 < 1/4$. Including v, the number of cluster leaders in $B_3(v)$ is therefore at most $2^{4\rho}$. □

We will now have a close look at the complexity of a single iteration of the while loop of Algorithm 1. From a complexity point of view, the most important part is the computation of the MIS in line 5. Everything else (computing the neighbors in $\mathcal{G}$ and informing neighbors about new $\mathcal{V}$) can be done in a constant number of rounds. The time complexity for computing a MIS by a distributed algorithm depends on the maximum degree Δ of the graph. For small Δ, the fastest known algorithms are based on coloring algorithms. A coloring with K colors can be turned into a MIS in K rounds of communication. Thus, if we can color a graph with K colors in t rounds, we can compute a MIS in $t+K$ rounds.[3] In [9], an extremely elegant algorithm which colors a graph with 3^{Δ} colors in $O(\log^* n)$ rounds is described, resulting in a MIS algorithm with time complexity $O(\log^* n + 3^{\Delta})$. The algorithm was improved in [11] where an algorithm for computing a MIS in time $O(\log \Delta(\Delta^2 + \log^* n))$ is given. In [20], it is shown that it is even possible to color a graph with $O(\Delta^2)$ colors in $O(\log^* n)$ rounds giving a time complexity of $O(\Delta^2 + \log^* n)$ for computing a MIS. The proof of [20] is based on the existence proof of a set system with certain properties. It is shown that such a set system exists and that it can be computed efficiently by a randomized algorithm. If Δ is constant, all three algorithms compute a MIS in $O(\log^* n)$ rounds. Lemma 5.2 bounds the maximum degree of $\mathcal{G}$.

Lemma 5.2. *In each iteration of Algorithm 1, the maximum degree of $\mathcal{G}$ is at most $2^{2\rho}$.*

Proof. Let ℓ be the length of the minimum distance between any two nodes of $\mathcal{G}$. Because the algorithm computes an independent set in each iteration, we have $\ell > r/2$. Therefore, every ball of radius $r/4$ contains only one node. All neighbors of a node $v \in \mathcal{V}$ are in the ball $B_r(v)$ of radius r around v. By the definition of the doubling dimension ρ, $B_r(v)$ can be covered by $2^{2\rho}$ balls of radius $r/4$. Therefore, the number of nodes in $B_r(v)$ is at most $2^{2\rho}$. □

When using the algorithm of [20] for computing the MIS, Lemma 5.2 implies the following corollary.

Corollary 5.3. *The time complexity of a single iteration of the while loop of Algorithm 1 is $O(\log^* n + 2^{4\rho})$, that is, for constant doubling dimension, the time complexity is $O(\log^* n)$.*

Before coming to the description of a faster implementation the while loop of Algorithm 1, we have a look at the complexity of lines 9 and 10. By Lemma 5.1, we know that each node has a cluster leader in its neighborhood. Line 9 thus can be computed in a single communication round. The time complexity of line 10 is more interesting. Similar to the construction of a MIS, a distributed coloring algorithm which colors a graph with K colors in time t can be turned into a coloring algorithm whith colors a graph with $\Delta + 1$ colors in time $t + K$ (Δ is the maximum degree). Algorithm 2 shows how this can be achieved.

By Lemma 5.1, the maximum degree of the cluster graph is at most $2^{4\rho} - 1$. If we use the algorithm of [20] for computing the initial coloring, this results in the following corollary.

Corollary 5.4. *The time complexity of line 10 of Algorithm 1 is $O(\log^* n + 2^{8\rho})$, that is, for constant doubling dimension, the time complexity is $O(\log^* n)$.*

[3]Converting a coloring to a MIS works essentially in the same way as the reduction of the number of colors which will be discussed in Algorithm 2. In fact, all nodes of color 1 form a MIS after applying algorithm 2

Algorithm 2 Color Reduction (node v_i)

```
Input: coloring with colors $\{1, \ldots, K\}$
Output: coloring with colors $\{1, \ldots, \Delta + 1\}$
1: for $c := 1$ to $K$ do
2:    send color($v_i$) to all neighbors;
3:    if color($v_i$) $= c$ then
4:       color($v_i$) := minimal possible color
5:    fi
6: od
```

5.2 Fast Implementation of the Basic Algorithm

In this section, we will have a second look at the complexity of Algorithm 1 resulting in the main result of this paper. We need to start with a few general considerations concerning the synchronous message passing model. If nodes communicate for k rounds, they can only gather information which is at most k hops away. In principle, every distributed k-round algorithm can be formulated as follows.

1. Collect complete k-neighborhood in graph in k communication rounds

2. Compute the output by locally simulating the relevant part of the distributed algorithm (no communication needed)

Collecting the complete k-neighborhood can be achieved if all nodes send their complete states to all their neighbors in every round. After round i, all nodes know their i-neighborhood. Learning the i-neighborhoods of all neighbors in round $i + 1$ suffices to know the $i + 1$-neighborhood. The above formulation of a distributed algorithm of course has the drawback that messages can get extremely large. We will show that the message size can be kept moderate in our example.

Let us again consider a single iteration of the while loop of Algorithm 1. All communication which is needed to compute an iteration of the while loop is on $\mathcal{G}$. Hence, all messages are sent on edges which have length at most r. If we communicate for k rounds and if all messages of those k rounds are on edges of length at most r, then all collected information comes from distance at most $k \cdot r$. In order to be able to compute everything locally, the nodes have to collect the complete neighborhood up to distance kr (w.r.t. the metric). That is, the nodes have to collect all information which is accessible by paths of length at most kr. Note that it is not necessary and it might not be possible to collect the whole ball of radius kr. Because of the triangle inequality, it is possible to collect this information in $2kr$ rounds. Applying this to Algorithm 1, we get Lemma 5.5.

Lemma 5.5. *Algorithm 1 can be computed in $O(\log^* n + 2^{8\rho})$ rounds, that is, for constant doubling dimension, the time complexity can be reduced to $O(\log^* n)$.*

Proof. By Corollary 5.3, the number of rounds of an iteration of the while loop of Algorithm 1 is $O(\log^* n + 2^{4\rho})$. Nodes therefore need to collect information from distance at most $O(r(\log^* n + 2^{4\rho}))$. To obtain the distance from which we need information in order to be able to locally compute the results of all iterations of the while loop, we have to sum up the distances for all iterations. We do not know the number of iterations. However because r grows exponentially by a factor of 2 in each iteration, we have a geometric series and can upper bound the sum by taking 2 times the maximum summand. Therefore, the whole while loop can be computed in $\mathrm{O}(\log^* n + 2^{4\rho})$ rounds. Together with Corallary 5.4, we get the required result. $\square$

Note that when collecting the whole neighborhood, it is not necessary that nodes know the minimum distance $d_{\min}$ between nodes. Because the radius grows exponentially, the locality of the problem is independent of the starting radius. Each node can just use the smallest distance in the collected neighborhood in order to locally simulate the distributed Algorithm 1. The complete algorithm to compute a $(\mathrm{O}(1), \mathrm{O}(1))$-decomposition in the given network model can be summarized as follows.

1. exchange 1-hop distances with neighbors
2. locally compute the while loop of Algorithm 1 for $r \in \mathrm{O}(1/(\log^* n + 2^{4\rho}))$ (up to the radius for which it suffices to know the 1-neighborhood).
3. collect $\mathrm{O}(\log^* n + 2^{4\rho})$-neighborhood (it is sufficient to only collect data about nodes which are still in $\mathcal{V}$)
4. compute the remaining iterations of the while loop
5. compute clusters and cluster coloring (lines 9,10 of Algorithm 1)

Computing the solution for small radii first and then collecting the rest of the neighborhood is done in order to obtain reasonable message sizes. We are now ready to formulate our main theorem.

Theorem 5.6. *In the unit ball graph model, the above algorithm computes a $(2, 2^{4\rho})$-decomposition in time $\mathrm{O}(\log^* n + 2^{8\rho})$ where ρ is the doubling dimension of the underlying metric. Given that all distances and node IDs can be represented by K bits, the maximal message size is at most*

$$\mathrm{O}\left[\left(\log^* n + 2^{4\rho}\right)^{\mathrm{O}(\rho)} + \Delta\right] \cdot K$$

bits. Hence, for constant ρ, the time complexity is $\mathrm{O}(\log^* n)$ and largest message needs at most $\mathrm{O}(((\log^* n)^{\mathrm{O}(1)} + \Delta)K)$ bits.

Proof. The time complexity follows from Lemma 5.5. For the correctness of the algorithm, it remains to prove that only collecting information about nodes in $\mathcal{V}$ for $r \geq \mathrm{O}(1/(\log^* n + 2^{4\rho}))$ (steps 3 and 4) is sufficient. Because all communication of Algorithm 1 is on $\mathcal{G}$, this is however clear.

For the bound on the message size, we need to have a closer look at steps 1, 3, and 5 where messages are exchanged. In step 1, all nodes send at most Δ distances and node IDs to their neighbors. This requires messages of size $\mathrm{O}(\Delta \cdot K)$. In step 3, a message can at most contain the whole R-neighborhood of a node, where $R := \mathrm{O}(\log^* n + 2^{4\rho})$. Let N be the maximum number of nodes which such a R-neighborhood can contain. If $r \in \Theta(1/(\log^* n + 2^{4\rho}))$ denotes the larges radius for which the while loop has been computed in step 2, we know that for all pairs of nodes $u, v \in \mathcal{V}$, we have $d(u, v) > r$. Therefore, balls of radius at most $r/2$ contain at most 1 such node. By the definition of ρ, the maximum number of nodes in a ball of radius R, therefore is

$$N \ \leq \ (2^\rho)^{(\log_2(R/r)+1)} \ = \ \left(\frac{R}{r}\right)^{\rho+1}.$$

The number of edges in the R-neighborhood is at most quadratic in N. By the definition of R and r, the theorem thus follows. $\square$

Remark 1:
Theorem 5.6 even holds if the nodes only know approximations of the distances to their neighbors or if the triangle inequality is not completely satisfied. If distances are known up to a constant factor and/or if the triangle inequality holds up to a constant factor, it is not hard to see that the results of this section remain true up to constant factors.

Remark 2:
The results of this section can be extended to other situations than the unit ball graph model. Assume for instance that we have given a doubling metric (X, d). All points in X have to provide their part of the solution of a global problem. Thereby, each member $x \in X$ has to base its decision the ball $B_r(x)$ for some radius r. Theorem 5.6 shows that chosing the radius $r \in \mathrm{O}(\log^* n)$ suffices for many natural problems. As a particular example, we might wish to construct an ϵ-net, that is, we want to select a set of points S such that any two selected points have distance at least ϵ and such that any point has a point in S at distance less than ϵ. In algorithms for metric spaces, ϵ-nets are a widely used structure [13]. Theorem 5.6 shows that every node can decide about whether it is in S based on its $\mathrm{O}(\epsilon \log^* n)$-neighborhood only.

6. CONCLUSIONS

The communication graphs typically encountered in wireless networking, peer-to-peer networks, or in the Internet tend to contain structure, which remains uncaptured when modeling them as general graphs. Specifically, these communication graphs often have the property of bounded volume growth.

In this paper, we have studied the distributed time complexity of many natural network coordination problems in growth-bounded graphs. In particular, we have presented an algorithm that computes a $(O(1), O(1))$-network decomposition in time $O(\log^* n)$, based only on the assumption that nodes know the distances to their neighbors. Using this decomposition algorithm, it is straightforward to obtain $O(\log^* n)$ time algorithms for computing a maximal independent set, a $\Delta + 1$ coloring, or a constant approximation to the minimum dominating set problem in graphs with bounded growth. The frequently studied unit disk graph being a growth-bounded graph, all these results directly carry over to unit disk graphs, thus greatly improving the fastest previously known algorithms for these problems in unit disk graphs.

Moreover, our result are asymptotically optimal due to a matching $\Omega(\log^* n)$ time-lower bound for computing a MIS on a ring [20]. We find it intriguing that from the point of view of locality, a simple toy-network such as the ring is as hard as the vast family of growth-bounded graphs.

7. REFERENCES

[1] I. Abraham, D. Dolev, and D. Malkhi. Lls: A locality aware location service for mobile ad hoc networks. In *Proc. of 2nd Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2004.

[2] K. Alzoubi, P.-J. Wan, and O. Frieder. Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHOC*, 2002.

[3] P. Assouad. Plongements lipschitziens dans $\mathbf{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.

[4] B. Awerbuch. Complexity of network synchronization. *Journal of the ACM (JACM)*, 32(4):804–823, 1985.

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

[6] B. Awerbuch and D. Peleg. Routing with polynomial communication-space trade-off. *SIAM J. Discret. Math.*, 5(2):151–162, 1992.

[7] Y. Bartal, J. W. Byers, and D. Raz. Global optimization using local information with applications to flow control. In *Proc. of the 38th IEEE Symp. on the Foundations of Computer Science (FOCS)*, pages 303–312, 1997.

[8] S. Capkun, M. Hamdi, and J. P. Hubaux. Gps-free positioning in mobile ad-hoc networks. *Cluster Computing*, 5(2), April 2002.

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

[10] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Discrete mobile centers. In *Proc. of the 17th annual symposium on Computational geometry (SCG)*, pages 188–196. ACM Press, 2001.

[11] A. Goldberg, S. Plotkin, and G. Shannon. Parallel symmetry-breaking in sparse graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 1(4):434–446, 1988.

[12] 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.

[13] J. Heinonen. *Lectures on Analysis of Metric Spaces.* Springer-Verlag, New York, 2001.

[14] 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.

[15] R. Krauthgamer and J. Lee. Navigating nets: Simple algorithms for proximity search. In *Proc. of 15th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, 2004.

[16] F. Kuhn, T. Moscibroda, and R. Wattenhofer. The price of being near-sighted. submitted, 2004.

[17] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be computed locally! In *Proc. of the 23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 300–309, 2004.

[18] F. Kuhn and R. Wattenhofer. Constant-time distributed dominating set approximation. In *Proc. of the 22nd ACM Symp. on Principles of Distributed Computing (PODC)*, pages 25–32, 2003.

[19] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric ad-hoc routing: Of theory and practice. In *Proc. of 22nd ACM Symp. on Principles of Distributed Computing (PODC)*, 2003.

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

[21] N. Linial and M. Saks. Low diameter graph decompositions. *Combinatorica*, 13(4):441–454, 1993.

[22] M. Luby. A simple parallel algorithm for the maximal independent set problem. *SIAM Journal on Computing*, 15:1036–1053, 1986.

[23] 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.

[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] P. Raghavan and C. D. Thompson. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374, 1987.

[26] 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.

[27] Y. Wang and X.-Y. Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. In *Proc. of 1st Joint Workshop on Foundations of Mobile Computing (DIALM-POMC)*, 2003.

[28] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proc. of 20th INFOCOM*, 2001.