LOCAL MAXIMAL MATCHING AND LOCAL 2-APPROXIMATION FOR VERTEX COVER IN UDGS

ANDREAS WIESE*,§ AND EVANGELOS KRANAKIS**,§§

Abstract. We present $1-\epsilon$ approximation algorithms for the maximum matching problem in location aware unit disc graphs and in growth-bounded graphs. The algorithm for unit disk graph is local in the sense that whether or not an edge is in the matching depends only on other vertices which are at most a constant number of hops away from it. The algorithm for growth-bounded graphs needs at most $O\left(\log \triangle \log^* n + \frac{1}{\epsilon}^{O(1)} \cdot \log^* n\right)$ communication rounds during its execution. Using these matching algorithms we can compute vertex covers of the respective graph classes whose size are at most twice the optimal.

1. Introduction

Unit Disk Graphs (UDGs) is a widely used concept for modeling ad hoc and wireless networks. In these graphs, the connectivity of two nodes is established if and only if their Euclidean distance of these two nodes is at most one. Therefore, UDGs model the setting of identical wireless devices on a plane without obstacles that could obscure the wireless signals. There are also other models for wireless networks, e.g., Quasi-Unit-Disk-Graphs (Q-UDGs) which were first introduced by Barriere et al. [2]. In Q-UDGs there is a certain radius ℓ such that two nodes which are closer to each other than ℓ are always connected whereas nodes with a larger distance than one unit are always disconnected. This and other models for wireless networks are captured by growth-bounded graphs. These are graphs in which for any vertex v the size of an independent set of the vertices which are at most r hops away from v is at most $f(r)$ (for a certain growth-function f).

In the setting of wireless and ad-hoc-networks there is usually no global communication backbone available. So for organizing the network traffic and solving problems like matching or vertex cover we need to find a method that does not rely on global information of the network. So we are interested in local algorithms. These are algorithms for which the result of a computation for a vertex or an edge depends only on the vertices and edges which are at most a certain distance away from them (the locality distance). With this constraint we ensure that we do not need knowledge of the entire network but only information about the network in

Research conducted while the authors were visiting the School of Computing Science at Simon Fraser University, Vancouver.

* Technische Universität Berlin, Institut für Mathematik, Germany.

§ Research supported by a scholarship from DAAD (German Academic Exchange Service).

** School of Computer Science, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6.

§§ Research supported in part by NSERC (Natural Science and Engineering Research Council of Canada). Research supported in part by MITACS (Mathematics of Information Technology and Complex Systems).

a certain neighborhood of a vertex or an edge. It is also of interest in dynamically changing networks since if only small changes occur local algorithms need to recompute only small parts of the solution.

In our UDG graph model we assume that every node is aware of its geographic position in the plane. Allowing this positional knowledge we will see that the locality distance of our algorithms can be bounded by a constant. Note that this constant does not depend on the overall size of the network or the maximal vertex degree. Since positioning systems like GPS become more and more common, this setting seems to be relevant.

For organizing communication in wireless networks matching is a useful concept. In one communication round a node can usually receive data from only sender (due to interference) and each sender can send only one package at a time (usually to one receiver). Thus the sender/receiver pairs form a matching in the underlying network graph. Research has been done on finding matchings with certain properties [3] in order to deal with interference and noise issues. Also, in the computation of schedules for allocating bandwidth the matching problem can arise [12].

1.1. **Related Work.**

The matching problem is in P for general graphs [5]. The first algorithm due to Edmonds requires a runtime of $O\left(n^3\right)$. For the restricted case of bipartite graphs there are improvements known, e.g., the Hopcroft-Karp algorithm [7].

The vertex cover problem is NP-hard in general graphs [6], but there are several polynomial time approximation algorithms which guarantee an approximation factor of 2, e.g., in [1]. However, it is NP-hard to approximate the problem with a factor better than $10\sqrt{5} - 21 \approx 1,3607$ [4]. Thus there can be no polynomial time approximation scheme (PTAS), unless $P = NP$. When we restrict the setting to unit disk graphs, vertex cover remains NP-hard. The same holds for growth-bounded graphs since this class includes unit graphs. However, for unit disk graphs PTASs are known. For the case where the embedding of the graph is known, Hunt III et al. [8] presented the first approximation scheme. The algorithm for independent set presented in [10] together with the technique in [14] yields a global PTAS for vertex cover that does not rely on the embedding of the graph. There is also a local PTAS known for the setting of location aware UDGs [14].

1.2. **Our Results.**

We present the first local approximation algorithms for matching in location aware Unit Disk Graphs. It achieves an approximation ratio of $1-\epsilon$ for arbitrarily small ϵ. In this setting we can show that the locality distance of our algorithm (i.e. the radius of the area that needs to be explored in order to compute the status of one edge) is bounded by a constant. In particular, this constant does not depend on the size of the entire network or the maximal degree of a vertex. For growth-bounded graphs we also give a $1-\epsilon$ approximation algorithm. For this setting we lift the assumption of positional information in the nodes and require only a unique ID in each vertex. The locality distance for this algorithm is in $O\left(\log \triangle \log^* n + \frac{1}{\epsilon}^{O(1)} \log^* n\right)$. All matchings computed by these algorithms are maximal.

Each matching algorithm yields a local approximation algorithm for the vertex cover problem. The locality properties of these algorithms are identical to the respective matching algorithms. The size of the computed vertex covers are at most twice the size of an optimal vertex cover. As mentioned above, for location aware unit disk graphs there is a local PTAS known [14]. However, the locality distance of this PTAS when executed with approximation factor 2 is a lot larger than the locality of Algorithm 3.

1.3. **Organization of the Paper.** In Section 2 we present our local $1-\epsilon$ approximation algorithm for matching in location aware unit disk graphs. In Section 3 we show how the ideas of this algorithm can be used in order to derive a local $1-\epsilon$ approximation algorithm for the same problem in growth-bounded graphs. Our local approximation algorithms for vertex cover with approximation factor 2 are presented in Section 4. Finally in Section 5 we summarize our results and address open problems.

2. Maximum matching in Location Aware UDGs

In this section we present a local $1-\epsilon$ approximation algorithm for the maximum matching problem in location aware unit disk graphs. First we give some basic definitions. Then we define a tiling of the plane that we are going to use in our algorithm. Finally we present the algorithm and prove its correctness.

2.1. **Definitions.** The graph $G = (V, E)$ considered in this section is a unit disk graph. For two vertices u and v let $d(u, v)$ be the hop-distance between u and v, that is the number of edges on a shortest path between these two vertices. Note that the hop-distance between two vertices does not necessarily equal the geometric distance between them. Denote by $N^r(v) = \{u \in V \mid d(u, v) \leq r\}$ the r-th neighborhood of a vertex v. In Section 3 we will consider growth-bounded graphs.

Definition 1. An undirected graph $G = (V, E)$ is called a *growth-bounded graph* if there exists a polynomial bounding function $f(r)$ such that for every $v \in V$ and $r \geq 0$, the size of the largest independent set in the r-neighborhood $N^r(v)$ is at most $f(r)$.

This is the definition that I copied from a paper. Actually, what is really meant is something like this:

Definition 2. Let $\mathcal{G}$ be a family of graphs. We call $\mathcal{G}$ *growth-bounded*, if there exists a polynomial bounding function $f(r)$ such that for every graph $G \in \mathcal{G}$, every vertex v in G and every $r \geq 0$, the size of the largest independent set in the r-neighborhood $N^r(v)$ in G is at most $f(r)$.

In Section 3 we will consider growth-bounded graphs. We will implicitly assume that the family $\mathcal{G}$ and its bounding function $f(r)$ are known and fixed.

What do you think?

Let $M \subseteq E$ be a set of edges. We call M a *matching*, if no two edges in M share an end-vertex. We call M a *maximum matching*, if for all matchings M' it holds that $|M'| \leq |M|$. A *maximal matching* is a matching which cannot be extended by adding another edge.

Let M be a matching. We call a path p an *M-alternating path*, if it contains alternating matching- and non-matching edges. We call a vertex v an *isolated* vertex, is it is not adjacent to an edge from M. We call an M-alternating path p an *M-augmenting path,* if it starts from and ends on isolated vertices. Note: An M-augmenting path has an odd number of edges.

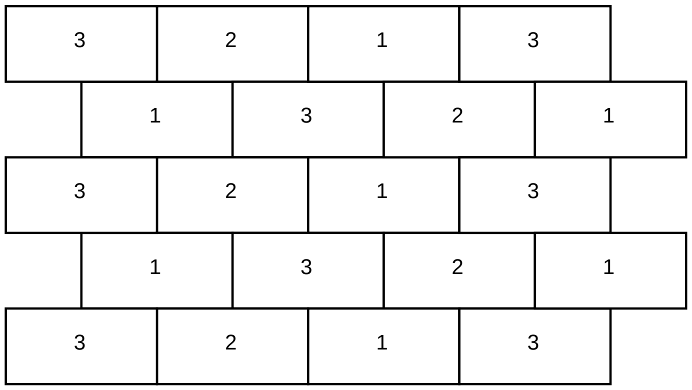

FIGURE 1. The tiling of the plane.

Lemma 1. A matching M is a maximum matching, if and only if there is no M-augmenting path.

2.2. **Tiling of the Plane.** Let $1 - \epsilon$ be the desired approximation ratio for the matching algorithm. We define k to be the smallest integer such that $\epsilon \geq \frac{2}{k+1}$. We tile the plane with an infinitely repeated pattern of rectangles as seen in Figure 1. Each rectangle is assigned class number 1, 2 or 3. The height of each rectangle is $2k + 2$, the width of each rectangle is $4k + 4$.

2.3. **The Algorithm.** Now we present the algorithm. It has three phases:

(1) For each rectangle R we compute a matching that includes only edges that have both end-vertices in R.
(2) For each class 1 rectangle R we check if there are augmenting paths in the subgraph induced by the vertices which are at most k hops away from R. If there are such paths, we augmented the matching until no such paths are left.
(3) For each class 2 rectangle R we check if there are augmenting paths in the subgraph induced by the vertices in R and the vertices in class 3 rectangles which are at most k hops away from R. As in the step before, we augment the matching along all these paths.

Now we present the algorithm in detail. For all rectangles R we do the following: Denote by V_R the vertices in R. For the subgraph induced by V_R we compute a maximum matching using a standard matching algorithm. Since all V_R for the different rectangles R are disjoint the order in which we do this does not matter. Now we come to phase 2: For each class 1 rectangle R we take the set of vertices which are in R or at most k hops away from a vertex in R. Denote this set by V'_R. In the subgraph induced by V'_R we augment the matching along all augmenting paths. Since the height of the rectangles is $2k + 2$ and their width is $4k + 4$, the order in which the class 1 rectangles are being processed does not matter. Finally we start phase 3: For each class 2 rectangle R we compute all vertices which are at most k hops away from R. Denote this set by V''_R. For the subgraph induced by V''_R we we augment the matching along all augmenting paths. Denote by M the resulting matching. We refer to the above as Algorithm 1.

Algorithm 1: Algorithm for finding a matching in a unit disk graph $G = (V, E)$

```
// Phase 1;
foreach rectangle $R$ do
    // denote by $V_R$ the vertices in $R$;
    determine a maximum matching $M_R$ for the subgraph induced by $V_R$;
end
Define $M := \bigcup_{R \in T} M_R$;
// Phase 2;
foreach rectangle $R$ with class$(R) = 1$ do
    Denote by $V_R$ all vertices in $R$;
    Explore all vertices which are at most $k$ hops away from vertices in $V_R$;
    // Denote these vertices by $V'_R$;
    Augment $M$ along augmenting paths in the subgraph induced by $V'_R$;
end
// Phase 3;
foreach rectangle $R$ with class$(R) = 2$ do
    Denote by $V_R$ all vertices in $R$;
    Explore all vertices which are at most $k$ hops away from vertices in $V_R$;
    // Denote these vertices by $V''_R$;
    Augment $M$ along augmenting paths in the subgraph induced by $V''_R$;
end
```

In the following theorem we prove that Algorithm 1 is a local algorithm that computes a valid matching with a competitive ratio of $1 - \epsilon$.

Theorem 1. *Algorithm 1 has the following properties:*

(1) *The computed matching M is a maximal matching for G.*
(2) *Let M_{OPT} be an optimal matching for G. It holds that $|M| \geq (1 - \epsilon) \cdot |M_{OPT}|$.*
(3) *Whether or not an edge $e = (u, v)$ is a matching edge depends only on the vertices which are at most $O\left(1/\epsilon^2\right)$ hops away from u or v, i.e. Algorithm 1 is local.*
(4) *The processing time for an edge $e = (u, v)$ is bounded by a cubic polynomial in the number of vertices which are at most $O\left(1/\epsilon^2\right)$ hops away from u or v.*

2.4. **Proof of Correctness.** We will prove the four parts of this theorem in four steps.

2.4.1. *Validity and Maximality.* We prove that M is a valid matching for G and that it is maximal.

Proof. (of part 1 of Theorem 1): The matchings constructed in phase 1 are clearly valid. Since in phase 2 and 3, M is only augmented along augmenting paths, the resulting matching is valid as well.

Now we want to prove that M is maximal. We call an edge that would extend M an *extending edge*. Since we augment the matching along augmenting paths a vertex which is adjacent to a matching edge once will be adjacent to a matching edge in the final matching as well. We see that after phase 1 all extending edges must have their adjacent vertices in different rectangles since we compute maximum matchings for each rectangle. From the construction of the tiling we see that these rectangles must have different class number (since the length of an edge is at most 1). After phase 2 there are no extending edges between class 1 and 2 rectangles left since the matching would be augmented along such "paths". With the same reasoning we see that after phase 3 there are no extending edges between class 2 and 3 rectangles left. So for the final matching there are no extending edges in the graph. This implies that the matching M is maximal. □

2.4.2. *Approximation Ratio.* Let M_{OPT} be an optimal matching for G. We prove that $|M| \geq (1-\epsilon) \cdot |M_{OPT}|$.

Proof. (of part 2 of Theorem 1): Denote by M_i the matching computed by the algorithm after phase i for $i \in \{1,2,3\}$. Let P_i be the set of augmenting paths for M_i with $i \in \{1,2,3\}$. From the construction of M_1 it follows that all paths in P_1 must have their start- and endvertices in two different rectangles. From the algorithm we see that all paths in P_2 either

- do not have their start- and endvertex in a rectangle of class 1 or
- are longer than k,

since all other augmenting paths in P_1 are eliminated in phase 2. Similarly, in P_3 all augmenting paths which are left are longer than k edges.

Consider $M' := M_3 \triangle M_{OPT} = (M_3 - M_{OPT}) \cup (M_{OPT} - M_3)$ and $G' := (V, M')$. All nodes in G' have a degree of at most two (since M_3 and M_{OPT} are both matchings). Its connected components are

- isolated vertices
- cycles of even length
- paths of three possible types
  - paths starting and ending with an edge from M. This cannot happen since this would be an augmenting path for M_{OPT} and M_{OPT} is optimal.
  - paths starting with an edge from M and ending with an edge from M_{OPT}. These paths have the same number of edges from M as from M_{OPT}.
  - paths starting and ending with an edge from M_{OPT}. These are augmenting paths for M. Denote all these paths by P_3'.

Every augmentation would increase the number of edges in M by one, so $|M| + |P_3'| = |M_{OPT}|$. Since $P_3' \subseteq P_3$ all paths in P_3' have more than k edges. So every path in P_3' contains at least $\frac{k+1}{2}$ edges of M_{OPT}. Since the paths are disjoint, it follows that $|P_3'| \leq |M_{OPT}| / \frac{k+1}{2}$. We then have

$$\begin{aligned} |M| &= |M_{OPT}| - |P_3'| \\ &\geq |M_{OPT}| - \frac{2\,|M_{OPT}|}{k+1} \\ &= \left(1 - \frac{2}{k+1}\right) \cdot |M_{OPT}| \\ &\geq (1-\epsilon)\,|M_{OPT}| \end{aligned}$$

□

2.4.3. *Locality.* We prove that whether or not an edge $e = (u, v)$ belongs to M depends only on the vertices which are at most $O\left(1/\epsilon^2\right)$ hops away from u or v. First we need to give a technical lemma.

Lemma 2. *Let R be a rectangle and $G[R]$ the graph G restricted to R. For each connected component C in $G[R]$ it holds that $diam(C) \leq 22k^2 + 58k + 39$. Let R' be a rectangle and $G[R']$ the graph G restricted to the vertices which are at most k hops away from R' (including the vertices in R' itself). Then for each connected component C' in $G[R']$ it holds that $diam(C') \leq 30k^2 + 70k + 31$.*

Proof. First we derive an upper bound for the maximum size of an independent set in $G[R]$. The area of R plus a surrounding belt of width 1/2 around it is $(2k+3)\cdot(4k+5) = \left(8k^2 + 22k + 15\right)$. So there can be at most $\left\lfloor \frac{8k^2+22k+15}{\pi/4} \right\rfloor$ centers of non-overlapping discs of radius 1/2 in R. We compute that $\left\lfloor \frac{8k^2+22k+15}{\pi/4} \right\rfloor \leq \left\lfloor \frac{32k^2}{\pi} \right\rfloor + \left\lfloor \frac{88k}{\pi} \right\rfloor + \left\lfloor \frac{60}{\pi} \right\rfloor \leq 11k^2+29k+20$. It follows that the cardinality of a maximum independent set in $G[R]$ is at most $11k^2 + 29k + 20$. Now consider a connected component C in $G[R]$ and two vertices $u, v \in C$ such that $d(u, v) = diam(C)$. Denote by p the shortest path between u and v in C. If we take every alternating vertex in p we get an independent set in R. As the size of such a set is bounded by $11k^2 + 29k + 20$, the length of p is bounded by $22k^2 + 58k + 39$ and therefore $diam(C) \leq 22k^2 + 58k + 39$.

Applying the same reasoning to R' we derive an upper bound of $15k^2+35k+16$ for an independent set in $G[R']$ (since $\left\lfloor \frac{(2k+3+k)\cdot(4k+4+k)}{\pi/4} \right\rfloor = \left\lfloor \frac{15k^2+27k+12}{\pi/4} \right\rfloor \leq \left\lfloor \frac{60k^2}{\pi} \right\rfloor + \left\lfloor \frac{108k}{\pi} \right\rfloor + \left\lfloor \frac{48}{\pi} \right\rfloor = 15k^2 + 35k + 16$) and therefore we get $diam(C') \leq 30k^2 + 70k + 31$ for any connected component C' in $G[R']$. □

Proof. (of part 3 of Theorem 1): Denote by a_i the maximum number of hops which we need to explore around u and v in order to compute whether $e \in M$ after phase i (for $i \in \{1, 2, 3\}$).

In order to determine the status of an edge e after phase 1, we need to explore only the connected component of e in its rectangle if u and v are in the same rectangles or nothing if u and v are in different rectangles. From Lemma 2 it follows that $a_1 \leq 22k^2 + 58k + 39$. For computing the status of e after phase 2 we need to explore the connected component V'_R with $u \in V'_R$ and $v \in V'_R$ (if it exists) and what edges in V'_R were assigned to M after phase 1. It follows that $a_2 \leq a_1 + 30k^2 + 70k + 31$ (see Lemma 2). Analogously for computing the status of e after phase 3 we need to explore the connected component V''_R such that $u \in V''_R$ and $v \in V''_R$ (if such a component exists) and what edges in V'_R were assigned to M after phase 2. This implies that $a_3 \leq a_2 + 30k^2 + 70k + 31$. So altogether we get that $a_3 \leq 22k^2+58k+39+30k^2+70k+31+30k^2+70k+31 = 82k^2+198k+101 \in O\left(k^2\right)$.

By definition k is the smallest integer such that $\epsilon \geq \frac{2}{k+1}$. This implies that $k \geq \frac{2}{\epsilon} - 1$ and thus $k \in O\left(1/\epsilon\right)$. It follows that $a_3 \in O\left(1/\epsilon^2\right)$. □

2.4.4. *Processing time.* We want to show that the processing time of Algorithm 1 for a single edge e is in $O\left(\bar{n}(e)^3\right)$ where $\bar{n}(e)$ is the number of vertices within the locality distance of e (i.e. the number of vertices which we really need to explore in order to compute the status of e).

Proof. (of part 4 of Theorem 1): In phase 1 we need to compute a maximum matching for edges in a single rectangle. This can be done in $O\left(\bar{n}(e)^3\right)$ using any algorithm for computing a maximum matching (e.g. Edmonds algorithm [5]). In phase 2 we need to find augmenting paths in the subgraph induced by V_R' for several class 1 rectangles R. Since in the locality distance of e there can be only a constant number of class 1 rectangles this requires $O\left(\bar{n}(e)^3\right)$ time (note that the number of such class 1 rectangles does not depend on the desired approximation ratio). Applying the same reasoning in phase 3 we need a processing time of $O\left(\bar{n}(e)^3\right)$. This leads to an overall processing time of $O\left(\bar{n}(e)^3\right)$. □

3. Maximum Matching Without Location Awareness

In this section we present a local algorithm which computes a $1-\epsilon$ approximation for the maximum matching problem in growth-bounded graphs. In contrast to the algorithm presented in Section 2 we assume a graph model in which the embedding of the graph is unknown. We will specify this in the following section.

3.1. **Graph Model.** Let $G = (V, E)$ be a growth-bounded graph. We assume that every node has a unique identifier (ID). Apart from that there is no information available to distinguish the nodes from each other.

3.2. **The Algorithm.** Let $1-\epsilon$ be the desired approximation ratio. The algorithm uses the same methodology as Algorithm 1 for ensuring the approximation ratio: We will compute a maximal matching M such that the length of each of the augmenting paths that could turn M into a maximum matching is at least a certain constant k. At the beginning of the algorithm we choose k according to ϵ.

The role of the rectangle classes in the algorithm above will be taken by a maximal independent set which is also a dominating set. In order to organize the computation distributively we use the same methods which were originally presented in [10].

Similarly as in Algorithm 1 we define k to be the smallest even integer such that $\epsilon \geq \frac{2}{k+1}$. We compute a maximal independent set I in G. This can be done locally using the distributed algorithm [5]. Then we define the clustergraph $\bar{G} = (\bar{V}, \bar{E})$ with radius $2k+2$ by $\bar{V} := I$ and

$$(u, v) \in \bar{E} \Leftrightarrow d_G(u, v) \leq 2k + 2$$

Since G is a growth-bounded graph, the maximum degree $\triangle_{\bar{G}}$ of $\bar{G}$ is bounded by a constant. This allows us to use the algorithm in [11] for coloring the vertices of $\bar{G}$ with at most $O\left(\triangle_{\bar{G}}^2\right)$ colors. We initialize our matching M with $M := \emptyset$. Then we iterate over the different colors of $\bar{G}$. For each color c we do the following: For each vertex v_c which was colored with color c we compute the subgraph induced by $N^{k+1}(v_c)$. Denote by $G_c(v_c)$ such a subgraph around a vertex v_c. From the definition of $\bar{G}$ we see that the subgraphs are all disjoint. In each subgraph $G_c(v_c)$ we augment our matching M along augmenting paths until we cannot find any more augmenting paths. This can be done using a standard matching algorithm, e.g., the algorithm by Edmonds [5]. Since the subgraphs are disjoint this can be done distributively. After having iterated over all colors, we output M. We refer to this as Algorithm 2.

Algorithm 2: Algorithm for finding a matching in a unit disk graph $G = (V, E)$

```
// Let $1 - \epsilon$ be the desired approximation ratio;
Define $k$ to be the smallest integer such that $\frac{k+1}{k+3} \geq 1 - \epsilon$;
Compute a maximal independent set $I$ for $G$;
Construct cluster graph $\bar{G}$ with radius $2k + 2$;
Color $\bar{G}$ with $\gamma = O\left(\triangle_{\bar{G}}^2\right)$ colors;
$M := \emptyset$;
for $i := 1$ to $\gamma$ do
    foreach vertex $v_c$ with color $c$ do do
        compute subgraph $N^{k+1}(v_c)$;
        augment $M$ along augmenting in $N^{k+1}(v_c)$;
    end
end
```

Theorem 2. *Algorithm 1 has the following properties:*

(1) *The computed matching M is a maximal matching for G.*
(2) *Let M_{OPT} be an optimal matching for G. It holds that $|M| \geq (1 - \epsilon) \cdot |M_{OPT}|$.*
(3) *The algorithm requires at most $O\left(\log \triangle \log^* n + \frac{1}{\epsilon}^{O(1)} \cdot \log^* n\right)$ communication rounds.*

3.3. **Proof of Correctness.** We will prove the four parts of this theorem in four steps.

3.3.1. *Validity and Maximality.* We want to prove that M is a matching and that it is maximal.

Proof. (of part 1 of Theorem 2): For the correctness of the subroutines for computing the maximal independent set and the vertex coloring we refer to their respective articles [9, 11]. In each iteration the matching is augmented along augmenting paths. This clearly constructs a valid matching. Now we want to prove that M is maximal. Assume on the contrary that there is an edge $e = (u, v)$ with $e \notin M$ but such that $M \cup \{e\}$ is a valid matching. Since I is a maximal independent set it is also a dominating set. So there is a vertex $u' \in I$ which is adjacent to u. Let c be the color of u'. There is an iteration in which u' was considered. Since we always augment our matching along augmenting paths, both u and v were unmatched in this iteration (in $G_c(u)$). Since e is in $G_c(u)$ and we augment M along all augmenting paths in $G_c(u)$, the edge e is added to M. In all future iterations u and v will always be matched (adjacent to a matching edge). This is contradiction. □

3.3.2. *Approximation Ratio.* We want to prove that for a maximum matching M_{OPT} for G it holds that $|M| \geq (1 - \epsilon) \cdot |M_{OPT}|$.

Proof. (of part 2 of Theorem 2): Like in the proof of Theorem 1 we show that there are no augmenting paths for M whose length is shorter or equal to k. Denote by $I_i \subseteq I$ all vertices in I which were colored with color i. Denote by P_i all vertices which are either in I_i or adjacent to a vertex in I_i and denote by M_i the computed matching after the ith iteration. In the ith iteration of the algorithm we check for augmenting paths in the subgraphs $G_c(v)$ (for each $v \in I_i$). Thus after the ith iteration there are no more augmenting paths which start with an isolated vertex in P_i and whose length is at most k.

Now consider $M_i' := M_i \triangle M_{OPT}$. The edges in M_i' form either circles of even length or augmenting paths. When we compare M_i' with M_j' for $j > i$ we see that in M_j the paths from M_i are either unchanged, eliminated (because we augmented the matching along them), or two paths are connected (because we augmented along a path that connected these two paths). In both cases it still holds that all augmenting paths starting with an isolated vertex in P_i are longer than k edges.

Since I is a dominating set for G it holds that $\bigcup P_i = V$. Thus after all iterations there are no augmenting paths left which have at most k edges. So with the same argumentation as in part 2 of Theorem 1 we can show that $|M| \geq (1-\epsilon) \cdot |M_{OPT}|$. □

3.3.3. *Locality.* We show that we need at most $O\left(\log \triangle \log^* n + \frac{1}{\epsilon}^{O(1)} \cdot \log^* n\right)$ communication rounds.

Proof. (of part 3 of Theorem 2): Computing the maximal independent set I can be done in $O(\log \triangle \log^* n)$ communication rounds [9]. The coloring of the cluster graph takes $O(k \cdot \log^* n)$ rounds [11]. The computation of the matchings needs $O\left(k \cdot \triangle_{\bar{G}}^2\right)$ communication rounds since we have $O\left(\triangle_{\bar{G}}^2\right)$ different colors and we explore the vertices which are at most $k+1$ hops away from each vertex $v \in I$. The maximum degree of the cluster graph $\bar{G}$ is bounded by $O(f(2k+2))$ where $f(n)$ is the growth-bounding-function of G. By definition k is the smallest integer such that $\epsilon \geq \frac{2}{k+1}$. This implies that $k \geq \frac{2}{\epsilon} - 1$ and thus $k \in O(1/\epsilon)$.

Altogether this implies that Algorithm 2 needs at most $O\left(T_{MIS} + \frac{1}{\epsilon}\left(\log^* n + f\left(\frac{2}{\epsilon}+2\right)^2\right)\right)$ communication rounds where T_{MIS} are the communication rounds needed for computing a maximal independent set. Using the algorithm in [9] for this task we need $O\left(\log \triangle \log^* n + \frac{1}{\epsilon}^{O(1)} \cdot \log^* n\right)$ communication rounds in total. □

4. Vertex Cover

In this section we present local approximation algorithms for the minimum vertex cover problem. We use the local matching algorithms presented in Sections 2 and 3 respectively as subroutines. First we compute a maximum matching. Then we assign all vertices which are adjacent to matched edges to the vertex cover. Using a well-known reasoning we prove that this gives a factor 2 approximation for vertex cover.

4.1. **The Algorithm.**

Let $G = (V, E)$ be a unit disk graph. First we use Algorithm 1 or Algorithm 2 in order to compute a maximal matching M. We modify the algorithm as follows: Since we are not interested in a good approximation for the matching problem we choose $k := 1$. In order to improve the runtime of the algorithm, we consider only augmenting paths of length 1 in each phase (this is effectively a greedy-algorithm for the matching problem). Then we define our vertex cover VC as follows: $VC := \{u, v | (u, v) \in M\}$.

Remark 1. Using Algorithms 1 and 2 we cannot only compute maximal matchings, but also maximal matchings which are not much smaller that maximum matchings. However, for this algorithm, we could not prove a better performance ratio if we computed a matching with a certain performance guarantee. So in order to achieve a small locality distance we just compute a maximal matching.

Algorithm 3: Algorithm for finding a vertex cover in a unit disk graph $G = (V, E)$

```
Define $k := 1$;
Compute a maximum matching $M$ using Algorithm 1 or Algorithm 2 and
only augmenting along paths of length 1;
Define $VC := \{u, v| (u, v) \in M\}$;
Output $VC$;
```

Depending an which algorithm we use for computing the matching we get a different algorithm for vertex cover. Theorem 3 represents the algorithm that we get by using Algorithm 1, Theorem 4 the algorithm which is the result of using Algorithm 2 as a subroutine.

Theorem 3. *There is an algorithm for location aware unit disk graphs which computes a set VC with the following properties:*

(1) *The computed set VC is a vertex cover for G.*
(2) *Let VC_{OPT} be an optimal vertex cover for G. It holds that $|VC| \leq 2 \cdot |VC_{OPT}|$.*
(3) *If a vertex v is in VC depends only on the vertices which are at most 381 hops away from v, i.e. Algorithm 3 is local.*
(4) *The processing time for a vertex v is bounded by a linear polynomial in the number of edges whose adjacent vertices are both at most 381 hops away from v.*

Theorem 4. *There is an algorithm for growth-bounded graphs with unique vertex-IDs which computes a set VC with the following properties:*

(1) *The computed set VC is a vertex cover for G.*
(2) *Let VC_{OPT} be an optimal vertex cover for G. It holds that $|VC| \leq 2 \cdot |VC_{OPT}|$.*
(3) *The algorithm requires at most $O\left(\log \triangle \log^* n + \frac{1}{\epsilon}^{O(1)} \log^* n\right)$ communication rounds.*

4.2. **Proof of Correctness.** Here we only prove Theorem 3. The proof of Theorem 4 can be done similarly.

Proof. (of Theorem 3): From Theorem 1 we know that M is a maximal matching. Now let $e = (u, v)$ be an edge. If $e \in M$ then $u \in VC$ and $v \in VC$ and therefore e is covered. If $e \notin M$ then either u or v is adjacent to a matching edge (since M is maximal). Thus either $u \in VC$ or $v \in VC$.

The cardinality of any matching in a graph forms a lower bound for the cardinality of a minimum vertex cover. This holds since every vertex of an optimal vertex cover can cover at most one edge of the matching. As we assign two vertices to VC for each edge in M we conclude that $|VC| \leq 2 \cdot |M| \leq 2 \cdot |VC_{OPT}|$.

For the locality distance we need to check only the locality distance of Algorithm 1 for $k = 1$. From the proof of Theorem 1 we conclude that this is 381.

Now we want to show the processing time for Algorithm 3. In the part where we compute the matching M we compute maximal matchings for certain subgraphs using a greedy method. This can be done in $O(|E|)$ where E is the number of edges in this subgraph. Note that for a single edge e the number of such subgraphs is constant. For the computation of a single vertex v only edges whose adjacent vertices are both at most 381 hops away from v need to be considered. Denote their cardinality by $\bar{E}$. So the overall processing time for v is $O\left(\bar{E}\right)$. $\square$

5. Conclusion

We presented local $1-\epsilon$ approximation algorithms for matching in the setting of location aware unit disk graphs and growth-bounded graphs without positional information. They are the first local approximation algorithms for matching in their respective settings. Since a local algorithm cannot perform optimally in all graph instances our approximation factors are the best possible. It remains open to find local algorithms which achieve the same approximation ratios but which need lower locality distances. For real applications low localities are always desirable since they reduce the size of the area that needs to be explored when computing the status of an edge. For Algorithm 2 the locality distance needed for computing a maximal independent set plays an important role. A local algorithm for this task with a lower locality would immediately lead to a lower locality distance of our algorithm. Also of interest would be lower bounds for the best possible approximation ratio of local algorithms for matching in these settings (depending on their locality distance).

In Section 4 we used the two matching algorithms for getting factor 2 approximation algorithms for vertex cover in the respective settings. Our algorithms achieve the best known locality distances for this approximation factor. For the setting of growth-bounded graphs without positional information, our algorithm is even the first non-trivial local algorithm for vertex cover. It remains open to fully analyze the price for good approximation ratios in terms of required locality distance. The first lower bounds on this are [13]. All improvements for the matching algorithms regarding locality distance would immediately lead to better locality distances for the vertex cover algorithms.

References

[1] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–45, 1985.

[2] Lali Barrière, Pierre Fraigniaud, and Lata Narayanan. Robust position-based routing in wireless ad hoc networks with unstable transmission ranges. In *DIAL-M*, pages 19–27. ACM, 2001.

[3] Steven A. Borbash and Anthony Ephremides. The feasibility of matchings in a wireless network. *IEEE/ACM Transactions on Networking*, 14(SI):2749–2755, June 2006.

[4] I. Dinur and S. Safra. The importance of being biased. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC-02)*, pages 33–42, New York, May 19–21 2002. ACM Press.

[5] J. Edmonds. Paths, trees, and flowers. *Canadian J. Math.*, 17:449–467, 1965.

[6] M. R. Garey and D. S. Johnson. Computers and intractability: A guide to the theory of NP-completeness. 1979.

[7] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, December 1973.

[8] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, 26(2):238–274, 1998.

[9] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer. Fast deterministic distributed maximal independent set computation on growth-bounded graphs. In Pierre Fraigniaud, editor, *DISC*, volume 3724 of *Lecture Notes in Computer Science*, pages 273–287. Springer, 2005.

[10] F. Kuhn, T. Nieberg, T. Moscibroda, and R. Wattenhofer. Local approximation schemes for ad hoc and sensor networks. In *DIALM-POMC '05: Proceedings of the 2005 joint workshop on Foundations of mobile computing*, pages 97–103, New York, NY, USA, 2005. ACM Press.

[11] N. Linial. Locality in distributed graph algorithms. *SIAM J. Comput.*, 21(1):193–201, 1992, February.

[12] L. Tassiulas and S. Sarkar. Maxmin fair scheduling in wireless networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, volume 2 of *Proceedings IEEE INFOCOM 2002*, pages 763–772, Piscataway, NJ, USA, June 23–27 2002. IEEE Computer Society.

[13] A. Wiese and E. Kranakis. Impact of locality on location aware unit disk graphs. *to appear*, 2007.

[14] A. Wiese and E. Kranakis. Local PTAS for Independent Set and Vertex Cover in Location Aware UDGs. *to appear*, 2007.