Distributed Approximation Algorithms in Unit-Disk Graphs

A. Czygrinow[1] and M. Hańćkowiak[2]

[1] Department of Mathematics and Statistics
Arizona State University
Tempe, AZ 85287-1804, USA
andrzej@math.la.asu.edu
[2] Faculty of Mathematics and Computer Science
Adam Mickiewicz University
Poznań, Poland
mhanckow@amu.edu.pl

Abstract. We will give distributed approximation schemes for the maximum matching problem and the minimum connected dominating set problem in unit-disk graphs. The algorithms are deterministic, run in a poly-logarithmic number of rounds in the message passing model and the approximation error can be made $O(1/\log^k |G|)$ where $|G|$ is the order of the graph and k is a positive integer.

1 Introduction

In this paper we will give efficient distributed approximation algorithms for two important graph-theoretic problems, the Maximum Matching (MM) Problem and the Minimum Connected-Dominating Set (MCDS) Problem. Our algorithms work in the message passing model and assume that the underlying network has a unit-disk graph topology. Both problems are classical problems in graph theory with many important practical applications. For example, efficient solutions for the MCDS Problem are particularly interesting because of their applications to routing in mobile ad-hoc networks (see for example [DW04]).

Studying distributed message passing approximation algorithms for unit-disk graphs was initiated by Kuhn et. al. in [KMNW05b] where efficient approximation schemes for the Maximum Independent Set (MaxIS) Problem and the Minimum Dominating Set (MDS) Problem are given. Research described in the series of papers [KMW05], [KMNW05b], and [KMNW05a] is the main motivation for our work here. We give efficient distributed approximation schemes for two additional classical graph-theoretic problems. In addition, both of our procedures are based on a rather general clustering framework that almost immediately gives efficient solutions to MaxIS and MDS Problems with potential for further applications to unit-disk graphs. Second motivation for our study comes from a recent work [CH06] and [CHS06] in which distributed approximations for MaxIS Problem, MM Problem, and MDS Problem in planar graphs

S. Dolev (Ed.): DISC 2006, LNCS 4167, pp. 385–398, 2006.

are given. Unit-disk graphs form another important class of graphs where simple clustering methods give efficient distributed approximations. Since unit-disk graphs are commonly used to model mobile ad-hoc networks or static radio networks this line of research is also of clear practical importance. As in the case of [KMNW05b] or [KMW05], we assume that there is a lower level protocol that resolves transmission conflicts (MAC) and communication is done in the message passing model.

1.1 Model and Notation

We consider a standard, distributed, synchronous message-passing model described for example in Linial [L92] or in [P00]. In this model a network is represented by an undirected graph where vertices correspond to processors and edges to communication links between them. We assume that vertices have unique identifiers and that communication is synchronized. In a single round of an algorithm, each vertex of the graph can send messages to its neighbors, can receive messages from its neighbors, and can perform local computations. Although the above model allows unlimited local computations, we will pay attention not only to the distributed complexity of a problem but will also indicate the sequential running time at each processor. We will focus entirely on the time complexity analysis leaving message complexity issues for future research. Although the above model is certainly an oversimplified version of a real-life system, it perfectly captures the most fundamental challenge faced by a distributed algorithm; the problem of finding a global solution in a network based on local information about the topology of the graph which is available to each node.

In this paper we will consider graphs with the unit-disk graph topology. A graph $G = (V, E)$ is called a unit-disk graph if there is a function $f : V \to R^2$ such that $uv \in E$ if and only if $||f(u) - f(v)||_2 \leq 1$. Although the fact that G is a unit-disk graph is critical to our analysis, we will assume that no information about geometrical representation of G is available to nodes. In fact, if such information is available, distributed running time of our algorithms can be reduced significantly as it is possible to find a maximal independent set in a much faster way [KMW05].

Finally, we will use standard graph-theoretic notation and terminology. In particular, following the convention from [D97], we will denote by $|G|$ the number of vertices in G and by $||G||$ the number of edges.

1.2 Results

We will describe efficient distributed algorithms for the MM and MCDS problems in unit-disk graphs. All algorithms are deterministic and the running time is poly-logarithmic in the order of the graph. Depending on the desired approximation ratio the time is a polynomial in $\log |G|$ of higher or smaller order. In the same way, the amount of local computations can be made more or less time consuming based on the desired approximation error. For the MM Problem, we give a deterministic distributed algorithm which given a positive integer k and

real number $\alpha \geq 0$ finds in a unit-disk graph $G = (V, E)$ a set $M \subset E$ such that M is a matching in G and $|M| \geq (1-\alpha-O(1/\log^k |G|))\beta(G)$ where $\beta(G)$ is the size of the maximum matching in G (Theorem 2). For the MCDS Problem, we give a deterministic distributed algorithm which given a positive integer k and real number $\alpha \geq 0$, finds in a connected unit-disk graph G finds a set $D \subset V(G)$ such that D induces a connected subgraph of G, D is a dominating set in G, and $|D| \leq (1+\alpha+O(1/\log^k |G|))\gamma_c(G)$ where $\gamma_c(G)$ is the size of the smallest connected dominating set in G (Theorem 3). Both algorithms run in a poly-logarithmic (in $|G|$) number of rounds and the amount of local computations is at most $T^M_\alpha(|G|)$ or $T^C_\alpha(|G|)$, where $T^M_\alpha(|G|)$ is the sequential complexity of finding a $(1-\alpha)$-approximation of a maximum matching in unit-disk graphs and $T^C_\alpha(|G|)$ is the sequential complexity of finding a $(1+\alpha)$-approximation of a minimum connected dominating set in unit-disk graphs. In the case of maximum matching $\alpha = 0$ can be accomplished in polynomial time and although finding a minimum connected dominating set in unit-disk graphs is NP-complete (see [L82]), there are very fast constant error approximations and in the case a local geometric representation of the graph is available, $(1+1/s)$-approximation can be found in $O(|G|^{O((s \log s)^2)})$ time (see [CHLWD03]). In addition, our methods yield a sequential polynomial time approximation scheme (PTAS) for the MCDS Problem extending the work of Nieberg and Hurink from [NH05] where a PTAS for the MDS Problem is given.

1.3 Related Work

There has been recently an explosive growth of interest in complexity issues of algorithms for geometric graphs. This line of research has roots in computational geometry (see for example [GGHZZ01]) as well as in wireless networks. Distributed algorithms in the message passing model for unit-disk graphs have been recently studied by Kuhn et. al. in [KMW05], [KMNW05a], and [KMNW05b]. In particular, [KMNW05a] contains a $O(\log \Delta(G) \log^* |G|)$-time distributed algorithm for the maximal independent set problem. In [KMNW05b] in turn, authors gave distributed approximation schemes for the MaxIS Problem and the MDS Problem. All of the results, as well methods described in this work, exploit the bounded growth property of unit-disk graphs (see [KMW05] for an example of formalization or Section 2 for a different approach). Although methods use the same properties which are inherent to unit-disk graphs, the approaches differ significantly. In particular, algorithms from [KMNW05b] work entirely in an underlying network which has the unit-disk graph property. In contrast, the first phase of our algorithms works in an auxiliary graph which arises from a maximal independent set in a graph and it is this auxiliary graph which is further clustered to obtain partition of the original graph.

1.4 Organization

In the rest of the paper we shall first give our clustering algorithm (Section 2) and then discuss algorithms for the MM and MCDS problems (Section 3). Finally

we conclude with brief remarks on how the methods can be used to give a PTAS for the MCDS Problem in unit-disk graphs.

2 Clustering Algorithm

In this section, we will give a distributed algorithm that finds a clustering of a unit-disk graph. By clustering we mean a partition of the vertex set of a graph with the property that each set in the partition induces a connected subgraph. Of course, additional properties of the clustering will be important to obtain approximation algorithms. The clustering procedure works in two phases. In the first phase we use the $O(\log \Delta(G) \log^* |G|)$-time algorithm from [KMNW05a] to find a maximal independent set I in graph G. In the second phase an auxiliary graph is constructed from I and we invoke a clustering algorithm to find a a partition of it. The clustering of the auxiliary graph gives the final a clustering of G. The analysis of our algorithms heavily exploits the so-called bounded growth property of an auxiliary graph arising from a unit-disk graphs. Consequently the method described here is limited to unit-disk graphs or some natural generalizations. The following fact about unit-disk graphs will be key to our analysis.

Lemma 1. *Let G be a unit-disk graph and let k be a positive integer. For any independent set I in G and any geometrical representation of G, the number of vertices from I which are contained in a ball in R^2 of radius k is at most $4(k + 0.5)^2$.*

Although the constant 4 is certainly not best possible it is sufficient for out considerations and is easy to prove. Indeed, any packing of balls with radius 0.5 into a ball of radius $(k+0.5)$ can have at most $4(k+0.5)^2$ members. The second phase of clustering works entirely in the auxiliary graph obtained in the first phase and only after partition of the auxiliary graph is found we return to the original unit-disk graph G. We will first describe the second phase and then give the main clustering algorithm.

2.1 Second Phase of Clustering Algorithm

In this section, we give a clustering algorithm of a C-bounded growth graph (see the definition below). It is important to mention that our notion of a C-bounded growth graph is different and maybe less standard than a similar concept considered in [KMNW05a].

Definition 1. *A graph H has a C-bounded growth if for every vertex v from H and every nonnegative integer k the number of vertices within distance (in H) k of v is at most $Ck^2 + 1$.*

Our algorithm for graphs with C-bounded growth will use the ruling set method described in [AGLP89]. Let $G = (V, E)$ be a graph with identifiers of vertices from the set $\{1, \ldots, m\}$ where m is globally known and satisfies $|G| \leq m \leq poly(|G|)$ for some polynomial $poly(|G|)$. A *D-ruling set* in G is a subset S of V with two properties:

- For any two distinct vertices s, s' from S, the distance (in G) between s and s' is at least D.
- For any vertex $v \in V \setminus S$ there is a vertex $s \in S$ such that the distance between s and v is at most $D \log |G|$.

There is an easy distributed algorithm which finds a D-ruling set in any graph.

Theorem 1 ([AGLP89]). *There is a distributed algorithm which in any graph G finds a D-ruling set in $O(D \log |G|)$ rounds.*

Our algorithm uses parameters ϵ, D, and F (used in Clustering) which can be set to specific values yielding different running times and approximation factors. For $0 < \epsilon < 1$, and fixed C, let l_* be the smallest positive integer with the property

$$(1+\epsilon)^{l_*} \geq C l_*^2 + 1. \tag{1}$$

It is easy to check that

$$l_* = O(1/\epsilon^2). \tag{2}$$

In addition, let D be such that

$$D > 2l_*. \tag{3}$$

In the next two procedures we will find a clustering of a graph which has C-bounded growth. First procedure is essentially one iteration of the main algorithm. We will denote the set of vertices which have a neighbor in U by $N(U)$.

ClusterSet
Input: Constant C. Graph $H = (V, E)$ which has C-bounded growth and such that identifiers of V are bounded by m. Parameters: $0 < \epsilon < 1$ (arbitrary) and D (must satisfy (3)).
Output: A family of subsets of V.

(1) Find a D-ruling set, $\{v_1, v_2, \ldots, v_s\}$ in H.
(2) For every v_i in parallel:
  (a) Let $U_i := \{v_i\}$, $N_i := N(U_i) \setminus U_i$.
  (b) while $|N_i| \geq \epsilon |U_i|$
    - $U_i := U_i \cup N_i$. $N_i := N(U_i) \setminus U_i$.
(3) Return $U_1, U_2, \ldots, U_s$.

The analysis of ClusterSet can be divided into a few lemmas.

Lemma 2. *The number of vertices in the D-ruling set obtained in step one of* ClusterSet *is at least*

$$|H|/(CD^2 \log^2 |H| + 1).$$

Proof. For every vertex v_i, where $i = 1, \ldots, s$ from the ruling set, consider the set W_i of vertices in H which are within distance $D \log |H|$ of v_i. From Definition 1, $|W_i| \leq CD^2 \log^2 |H| + 1$. On the other hand, since v_i's form a D-ruling set, $|H| \leq |\bigcup_{i=s}^{l} W_i|$ and so

$$|H| \leq s(CD^2 \log^2 |H| + 1).$$

Next two lemmas show that U_i's have small diameter and more importantly the total number of edges that intersect two different U_i's is small.

Lemma 3. *Let l_* be such that inequality (1) holds and let $r(U_i)$ denote the radius of $H[U_i]$. Then*

$$r(U_i) \leq l_*.$$

Proof. Let $U_i^{(l)}$ denote the set U_i in the lth iteration of the while loop from step 2(b). Then $|U_i^{(0)}| = 1$ and in the lth iteration $|U_i^{(l)}| \geq (1+\epsilon)^l$. On the other hand, by Definition 1, $|U_i^{(l)}| \leq Cl^2 + 1$. Consequently, if l_* is the smallest positive integer such that $(1+\epsilon)^{l_*} \geq Cl_*^2 + 1$ then $l \leq l_*$. Since the radius of $G[U_i]$ is at most l, we have $r(U_i) \leq l_*$.

Lemma 4. *Sets U_i returned by* ClusterSet *are pair-wise disjoint. In addition, if $e(U_i, V \setminus U_i)$ denotes the number of edges between U_i and $V \setminus U_i$ then*

$$\sum_{i=1}^{s} e(U_i, V \setminus U_i) \leq C\epsilon |\bigcup_{i=s}^{l} U_i|.$$

Proof of Lemma 4. From Lemma 3 every U_i is such that $r(U_i) \leq l_*$. and so if $U_i \cap U_j$ is non-empty then the distance between v_i and v_j is at most $2l_*$ which contradicts the fact that v_i, v_j are in the D-ruling set where $D > 2l_*$ by (3). To prove the second part, note that for every U_i returned in step 3, the set $N_i = N(U_i) \setminus U_i$ is such that $|N_i| < \epsilon |U_i|$. Since H has the maximum degree of at most C, the number of edges between U_i and N_i is at most $C\epsilon|U_i|$.
Finally, we note that the running time of ClusterSet is $O(D \log m + 1/\epsilon^2)$.

Lemma 5. *The number of rounds of* ClusterSet *is $O(D \log m + 1/\epsilon^2)$.*

Proof. There are $O(D \log m)$ rounds to find the D-ruling set in step 1. This is followed by $l_* = O(1/\epsilon^2)$ iterations in step 2.

Our main clustering procedure will call ClusterSet a repeated number of times. In each call, sets $U_1, \ldots, U_l$ are obtained and vertices from $\bigcup U_i$ are deleted from the graph H. Finally, after trimming H with repeated application of ClusterSet, the remaining vertices will form one-element clusters.

Clustering
Input: Constant C. Graph $H = (V, E)$ which has C-bounded growth and such that the identifiers of V are less than or equal to m. Parameters: $0 < \epsilon < 1$ (arbitrary), D (must satisfy (3)), F (arbitrary).
Output: A partition $\mathcal{P}$ of V.

(1) Repeat F times:
   (a) Call ClusterSet in H. Add all sets U_i obtained from ClusterSet to family $\mathcal{P}$.
   (b) Delete from H vertices from $\bigcup U_i$ and edges incident to these vertices.
(2) For every vertex left in H create a set which contains only this vertex and add it to $\mathcal{P}$. Return $\mathcal{P}$.

We note the following property of the partition obtained by Clustering.

Lemma 6. *Let $\mathcal{P} = (V_1, \ldots, V_t)$ be a partition of V returned by* Clustering. *The number of edges of H connecting vertices from different V_i's is*

$$O\left(\left(\left(1 - \frac{1}{CD^2 \log^2 |H| + 1}\right)^F + \epsilon\right)|H|\right).$$

Proof. First note that since H is a graph with a constant maximum degree, $||H|| \leq C|H|/2$. Consider sets added to $\mathcal{P}$ in iterations from step 1. Edges which have exactly one endpoint in these sets are deleted in step 1(b) and by Lemma 4, the number of them is at most $C\epsilon|H|$. The remaining edges which must be counted are the edges of H from step two. To estimate these, we note that by Lemma 2, the number of vertices in this graph is $O\left(\left(1 - \frac{1}{CD^2 \log^2 |H|+1}\right)^F |H|\right)$. Consequently, the number of edges of H connecting vertices from different V_i's is $O\left(\left(\left(1 - \frac{1}{CD^2 \log^2 |H|+1}\right)^F + \epsilon\right)|H|\right)$.

Lemma 7. Clustering *runs in $O\left(F\left(D \log m + 1/\epsilon^2\right)\right)$ rounds.*

Proof. There are F iterations of step 1, in which, by Lemma 5, sets are found in $O\left(D \log m + 1/\epsilon^2\right)$ rounds.

Corollary 1. *Let C be a fixed constant. For a C-bounded growth graph H with identifiers that are less than or equal to m, there is a distributed algorithm which finds in $O(1/\epsilon^6 \cdot \log^3 m \log 1/\epsilon)$ rounds a partition of V such that the number of edges between different partition classes is $O(\epsilon|H|)$.*

Proof. Let $D := 2l_* + 1 = O(1/\epsilon^2)$ and let $F := \lceil (\ln 1/\epsilon)(CD^2 \log^2 H + 1) \rceil = O((\log 1/\epsilon)D^2 \log^2 m)$. Then the number of rounds is $O(F \log m/\epsilon^2) = O(1/\epsilon^6 \cdot \log^3 m \log 1/\epsilon)$ (Lemma 7). In addition, the number of edges which connect different clusters is $O(\epsilon|H|)$ by Lemma 6 as $\left(1 - \frac{1}{CD^2 \log^2 |H|+1}\right)^F = O(\epsilon)$.

2.2 Main Clustering Algorithm

We will now give the main clustering algorithm for unit-disk graphs. We first find a maximal independent set I using an algorithm from [KMNW05a] and then apply Clustering in the auxiliary graph that arises from I. This gives clusters in the original graph.

Definition 2 (Auxiliary graph). *Let $I = \{v_1, \ldots, v_l\}$ be a maximal independent set in graph $G = (V, E)$. Let V_i be the set of neighbors of v_i such that if $w \in V_i$ then v_i is the neighbor of w in I with the least identifier, i.e. $V_i = \{w \in N(v_i) | ID(v_i) = \min\{ID(a) | a \in N(w) \cap I\}\}$, and let $\bar{V}_i = V_i \cup \{v_i\}$. Let $Aux(G)$ be the graph $(\mathcal{W}, \mathcal{E})$ with $\mathcal{W} = \{\bar{V}_1, \ldots, \bar{V}_l\}$ and $\{\bar{V}_i, \bar{V}_j\} \in \mathcal{E}$ whenever $i \neq j$ and there is an edge in G between a vertex from $\bar{V}_i$ and a vertex from $\bar{V}_j$.*

From Lemma 1, we see that $Aux(G)$ has C-bounded growth with $C = 48$ as if there are p vertices within distance k of some vertex v in $Aux(G)$ then there are is an independent set of size p in a ball of radius $3k$. Consequently $p \leq 4(3k + 0.5)^2 \leq 48k^2 + 1$.

Lemma 8. *$Aux(G)$ has 48-bounded growth.*

In particular the maximum degree of $Aux(G)$ is at most 48.

ClusteringUDG
Input: Unit disk graph $G = (V, E)$, $0 < \epsilon < 1$ (arbitrary).
Output: Partition $\mathcal{Q}$ of V.

(1) Call Kuhn et. al. algorithm from [KMNW05a] to find a maximal independent set I. Consider the auxiliary graph $Aux(G)$.
(2) Call Clustering with constants as in Corollary 1 to find a partition $\mathcal{P}$ of $Aux(G)$.
(3) As vertices in $Aux(G)$ correspond to pair-wise disjoint subsets of vertices in G, for every partition class from $\mathcal{P}$ add the union of the subsets of vertices from G that are contained in this class to $\mathcal{Q}$.

Clusters obtained by ClusteringUDG have additional properties which are very useful in our analysis. These attributes, however, must be stated in terms of clusters in $Aux(G)$ rather than clusters in G. In particular, it is not true that the number of edges connecting different clusters in G is "small" with respect to the total number of edges but the property holds in $Aux(G)$. Intuitively, what we need from clusters in G is that an objective function to be approximated (for example the size of a connected dominating set) has a small value on the boundary of each cluster. At this moment let us indicate the running time of ClusteringUDG and we will use previous lemmas to extract useful properties when they are needed.

Lemma 9. *Let $G = (V, E)$ be a unit-disk graphs with identifiers bounded by $m = poly(|G|)$.* ClusteringUDG *runs in $O(1/\epsilon^6 \cdot \log^3 |G| \log 1/\epsilon)$ rounds.*

Proof. Kuhn et. al. algorithm runs in $O(\log |G| \log^* |G|)$ rounds. The complexity of Clustering is $O(1/\epsilon^6 \cdot \log^3 |G| \log 1/\epsilon)$ in $Aux(G)$. Since every vertex in $Aux(G)$ corresponds to a subgraph of diameter which is less than or equal to two, the running time in G will be asymptotically the same.

3 Applications

In this section, we will describe applications to the MM Problem and the MCDS Problem. Approximations for both problems (as well as to the maximum independent set and the minimum dominating set) follow a similar pattern: First find a clustering using ClusteringUDG and then find an optimal solution locally in each cluster. Finally, modify local solutions and return the union of them. Since our main clustering algorithm is executed in the auxiliary graph which arises from yet another clustering of G obtained from a maximal independent set, we will have three types of clusters.

- *Small clusters*: Clusters in graph G which arise from the maximal independent set obtained in phase one and correspond to vertices in $Aux(G)$.
- *Auxiliary clusters*: Clusters in $Aux(G)$ obtained by Clustering.
- *Big clusters*: Clusters in G obtained by ClusteringUDG. These clusters correspond in an obvious way to clusters in $Aux(G)$.

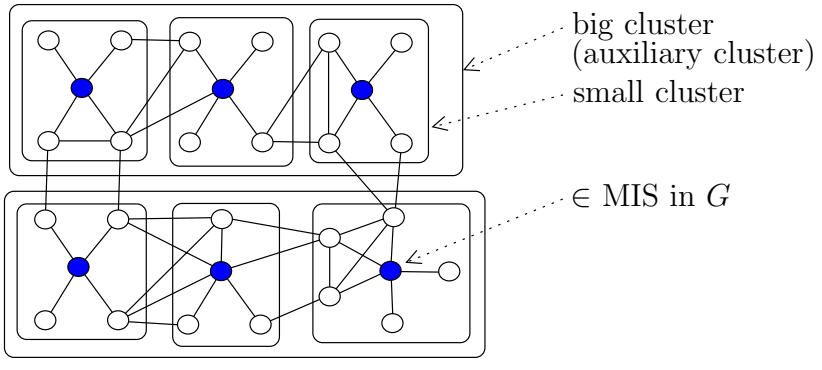

Fig. 1. Clusters in UDG

Analysis of algorithms relies on properties of $Aux(G)$ and it will be useful to develop more terminology related to clusters of $Aux(G)$. Let v be a vertex in $Aux(G)$. If v is in cluster C of $Aux(G)$ but has a neighbor in a different cluster of $Aux(G)$ then v will be called a *border vertex* and the small cluster in G which corresponds to it will be called a *border cluster*. We note, that since $Aux(G)$ has a constant maximum degree, Lemma 6 implies that the number of border vertices in $Aux(G)$ is much smaller than $|Aux(G)|$.

3.1 Maximum Matching

In our first application we will approximate a maximum matching. Recall that if M is a matching in graph G then a vertex is called M-saturated if it is an endpoint of an edge from M. Otherwise it is called M-free.

ApproxMaxMatching
Input: Unit-disk graph $G = (V, E)$, $0 < \epsilon < 1$, $0 \leq \alpha < 1$.
Output: A matching in G.

(1) Call ClusteringUDG to find a clustering of G with constants as in Corollary 1.
(2) In each cluster C, find a matching M_C in the subgraph of G induced by C which is a $(1-\alpha)$-approximation of a maximum matching in $G[C]$.
(3) Return $M := \bigcup_C M_C$.

Note that the second step of the procedure takes $polylog(|G|)$ rounds as the diameter of each big cluster is $polylog(|G|)$.

Lemma 10. *Let G be a unit-disk graph. The matching M returned by* ApproxMaxMatching *satisfies*

$$|M| \geq (1 - \alpha - O(\epsilon))\beta,$$

where β is the size of a maximum matching in G.

Proof. With a matching N we can associate the function $I_N : V(G) \to \{0, 1\}$ defined as $I_N(v) = 1$ if v is N-saturated and $I_N(v) = 0$ otherwise. Clearly, we have $2|N| = \sum_{v \in V(G)} I_N(v)$.

Let M^* be a maximum matching in G. We will first obtain a matching $\bar{M}$ from M^* as follows. For every border cluster D in a big cluster C, delete all edges from M^* which have one endpoint in D and another in $V - C$. Extend the obtained matching to a maximal matching $\bar{M}$ with the property that all edges from $\bar{M}$ are contained in $G[C]$ for some big cluster C. We claim that for any big cluster C

$$\sum_{v \in C} I_{\bar{M}}(v) \geq \sum_{v \in C} I_{M^*}(v) - 5s_C, \tag{4}$$

where s_C is the number of border clusters in C. Indeed, a subgraph of G induced by a border cluster D has a maximum independent set with at most 5 vertices (this is true for a closed neighborhood of any vertex in a unit-disk graph) and so the number of $\bar{M}$-free vertices in D which are possibly M^*-saturated is at most 5.

Since matching M_C from step (2) is a $(1-\alpha)$-approximation of a maximum matching in $G[C]$, we have

$$\sum_{v \in C} I_M(v) \geq (1 - \alpha) \sum_{v \in C} I_{\bar{M}}(v)$$

and so

$$|M| \geq (1 - \alpha)|\bar{M}|.$$

Summing (4) over all C's yields

$$|\bar{M}| \geq |M^*| - 5s/2$$

where s is the total number of border clusters. We have $s = O(\epsilon|Aux(G)|)$ and $|M^*| = \Omega(|Aux(G)|)$ as $Aux(G)$ has a constant maximum degree. Therefore,

$$|M| \geq (1 - \alpha - O(\epsilon))|M^*|.$$

Theorem 2. *Let k be a positive integer, let $0 \leq \alpha < 1$, and let $T_\alpha(m)$ be the running time of a sequential $(1-\alpha)$-approximation algorithm for the maximum matching problem in a unit-disk graph of order m. There is a distributed algorithm which finds in a unit-disk graph G a matching M such that*

$$|M| \geq \left(1 - \alpha - O(1/\log^k |G|)\right) \beta$$

where β is the size of a maximum matching in G. The number of rounds of the algorithm is poly-logarithmic in $|G|$ and the sequential running time at each vertex of G is at most $T_\alpha(|G|)$.

Proof. Set $\epsilon = 1/\log^k |G|$ and apply Lemma 10 and Lemma 9.
Since it is possible to find a maximum matching in a polynomial time, we can obtain $\alpha = 0$ in Theorem 2 and have sequential running time which is polynomial in $|G|$.

3.2 Minimum Connected Dominating Set

Algorithm for the connected dominating set is surprisingly simple. It is the analysis that requires some additional work.

ApproxMinCDS
Input: A connected unit-disk graph $G = (V, E)$, $0 < \epsilon < 1$ and $0 \leq \alpha$.
Output: A dominating set in G.

(1) Call ClusteringUDG to find a clustering of G with constants as in Corollary 1.
(2) In every big cluster C find a connected dominating set D_C which is a $(1+\alpha)$-approximation of a minimum connected dominating set in $G[C]$.
(3) For every two big clusters C, C' if there is an edge in $Aux(G)$ connecting a border cluster from C with a border cluster from C' then connect a vertex from D_C and a vertex from $D_{C'}$ by a path of length less than or equal to three. In other words, let $D_{C,C'}$ be the set of at most four vertices on the path connecting C with C'.
(4) Return $D := \bigcup_C D_C \cup \bigcup_{\{C,C'\} \in E(Aux(G))} D_{C,C'}$.

It is easy to see that step three can be completed and that D is a dominating set which induces a connected subgraph of G. It is the fact that $|D|$ is close to the optimal that requires a proof. We will first observe that for any connected dominating set D there is a connected dominating set D' such that $D' \cap C$ induces a connected dominating set for any big cluster C and the sizes of D and D' do not differ much.

Lemma 11. *Let G be a connected unit-disk graph and let D be a connected dominating set in G. Then there is a dominating set D' with the following properties.*

1. *$D \subseteq D'$.*
2. *For any big cluster C, $C \cap D'$ induces a connected dominating set in $G[C]$.*
3. *$|D'| - |D| = O(s)$ where s is the number of border clusters.*

Proof. Let D be a connected dominating set. For every border cluster B there is a vertex v_B which dominates all of the vertices from B. Consider the set $D^* = D \cup \bigcup_B \{v_B\}$. For a big cluster C, let s_C be the number of border clusters contained in C. We first observe that the number of connected components in graph $G[D^* \cap C]$ is $O(s_C)$. Indeed, since $G[D^*]$ is connected, every connected component in $G[D^* \cap C]$ contains a vertex from a border cluster in C. Recall that for any border cluster B, $G[B]$ cannot have six independent vertices and so $G[D^* \cap B]$ has at most five connected components. Consequently, the number of connected components in $G[D^* \cap C]$ is $O(s_C)$. Next, consider graph $Conn(D^*, C)$ defined as follows. Vertices of $Conn(D^*, C)$ are connected components in $G[D^* \cap C]$ and we put an edge between w_1 and w_2 if there is a path of length at most three in G connecting a vertex from w_1 with a vertex from w_2. We claim that $Conn(D^*, C)$ is a connected graph. Indeed, assume that $Conn(D^*, C)$ is disconnected and let X_1, X_2 be two connected components in $Conn(D^*, C)$. Let Y_1 and Y_2 be the sets of vertices in $G[C]$ such that Y_i is the union of vertices in clusters from X_i. Let P be a shortest path in $G[C]$ connecting a vertex from Y_1 with a vertex form Y_2. The length of P is at least four. Let $p = u_1, u_2, u_3, u_4, \ldots, u_l$ with $u_1 \in Y_1$, $u_l \in Y_2$ and $l \geq 5$. Observe that u_3 cannot be contained in a border cluster as if $u_3 \in B$ then either $u_3 = v_B$ (the "center" of B) or u_3 is connected with v_B. In either case $v_B \in Y_1$ and there is a shorter path connecting Y_1 with Y_2. If $u_3 \notin B$ then u_3 must be dominated by a vertex v from $D \cap C$ and so there is a path of length at most 3 between u_1 and v. Thus $v \in Y_1$ and there is a shorter path connecting Y_1 with Y_2. Contradiction shows that $Conn(D^*, C)$ is a connected graph. As a result $Conn(D^*, C)$ contains a spanning tree in which an edge corresponds to path of length at most three. Fix one such spanning tree and let D' be the set of vertices in D^* in addition with vertices on each path corresponding to an edge in the spanning tree of $Conn(D^*, C)$. Clearly $D \subseteq D'$. In addition $G[D \cap C]$ is a connected graph. Finally, the number of vertices in $Conn(D^*, C)$ is $O(s_C)$ and so we add $O(s_C)$ vertices to D^* to obtain D'. Summing over all C's gives $|D'| - |D| = O(s)$.

Now the next lemma is very easy.

Lemma 12. *Let G be a connected unit-disk graph. The set D returned by* ApproxMinCDS *is a dominating set which induces a connected subgraph of G and such that*

$$|D| \leq (1 + \alpha + O(\epsilon))\gamma_c,$$

where γ_c is the size of a minimum dominating set in G.

Proof. Let D^* be a connected dominating set of size γ_c. By Lemma 11, there is a dominating set D' such that $D^* \subseteq D'$, $D' \cap C$ induces a connected dominating

set in $G[C]$ for each C, and $|\bar{D}|-|D^*| = O(s)$. If D is the dominating set returned by algorithm ApproxMinCDS then $D_C = D \cap C$ is a $(1+\alpha)$-approximation of a minimum connected dominating set in $G[C]$ and so if D_C^* is a minimum connected dominating set in $G[C]$ then $|D_C| \leq (1+\alpha)|D_C^*|$. Since $D' \cap C$ is an optimal set in $G[C]$, we have $|D_C^*| \leq |D' \cap C|$ and so

$$|D| = \sum_C |D_C| + O(s) \leq \sum_C (1+\alpha)|D_C^*| + O(s) \leq \sum_C (1+\alpha)|D' \cap C| + O(s)$$

$$= (1+\alpha)|D'| + O(s) = (1+\alpha)|D^*| + O(s) = (1+\alpha)\gamma_c + O(\epsilon|Aux(G)|).$$

As before $\gamma_c = \Omega(|Aux(G)|)$ and so

$$|D| = (1+\alpha+O(\epsilon))\gamma_c.$$

Theorem 3. *Let k be a positive integer, let $0 \leq \alpha$, and let $T_\alpha(m)$ be the running time of a sequential $(1+\alpha)$-approximation for the minimum connected dominating set problem in a unit-disk graph of order m. There is a distributed algorithm which finds in a connected unit-disk graph G a dominating set D such that $G[D]$ is connected and*

$$|D| \leq \left(1+\alpha+O(1/\log^k |G|)\right)\gamma_c$$

where γ_c is the size of the minimum connected dominating set in G. The number of rounds of the algorithm is poly-logarithmic in $|G|$ and the sequential running time at each vertex of G is at most $T_\alpha(|G|)$.

4 Additional Remarks

As pointed out by one of the referees, in addition to Theorem 3, the methods described in the paper yield a sequential PTAS for the MCDS Problem in unit-disk graphs when the representation of the graph is unknown. Indeed, for a fixed $\epsilon > 0$ from (1), the radius of each cluster obtained by Cluster Set (and so the Clustering) is $O(1/\epsilon^2)$ by Lemma 3. Since in a unit-disk graph of diameter r any indepdenet set has size of at most $O(r^2)$ and each independent set is also a dominating set in the graph, the size of the minimum dominating set is $O(1/\epsilon^4)$. As a result, the size of the minimum connected dominating set in each cluster is also $O(1/\epsilon^4)$. Consequently, to find an optimal solution in each cluster it is enough to check $O(n^{1/\epsilon^4})$ subsets in a cluster and select an optimal. This extends ther result of Nieberg and Hurink from [NH05] where a PTAS for the MDS Problem is given.

References

[AGLP89] B. Awerbuch, A. V. Goldberg, M. Luby, S. A. Plotkin, Network Decomposition and Locality in Distributed Computation, Proc. 30th IEEE Symp. on Foundations of Computer Science , (1989), 364–369.

[CHLWD03] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z. Du, Polynomial-time approximation scheme for minimum connected dominating set in ad hoc wireless networks, Volume 42, Issue 4, Networks, (2003), 202–208.

[CH06] A. Czygrinow, M. Hańćkowiak, Distributed algorithms for weighted problems in sparse graphs, Journal of Discrete Algorithms, in press, (2006).

[CHS06] A. Czygrinow, M. Hańćkowiak, E. Szymańska, Distributed approximation algorithms for planar graphs, 6th Conference on Algorithms and Complexity, CIAC 2006, accepted, (2006).

[D97] R. Diestel, *Graph Theory*, Springer, New York, (1997).

[DW04] F. Dai, J. Wu, An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks, IEEE Transactions on Parallel and Distributed Systems, vol. 15, no. 10, (2004), 908–920.

[DPRS03] D. Dubhashi, A. Mei, A. Panconesi, J. Radhakrishnan, and A. Srinivasan, Fast Distributed Algorithms for (Weakly) Connected Dominating Sets and Linear-Size Skeletons, In Proc. of the ACM-SIAM Symposium on Discrete Algorithms (SODA), (2003), 717–724.

[GGHZZ01] 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), (2001), 188–196.

[KMW05] F. Kuhn, T. Moscibroda, and R. Wattenhofer, On the Locality of Bounded Growth, 24th ACM Symposium on the Principles of Distributed Computing (PODC), Las Vegas, Nevada, USA, (2005), 60–68.

[KMNW05a] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer, Fast Deterministic Distributed Maximal Independent Set Computation on Growth-Bounded Graphs, 19th International Symposium on Distributed Computing (DISC), Cracow, Poland, September (2005), 273–287.

[KMNW05b] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer, Local Approximation Schemes for Ad Hoc and Sensor Networks, 3rd ACM Joint Workshop on Foundations of Mobile Computing (DIALM-POMC), Cologne, Germany, (2005), 97–103.

[L82] D. Lichtenstein, Planar Formulae and Their Uses, SIAM Journal of Computing, 11(2), (1998), 329–343.

[L92] N. Linial, Locality in distributed graph algorithms, SIAM Journal on Computing, 21(1), (1992), 193–201.

[NH05] T. Nieberg, J. L. Hurink, A PTAS for the Minimum Dominating Set Problem in Unit Disk Graphs, 3rd Workshop on Approximation and Online Algorithms, WAOA 2005, Mallorca, Spain, October 2005, Springer LNCS 3879, (2005), 296–306.

[PR01] A. Panconesi, R. Rizzi, Some Simple Distributed Algorithms for Sparse Networks, Distributed Computing 14, (2001), 97–100.

[P00] D. Peleg, *Distributed Computing: A Locality-Sensitive Approach*, SIAM, 2000.