LOCAL CONSTRUCTION AND COLORING OF SPANNERS OF LOCATION AWARE UNIT DISK GRAPHS

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

Abstract. We look at the problem of coloring locally specially constructed spanners of unit disk graphs. First we present a local approximation algorithm for the vertex coloring problem in Unit Disk Graphs (UDGs) which uses at most four times as many colors as an optimal solution requires. Next we look at the colorability of spanners of UDGs. In particular we present a local algorithm for constructing a 4-colorable spanner of a unit disk graph. The output consists of the spanner and the 4-coloring. The computed spanner also has the properties that it is planar, the degree of a vertex in the spanner is at most 5 and the angles between two edges are at least $\pi/3$. By enlarging the locality distance (i.e. the size of the neighborhood which a vertex has to explore in order to compute its color) we can ensure the total weight of the spanner to be arbitrarily close to the weight of a minimum spanning tree.

We prove that a local algorithm cannot compute a bipartite spanner of a unit disk graph and therefore our algorithm needs at most one color more than any local algorithm for the task requires. Moreover, we prove that there is no local algorithm for 3-coloring UDGs or spanners of UDGs, even if the 3-colorability of the graph (or the spanner respectively) is guaranteed in advance.

1. Introduction

Graph coloring problems have numerous applications in scheduling and channel assignment problems. For example in channel assignment, they are modeled by a graph in which two vertices are connected by an edge if the broadcasting units of their respective nodes interfere and therefore have to be assigned different channels. Since channels in the frequency band are limited and expensive resources the aim is to minimize the total number of used frequencies.

In the case of ad hoc networks, where there is no global entity which could assign channels (colors) to the nodes, we are interested in local algorithms. These are algorithms where the color of a vertex v depends only on the vertices which are a constant number of hops (edges) away from v. This ensures that messages do not propagate uncontrollably far through the network. This concept of locality is also advantageous in dynamically changing networks, since if only local changes occur we do not have to recompute the entire solution, but only parts of it. Also in the

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).

event of a disaster recovery we can take advantage of the fact that we can recover parts of the solution without having to repeat the computation for the entire graph.

Unit Disk Graphs (UDGs) are widely used for modelling wireless networks. In these graphs connectivity between two nodes is established if and only if their Euclidean distance is not larger than one unit, i.e. we assume that the wireless devices have an identical transmission range. In the graph model used in this paper we also assume that each node knows about its geographic position in the plane, e.g. from a GPS receiver or from virtual coordinates assigned by another source. As GPS receivers become more and more customary this model seems to be relevant.

Spanners of unit disk graphs are used for maintaining topology control of the network. This is important for routing and conservation of resources like power and memory which are often limited in wireless devices. There are several properties which are desirable for a spanner, e.g. connectivity, planarity, small node degree, small stretch factor and small total weight. As no spanner can perform well for all of them, one wants to obtain the best possible trade-offs between these properties.

1.1. **Related work.** Graph coloring is a well studied subject in the literature. For general graphs it is NP-complete and even approximating it within a constant ratio is NP-hard [15]. For unit disk graphs the problem remains NP-complete [6], even when it is restricted to a fixed number of colors $k \geq 3$ [10]. However, for UDGs it can be approximated within a constant factor. Marathe et al. [16] present an offline-coloring algorithm with an approximation factor of 3 and an online-coloring algorithm with an approximation ratio of 6. Both algorithms do not need the embedding of the graph as part of the input. The online-algorithm is essentially the sequential coloring algorithm (consider the vertices in any order and color a vertex with the smallest color number allowed for this vertex). In [10] it is stated that Peeters in [19] has shown that this method applied to a "lexicographic" vertex ordering colors a unit disk graph G with at most $3\omega(G) - 2$ colors (where $\omega(G)$ is the clique number of G).

Gräf et al. present a factor 3 approximation algorithm [10] for the case where the embedding of the graph is known. Their algorithm exploits the topology of the graph. For the setting of location aware nodes no algorithm has been known before which outperforms the online algorithm mentioned above (whose idea could be applied in this setting).

The problem of constructing spanning subgraphs (spanners) of geometric graphs has been studied widely in the literature. There are many optimization results for tradeoffs between size, diameter, maximum degree and strech factor of the computed spanner, e.g. Eppstein [8], Arya et al. [2], Narasimhan and Smid [18] and Bose et al. [3]. However, all these algorithms are global, i.e. they need the whole graph as the input.

When looking for local algorithms for constructing spanners of unit disk graphs, Bose et al. [4] address this problem by constructing a planar spanner using the Gabriel test [9]. Li et al. [11] present a local algorithm which computes a planar spanner with constant stretch factor. In [20, 12] Li and Wang introduce local algorithms which compute planar spanners with constant stretch factor and a constant maximum degree. However, the resulting maximum degree can be up to 20 and the weight of the edges can be much higher than in a minimum spanning tree (MST). Li et al. [13] present a local algorithm which computes a planar spanner of a unit disk graph with a node degree bounded by 6. Chavez et al. [5] further analysed this algorithm when operating on quasi unit disk graphs, proved an upper bound for the weight of the spanner in comparison with an MST, and improved the maximum node degree to 5 for the case of unit disk graphs. Every planar graph and therefore every planar spanner of a unit disk graph can be colored with at most 4 colors due to the well known Four-Color-Theorem [1]. However, the algorithm presented there cannot be implemented as a local algorithm. Czyzowicz et al. [7] present a local algorithm which colors a given planar spanner of a unit disk graph with at most 7 colors.

1.2. **Main result and outline of the paper.** In this paper we present a local algorithm with polynomial processing time which colors the vertices of a unit disk graph and needs at most most 4 times as many colors as an optimal coloring requires. It is the first local algorithm for this task. Its approximation ratio is better than the ratio of 6 which is guaranteed by the online algorithm [16], but a bit higher than the performance ratio of 3 which is achieved by the best global polynomial time algorithms [16, 10]. Allowing exponential processing time we improve the approximation ratio to 3.

We also present a local algorithm which computes a 5-colorable spanner of a unit disk graph. It also finds a 5-coloring for the computed spanner. By employing the local algorithm presented in [5] for preprocessing, we can guarantee that our spanner is planar, the maximum node degree is bounded by 5 and any angle between two edges is at least $\pi/3$. As described in [5] we can also force its weight to be at most $(k+1)/(k-1)$ times the weight of a minimum spanning tree for an arbitrary large k. The locality distance (the size of the neighborhood which a vertex has to explore in order to compute its color) of the algorithm is $34+k$. We improve this to a local algorithm which computes a 4-colorable spanner of a unit disk graph and the 4-coloring for it but at the cost of using a higher locality distance of $136+k$. This spanner also has the above properties. These are the first local algorithms which compute spanners of unit disk graph while computing colorings for them. Using at most 4 colors, we need fewer colors than the local 7-coloring algorithm in [7] which colors an arbitrary planar spanner of a unit disk graph.

Further we show that there is no local algorithm for computing bipartite spanners, even if we do not compute the coloring but only the spanner itself. We also show that there is no local algorithm for coloring 3-colorable unit disk graphs or 3-colorable spanners of unit disk graphs using at most 3 colors. Finally we prove a lower bound for the approximation ratio of a local algorithm for vertex coloring.

The remainder of the paper is organized as follows: First we introduce some preliminaries in Section 2. In Sections 3 and 4 we present our local algorithms for vertex coloring with approximation ratios 4 and 3 respectively. In Sections 5 and 6 we present our algorithms for computing the 5- and 4-colorable spanners. We prove the impossibility results for local algorithms mentioned above in Section 7. Finally in Section 8 we summarize all our results and address open problems.

2. Preliminaries

In this section we introduce some definitions and notations that we are going to use, including the concept of local algorithms. All algorithms presented in this paper are local algorithms for unit disk graphs. An undirected graph $G=(V,E)$ is a *unit disk graph* if there is an embedding in the plane for G such that two vertices

u and v are connected by an edge if and only if the Euclidean distance between them is at most 1. The graph G we consider for all our algorithms is a connected unit disk graph.

Definition 1. 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. For ease of notation we set $N^0(v) := \{v\}$, $N(v) := N^1(v)$ and for a set $V' \subseteq V$ we define $N(V') = \bigcup_{v' \in V'} N(v')$. Note that $v \in N(v)$. We define the diameter of a set of vertices $V' \subseteq V$ as $diam(V') := \max_{u,v \in V'} d(u,v)$. It is assumed that in all our algorithms an embedding for G is given. For a vertex v we denote by v_x its x-coordinate and by v_y its y-coordinate. We denote by $ht(G)$ the *height* of G, defined by $ht(G) := \underset{u,v \in V}{max}\{u_y - v_y\}$.

We denote by the *locality distance* (or short the *locality*) of an algorithm the minimum α such that the status of any vertex v (e.g. its color, whether or not it is in a computed set etc.) depends only on the vertices in $N^\alpha(v)$. So for any vertex v messages emanating from v never propagate beyond $N^\alpha(v)$. For all algorithms presented in this paper we will prove that α is constant. In the graph model which we use we assume that each vertex v is aware of its geographic position in the plane. We also assume that each vertex v can find out the geographic position of the vertices which are at most α hops away from v by message passing.

A *coloring* of a graph G is a map $color : V \to \{1, ..., c\}$ such that $(v_1, v_2) \in E \Rightarrow color(v_1) \neq color(v_2)$. For ease of notation we define $|color| := c$. We denote by $\chi(G)$ the chromatic number of G. That is the minimum number of colors that is needed for a coloring of G. We define $\omega(G)$ to be the clique number, i.e. the size of the largest clique in G.

We denote by $\triangle(G)$ the maximum degree of a node in G. If G is a geometric graph, we define $cost(G)$ as the sum of Euclidean lengths of the edges of G. For a graph G we denote by $E(G)$ the set of its edges and by $V(G)$ the set of its vertices. For a set of vertices V' we denote by $G[V']$ the subgraph of G induced by V'. If an embedding of G is given, then for a rectangle R in the plane we denote by $G[R]$ the subgraph of G induced by the vertices in R.

3. Local $4 \cdot \chi(G)$ Vertex Coloring of UDGs

In this section we present a local approximation algorithm for vertex coloring in a unit disk graph G. We prove that it achieves a competitive ratio of 4 and that the processing time for each vertex is bounded by a polynomial. We employ a result by Gräf et al. [10] that enables us to use an algorithm by Möhring [17] as a subroutine which colors a unit disk graph optimally in polynomial time when its height is at most $\sqrt{3}/2$. Before we present the algorithm we introduce a tiling of the plane that we are going to use.

3.1. **Tiling of the plane.** We divide the plane into rectangles and assign a class number to each rectangle. The tiling achieves the following properties:

- Each vertex of G is in exactly one rectangle.
- The height of each rectangle is smaller than $\sqrt{3}/2$.

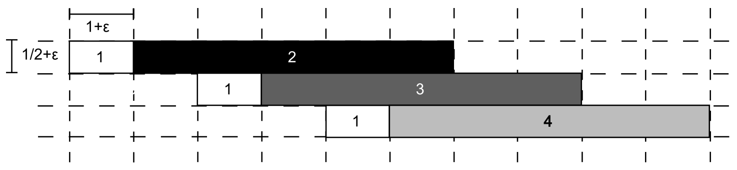

FIGURE 1. One tile of the tiling. The numbers in the rectangles indicate the class number of the respective rectangle.

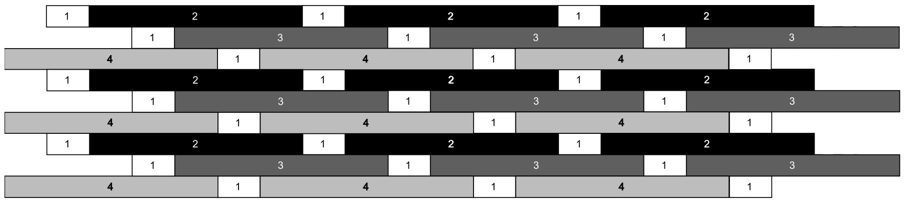

FIGURE 2. A part of the tiling of the plane used in Algorithm 1.

- Each rectangle has a class number between 1 and 4.
- The Euclidean distance between two rectangles with the same class number is strictly greater than one.

We achieve these properties as follows: We divide the plane into a grid where each grid cell is a rectangle with height $1/2 + \epsilon$ and width $1 + \epsilon$. We choose ϵ such that $0 < \epsilon \leq \frac{1}{64}$. We place tiles of rectangles into the grid. Figure 1 shows one tile. The rectangles of class 1 have the size of 1 grid cell, the rectangles of classes $2, 3$ and 4 have the size of 5 grid cells (later, we use the different sizes of rectangles in order to achieve a lower locality distance in our algorithm). The class numbers of the rectangles are assigned according to Figure 1 (white=class 1, black=class 2, dark gray=class 3 and light gray=class 4). We tile the whole plane with such tiles, starting at an arbitrary position. Figure 2 shows an extract of this tiling.

Each vertex of G is contained in exactly one rectangle. Ambiguities caused by vertices on the border of a rectangle are resolved by assigning them to the rectangle with the lowest class number which containes them. From the construction it follows that two different rectangles of the same class have an Euclidean distance of strictly more than one. So we conclude with the following proposition.

Proposition 1. *Two vertices in different rectangles of the same class are not adjacent.*

We observe that every vertex can determine its class number in constant time by only using its coordinates.

3.2. **The algorithm.** Now we present our algorithm. The main idea is to solve the coloring problem optimally for the rectangles of each class separately. First we color the vertices in all class 1 rectangles optimally. Then we solve the problem for each connected component C in each class 2 rectangle R optimally under the condition that we are not allowed to use colors that have been used by vertices which are adjacent to any vertex in C. Then we do the same for all class 3 rectangles and then for all class 4 rectangles.

Now we present our algorithm in detail. We start with a coloring *color* defined by $color(v) := 0$ for all $v \in V$. For $i := 1, 2, 3, 4$ we do the following: Consider a connected component C in a rectangle R of class i and denote its vertices by V_C. Denote by G_C the subgraph induced by V_C. By construction of the tiling the height of R is smaller than $\sqrt{3}/2$ and thus $ht(G_C) \leq \sqrt{3}/2$. In [10] Gräf et al. state an algorithm which computes an optimal coloring for a unit disk graph G_C in time $O\left(|V_C|\,\omega\,(G_C)^2\right)$ if $ht\,(G_C) \leq \sqrt{3}/2$[1]. We use this algorithm to compute an optimal coloring $color_C$ for G_C. We might not be able to use the assignment of colors in $color_C$ directly in the coloring *color* which has been computed so far since a vertex $v \in C$ might be adjacent to a vertex v' (in another rectangle) such that $color_C(v) = color(v')$. Let c be the highest number of a color that has already been assigned to any vertex in $N(V_C)$ by *color* (i.e. $c = \max\limits_{v \in N(V_C)} color(v)$). We define $color(v_C) := color_C(v_C) + c$ for all $v_C \in V_C$. We do this for all connected components in all rectangles of class i. As two vertices in two different connected components in rectangles of the same class number are not adjacent (see Proposition 1) the order in which the connected components are being processed does not matter. We output the coloring *color*. We refer to the above as Algorithm 1.

Algorithm 1: Algorithm for finding a vertex-coloring in a unit disk graph G

```
1  for i:=1 to 4 do
2      // $i$ denotes the class number of the current iteration;
3      // denote by $\mathcal{R}_i$ the set of all rectangles of class $i$;
4      forall $R \in \mathcal{R}_i$ do
5          // denote by $V_R$ the vertices in $R$;
6          find an optimal coloring $color_C$ for the vertices $V_R$;
7          let $c := \max_{v \in N(V_R)} color(v)$;
8          $color(v) := color_C(v) + c$ for all $v \in V_R$;
9      end
10 end
   output: Coloring color
```

3.3. **Proof of correctness.** In the following theorem we prove that Algorithm 1 is a local algorithm that computes a valid coloring with a competitive ratio of 4.

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

(1) *The computed coloring is a valid coloring for G.*
(2) *It holds that $|color| \leq 4 \cdot \chi(G)$.*
(3) *The color of a vertex v depends only on the vertices which are at most* 71 *hops away from v, i.e. Algorithm 1 is local.*
(4) *The processing time for a vertex v is bounded by a cubic polynomial in the number of vertices which are at most* 71 *hops away from v.*

We will prove the four parts of this theorem in four steps.

[1]In [10] Gräf et al. call such graphs $\sqrt{3}/2$-stripes. They show that a unit disk graph G_C is a cocomparability graph if $ht(G_C) \leq \sqrt{3}/2$. This allows to employ an algorithm by Möhring [17] for coloring cocomparability graphs in order to compute an optimal coloring for G_C.

3.3.1. *Correctness.* We prove that the coloring *color* computed by Algorithm 1 above is a valid coloring, i.e. $(v_1, v_2) \in E \Rightarrow \mathit{color}(v_1) \neq \mathit{color}(v_2)$.

Proof. (of part 1 of Theorem 1): For the correctness of the subroutine for computing an optimal coloring for one rectangle we refer to [10]. Now assume on the contrary that there are two vertices v_1 and v_2 such that $(v_1, v_2) \in E$ and $\mathit{color}(v_1) = \mathit{color}(v_2)$. We distinguish three cases:

Case 1: the vertices v_1 and v_2 are in the same rectangle R of class i. So they are in the same connected component C. Then in iteration i an optimal coloring color_C for C was computed with $\mathit{color}_C(v_1) \neq \mathit{color}_C(v_2)$. It follows that $\mathit{color}_C(v_1) + c \neq \mathit{color}_C(v_2) + c$ for any c and therefore $\mathit{color}(v_1) \neq \mathit{color}(v_2)$ which is a contradiction.

Case 2: the vertices v_1 and v_2 are in different rectangles of different classes R_1 and R_2 respectively. W.l.o.g. let R_1 be in a smaller class than R_2. Let V_1 be the vertices in R_1 and let V_2 be the vertices in R_2. Then in the iteration where the rectangle R_2 was considered, an optimal coloring color_{R_2} for R_2 was computed. Let c be the highest number of a color that has been assigned to any vertex in $N(V_2)$ by *color* so far. As $v_1 \in N(V_2)$ it follows that $\mathit{color}(v_1) \leq c$. As $\mathit{color}_{R_2}(v_2) > 0$ we have that $\mathit{color}(v_2) = \mathit{color}_{R_2}(v_2) + c > \mathit{color}(v_1)$ so $\mathit{color}(v_1) \neq \mathit{color}(v_2)$ which is a contradiction.

Case 3: the vertices v_1 and v_2 are in different rectangles of the same class. Then from Proposition 1 it follows that $(v_1, v_2) \notin E$ which is a contradiction. □

3.3.2. *Approximation ratio.* We prove that our algorithm has an approximation ratio of 4, i.e. $|\mathit{color}| \leq 4 \cdot \chi(G)$. The main idea is that the number of colors needed for each rectangle class is a lower bound for the optimal coloring and as we have four rectangle classes we achieve a competitive ratio of 4.

Proof. (of part 2 of Theorem 1): Let color_k be the coloring computed after the kth iteration of the algorithm and let $c_k := |\mathit{color}_k|$. We prove that $c_k \leq k \cdot \chi(G)$ and therefore $|\mathit{color}| = c_4 \leq 4 \cdot \chi(G)$.

Proof by induction. Let $k := 1$. Let G_1 be the restriction of G to vertices in class 1 rectangles. It holds that $\chi(G_1) \leq \chi(G)$. As the coloring for the vertices in rectangles of class 1 is optimal, it follows that $c_1 = \chi(G_1) \leq \chi(G)$.

Assume the claim is true for all $k \leq i - 1$. Let G_i be the restriction of G to vertices in class i rectangles. It holds that $\chi(G_i) \leq \chi(G)$. As we color the vertices in G_i with the least number of colors as possible and do not skip any color numbers between 1 and c_i it follows that $c_i \leq c_{i-1} + \chi(G_i) \leq (i-1) \cdot \chi(G) + \chi(G) = i \cdot \chi(G)$. □

3.3.3. *Locality.* We prove that the color of a vertex v depends only on the vertices which are at most 68 hops away from v, i.e. Algorithm 1 is local. First we prove an upper bound for the diameter of the restriction of G to one rectangle. Note that the rectangles of class 1 are smaller than rectangles of class 2, 3 and 4. This is the only part of the proof where the different sizes of the rectangles matter.

Lemma 1. *Let R be a rectangle of class 1 and $G[R]$ the graph G restricted to R. For each connected component C in $G[R]$ it holds that $\mathit{diam}(C) \leq 5$. Let R' be a rectangle of class 2, 3 or 4 and $G[R']$ the graph G restricted to R'. For each connected component C' in $G[R']$ it holds that $\mathit{diam}(C') \leq 21$.*

Proof. We start with proving the claim for the rectangle R of class 1. 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 $(2+\epsilon)\cdot(1.5+\epsilon) = (3+3.5\epsilon+\epsilon^2)$. So there can be at most $\left\lfloor \frac{3+3.5\epsilon+\epsilon^2}{\pi/4} \right\rfloor$ centers of non-overlapping discs of radius $1/2$ in R. As $\epsilon \leq \frac{1}{64}$ we compute that $\left\lfloor \frac{3+3.5\epsilon+\epsilon^2}{\pi/4} \right\rfloor = 3$. It follows that the cardinality of a maximum independent set in $G[R]$ is at most 3. 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 vertice in p we get an independent set in R. As the size of such a set is bounded by 3, the length of p is bounded by 5 and therefore $diam(C) \leq 5$.

Applying the same reasoning to R' we derive an upper bound of 11 for an independent set in $G[R']$ (as $\left\lfloor \frac{(6+5\epsilon)\cdot(1.5+\epsilon)}{\pi/4} \right\rfloor = \left\lfloor \frac{9+13.5\epsilon+5\epsilon^2}{\pi/4} \right\rfloor = 11$ for $\epsilon \leq \frac{1}{64}$) and therefore we get $diam(C') \leq 21$ for any connected component C' in $G[R']$. □

Proof. (of part 3 of Theorem 1): Let v be a vertex and k be the class number of its rectangle. Let a_k be the smallest integer such that the color of v depends only on the vertices which are at most a_k hops away from v. We prove that $a_k \leq 5+22\cdot(k-1)$.

Proof by induction. Suppose $k = 1$. Let v be in a connected component C of a class 1 rectangle R. From Lemma 1 we know that the diameter of C is at most 5. So all other vertices in C are at most 5 hops away from v and the color of v only depends on them. So $a_1 \leq 5$.

Suppose the claim is true for all vertices in classes $k \leq i-1$. Now let v be in a connected component C in a class i rectangle R with $i \geq 2$. From Lemma 1 we know that the diameter of C is at most 21. The color of v depends only on the vertices in C and the colors of the vertices in rectangles of class $i' < i$ in $N(C) \setminus C$. So the color of v depends only on the vertices which are at most $21 + 1 + a_{i-1} \leq 21 + 1 + (5 + 22\cdot(i-1-1)) = 5 + 22\cdot(i-1)$ hops away from v. So $a_i \leq 5 + 22\cdot(i-1)$. As we have four different classes of rectangles, the locality distance of Algorithm 1 equals a_4 and it holds that $a_4 \leq 71$.

□

3.3.4. *Processing time.* The processing time is the time that a single vertex needs in order to compute its color. As our algorithm is local it is not very suitable to quantify the processing time in comparison to the total number of vertices in G. Instead we measure it with respect to the number of vertices which are at most 71 hops away from a vertex v since these are all vertices that a vertex v needs to explore when computing its status. We denote this number by $\bar{n}(v)$ (so $\bar{n}(v) = \left|N^{71}(v)\right|$). We show that the processing time is bounded by $O\left(\bar{n}(v)^3\right)$.

Proof. (of part 4 of Theorem 1): Denote by $\tilde{G}$ the subgraph of G restricted to $N^{71}(v)$. When computing the color for a vertex v, for some rectangles we need to compute a perfect coloring for $\tilde{G}$ restricted to the respective rectangle. With the algorithm stated in [10] this can be done in time $O\left(|V_R|\cdot\omega\left(G_R\right)^2\right)$ for one rectangle R. The number of rectangles whose colorings need to be computed is bounded by a constant (as there is only a limited number of rectangles which can have vertices in $\tilde{G}$). The computation of the colorings for the rectangles dominates

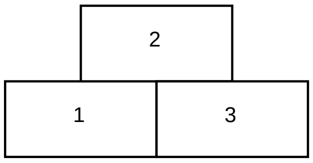

FIGURE 3. One tile for the tiling of the plane

the processing time of the entire algorithm. Thus we get an overall processing time of $O\left(\bar{n}(v)\cdot\omega\left(\tilde{G}\right)^2\right)\subseteq O\left(\bar{n}(v)^3\right)$. □

4. LOCAL $3\cdot\chi(G)$ VERTEX COLORING OF UDGS

In Section 3 we presented a local algorithm with a performance ratio of 4. In this section we present a local algorithm which has a performance ratio of 3, but where the processing time of each node is exponential in the number of nodes in its local neighborhood. Before we present the algorithm itself, we present a tiling of the plane which we are going to use.

4.1. **Tiling of the plane.** The plane is divided into tiles of rectangles. Figure 3 shows one tile. A class number between 1 and 3 is assigned to each rectangle as shown in Figure 3. The whole plane is tiled with such tiles, starting at an arbitrary position. Figure 4 shows an extract of this. The width of each rectangle is $2+\epsilon$, the height of each rectangle is $1+\epsilon$ for any fixed ϵ with $0<\epsilon\leq 1/32$. Each vertex is assigned to the rectangle which contains it. Ambiguities caused by vertices on the edge of rectangles are being resolved by assigning them to the rectangle with the smallest class number which contains them (any other resolving method works as well). We observe that two rectangles of the same class have a Euclidean distance of strictly more than one. So we conclude the following proposition:

Proposition 2. *Two vertices in different rectangles of the same class are not adjacent.*

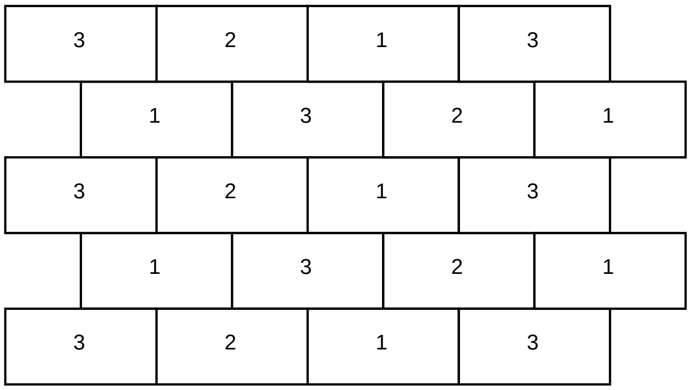

FIGURE 4. Tiling of the plane

Each vertex can compute the rectangle that it belongs to from its coordinates in the plane.

4.2. **The algorithm.** Essentially we use the same ideas as in Algorithm 1. However, in this algorithm we have three different class numbers rather than four as in Algorithm 1. This leads us to an approximation ratio of 3. The price we have to pay for this improved approximation factor is that the processing time in each vertex is now exponential rather than polynomial as in Algorithm 1.

Now we present our algorithm in detail. We start with a coloring *color* defined by $\mathit{color}(v) := 0$ for all $v \in V$. For $i := 1, 2, 3$ we do the following: Consider a connected component C in a rectangle R of class i and denote its vertices by V_C. Denote by G_C the subgraph induced by V_C. We compute an optimal coloring color_C for G_C by enumeration. We might not be able to use the assignment of colors in color_C directly in the coloring *color* which has been computed so far since a vertex $v \in C$ might be adjacent to a vertex v' (in another rectangle) such that $\mathit{color}_C(v) = \mathit{color}(v')$. Let c be the highest number of a color that has already been assigned to any vertex in $N(V_C)$ by *color* (i.e. $c = \max_{v \in N(V_C)} \mathit{color}(v)$). We define $\mathit{color}(v_C) := \mathit{color}_C(v_C) + c$ for all $v_C \in V_C$. We do this for all connected components in all rectangles of class i. As two vertices in two different connected components in rectangles of the same class number are not adjacent (see Proposition 2) the order in which the connected components are being processed does not matter. We output the coloring *color*. We refer to the above as Algorithm 2.

Algorithm 2: Algorithm for finding a vertex-coloring in a unit disk graph G

```
1  for $i$:=1 to 3 do
2      // $i$ denotes the class number of the current iteration;
3      // denote by $\mathcal{R}_i$ the set of all rectangles of class $i$;
4      forall $R \in \mathcal{R}_i$ do
5          // denote by $V_R$ the vertices in $R$;
6          find an optimal coloring $\mathit{color}_C$ for the vertices $V_R$;
7          let $c := \max_{v \in N(V_R)} \mathit{color}(v)$;
8          $\mathit{color}(v) := \mathit{color}_C(v) + c$ for all $v \in V_R$;
9      end
10 end
   output: Coloring color
```

4.3. **Proof of correctness.** In the following theorem we prove that Algorithm 2 is a local algorithm that computes a valid coloring with a competitive ratio of 3.

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

(1) *The computed coloring is a valid coloring for G.*
(2) *It holds that $|\mathit{color}| \leq 3 \cdot \chi(G)$.*
(3) *The color of a vertex v depends only on the vertices which are at most* 42 *hops away from v, i.e. Algorithm 2 is local.*

Note that in contrast to Algorithm 1 we cannot give a polynomial bound for the processing time in each vertex. This is due to the exponential time that it takes to compute an optimal vertex coloring for one rectangle.

We will prove the four parts of this theorem in four steps. The proof uses concepts which are similar to the proof of Theorem 1.

4.3.1. *Correctness.* We prove that the coloring *color* computed by Algorithm 2 above is a valid coloring, i.e. $(v_1, v_2) \in E \Rightarrow \mathit{color}\,(v_1) \neq \mathit{color}\,(v_2)$.

Proof. (of part 1 of Theorem 2): We assume that the enumeration subroutine which computes optimal colorings for one rectangle is correct.

Now assume on the contrary that there are two vertices v_1 and v_2 such that $(v_1, v_2) \in E$ and $\mathit{color}\,(v_1) = \mathit{color}\,(v_2)$. We distinguish three cases:

Case 1: the vertices v_1 and v_2 are in the same rectangle R of class i. So they are in the same connected component C. Then in iteration i an optimal coloring color_C for C was computed with $\mathit{color}_C\,(v_1) \neq \mathit{color}_C\,(v_2)$. It follows that $\mathit{color}_C\,(v_1) + c \neq \mathit{color}_C\,(v_2) + c$ for any c and therefore $\mathit{color}\,(v_1) \neq \mathit{color}\,(v_2)$ which is a contradiction.

Case 2: the vertices v_1 and v_2 are in different rectangles of different classes R_1 and R_2 respectively. W.l.o.g. let R_1 be in a smaller class than R_2. Let V_1 be the vertices in R_1 and let V_2 be the vertices in R_2. Then in the iteration where the rectangle R_2 was considered, an optimal coloring color_{R_2} for R_2 was computed. Let c be the highest number of a color that has been assigned to any vertex in $N\,(V_2)$ by *color* so far. As $v_1 \in N(V_2)$ it follows that $\mathit{color}\,(v_1) \leq c$. As $\mathit{color}_{R_2}\,(v_2) > 0$ we have that $\mathit{color}\,(v_2) = \mathit{color}_{R_2}\,(v_2) + c > \mathit{color}\,(v_1)$ so $\mathit{color}\,(v_1) \neq \mathit{color}\,(v_2)$ which is a contradiction.

Case 3: the vertices v_1 and v_2 are in different rectangles of the same class. Then from Proposition 2 it follows that $(v_1, v_2) \notin E$ which is a contradiction. □

4.3.2. *Approximation ratio.* We prove that our algorithm has an approximation ratio of 3, i.e. $|\mathit{color}| \leq 3 \cdot \chi(G)$. The main idea is that the number of colors needed for each rectangle class is a lower bound for the optimal coloring and as we have three rectangle classes we achieve a competitive ratio of 3.

Proof. (of part 2 of Theorem 2): Let color_k be the coloring computed after the kth iteration of the algorithm and let $c_k := |\mathit{color}_k|$. We prove that $c_k \leq k \cdot \chi(G)$ and therefore $|\mathit{color}| = c_3 \leq 3 \cdot \chi(G)$.

Proof by induction. Let $k := 1$. Let G_1 be the restriction of G to vertices in class 1 rectangles. It holds that $\chi\,(G_1) \leq \chi(G)$. As the coloring for the vertices in rectangles of class 1 is optimal, it follows that $c_1 = \chi\,(G_1) \leq \chi(G)$.

Assume the claim is true for all $k \leq i - 1$. Let G_i be the restriction of G to vertices in class i rectangles. It holds that $\chi\,(G_i) \leq \chi(G)$. As we color the vertices in G_i with the least number of colors as possible and do not skip any color numbers between 1 and c_i it follows that $c_i \leq c_{i-1} + \chi\,(G_i) \leq (i-1) \cdot \chi\,(G) + \chi\,(G) = i \cdot \chi(G)$. □

4.3.3. *Locality.* We prove that the color of a vertex v depends only on the vertices which are at most 42 hops away from v, i.e. Algorithm 2 is local. First we prove an upper bound for the diameter of the restriction of G to one rectangle. This lemma uses the same technique as Lemma 1.

Lemma 2. *Let R be a rectangle and let $G[R]$ be the graph G restricted to R. For each connected component C in $G[R]$ it holds that $\mathit{diam}(C) \leq 13$.*

Proof. We start with proving the claim for the rectangle R of class 1. 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 $(2+\epsilon)\cdot(3+\epsilon) = (6+5\epsilon+\epsilon^2)$. So there can be at most $\left\lfloor \frac{6+5\epsilon+\epsilon^2}{\pi/4} \right\rfloor$ centers of non-overlapping discs of radius 1/2 in R. As $\epsilon \leq \frac{1}{32}$ we compute that $\left\lfloor \frac{6+5\epsilon+\epsilon^2}{\pi/4} \right\rfloor = 7$. It follows that the cardinality of a maximum independent set in $G[R]$ is at most 7. 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 vertice in p we get an independent set in R. As the size of such a set is bounded by 7, the length of p is bounded by 13 and therefore $diam(C) \leq 13$. □

Proof. (of part 3 of Theorem 2): Let v be a vertex and k be the class number of its rectangle. Let a_k be the smallest integer such that the color of v depends only on the vertices which are at most a_k hops away from v. We prove that $a_k \leq 14\cdot k$.

Proof by induction. Suppose $k = 1$. Let v be in a connected component C of a class 1 rectangle R. From Lemma 2 we know that the diameter of C is at most 13. So all other vertices in C are at most 13 hops away from v and the color of v only depends on them. So $a_1 \leq 14$.

Suppose the claim is true for all vertices in classes $k \leq i-1$. Now let v be in a connected component C in a class i rectangle R with $i \geq 2$. From Lemma 2 we know that the diameter of C is at most 13. The color of v depends only on the vertices in C and the colors of the vertices in rectangles of class $i' < i$ in $N(C) \setminus C$. So the color of v depends only on the vertices which are at most $13 + 1 + a_{i-1} \leq 13 + 1 + 14\cdot(i-1) = 14\cdot i$ hops away from v. So $a_i \leq 14\cdot i$. As we have three different classes of rectangles, the locality distance of Algorithm 2 equals a_3 and it holds that $a_3 \leq 42$. □

5. Local construction of 5 colorable spanner

In this section we present a local algorithm for computing a 5-colorable spanner of a given unit disk graph. It computes the spanner and the 5-coloring for it. For preprocessing the graph we employ the local algorithm presented in [5, 13]. This ensures that the resulting spanner is planar, it does not contain any angle smaller than $\pi/3$ and the degree of any node is at most 5. For arbitrarily large k this subroutine can also guarantee the weight of the spanner to be at most $\frac{k+1}{k-1}$ times the weight of a minimum spanning tree. The locality distance of the algorithm is in $O(k)$.

5.1. **Outline of the algorithm.** We consider a unit disk graph $G = (V, E)$. The algorithm colors the vertices of G in five colors and computes a set $E' \subseteq E$ such that $G' = (V, E')$ forms a spanner for G. The algorithm has three steps:

(1) We fix an integer k and compute a spanner G_k of G using the algorithm presented in [5, 13]. We continue the computation with G_k.
(2) The plane is divided into rectangles. A bipartite spanner for each rectangle is computed and the vertices in each rectangle are colored using two colors. We employ altogether four different colors in this step.
(3) Collisions (i.e. two adjacent vertices with the same color) are being resolved by using an additional fifth color.

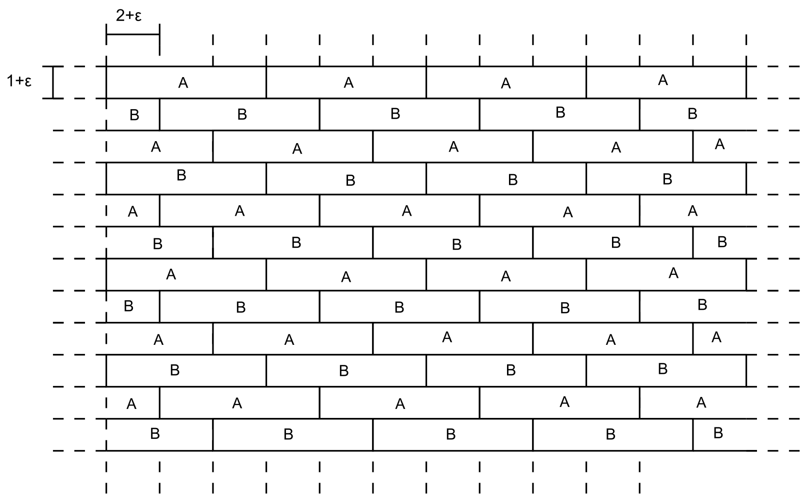

FIGURE 5. The tiling of the plane in rectangles. The line segments between two rectangles of the same class form the glues.

5.2. **The algorithm.** We consider a unit disk graph $G = (V, E)$. First we present the tiling of the plane which we are going to use. Then we present the three steps of the algorithm as outlined above. We will compute a set $E' \subseteq E$ such that $G' = (V, E')$ forms a spanner for G. We will also compute a coloring *color* for G' with $|color| \leq 5$.

5.2.1. *Tiling of the plane.* We divide the plane into a grid where each grid cell has a height of $1 + \epsilon$ and a width of $2 + \epsilon$ (for any ϵ with $0 < \epsilon \leq 1/128$). Rectangles with the size of 3 grid cells are placed in the grid according to Figure 5. Each vertex of G is in exactly one rectangle. Ambiguities caused by vertices at the edge of rectangles are resolved by assigning them to the rectangle whose upper left corner has the lowest x-coordinate (any other resolving method works as well).

We see that the rectangles form rows. The rows are assigned to classes A and B such that two adjacent rows have different class numbers. We say a rectangle is of class A or B if it is in a row of class A or B respectively (see Figure 5). Denote by $class(R)$ the class of the rectangle R. We define by *glue* the line segment between two rectangles of the same class (see Figure 6). We observe the following proposition:

Proposition 3. *Two different glues have an Euclidean distance of at least* $2 + \epsilon$.

Proposition 4. *Two adjacent vertices in rectangles of the same class are in the same row.*

5.2.2. *Step 1: Computing the spanner* G_k. In this step we compute a spanner G_k for G. This routine is taken from [5, 13]. As the spanner which we output later will be a subgraph of G_k, our spanner will inherit some properties from G_k. First we need to define an ordering on the edges of G.

Definition 2. *(Compatible Linear Order).* Each edge (u, v) is assigned a 5-tuple $(|u, v|, x_1, y_1, x_2, y_2)$, where $|u, v|$ is the Euclidean length of the edge, x_1, y_1 and

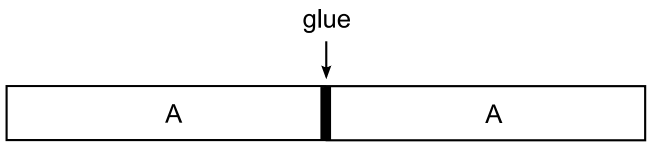

FIGURE 6. Glue between two rectangles of class A

x_2, y_2 are the coordinates of the endnodes of the edge with either $x_1 > x_2$ or both $x_1 = x_2$ and $y_1 > y_2$. Clearly this gives a unique 5-tuple to any edge, and 5-tuples assigned to any two edges are distinct. The linear order $\prec$ is defined using the lexicographic ordering of the assigned 5-tuples.

A graph may have several minimum spanning trees (MST) when the Euclidean length of the edges is the cost function. However, if we break the ties when an edge is chosen in the MST-algorithm (e.g. in Kruskal's algorithm) by the linear order $\prec$, then the graph has a unique MST (which can be computed for example by Kruskal's algorithm).

In step 1 of our algorithm we do the following: First we fix an integer $k \geq 2$. Then we compute the spanner G_k as explained in the description of the algorithm.

Step 1 of Algorithm 6: Computing a planar spanner with certain properties. This algorithm was presented in [13, 5]

```
1 // Algorithm is executed independently by each node;
2 // the parameter $k$ is fixed
3 Learn your distance $k$ neighborhood $N^k(v)$;
4 Constuct locally the unique MST $T^k(v)$ of $N^k(v)$;
5 Broadcast in $N^k(v)$ the edges of $N^1(v)$ which have been retained in $T^k(v)$
  (i.e. $N^1(v) \cap T^k(v)$);
6 The output spanner $G_k$ is defined as follows: an edge is selected into $G_k$ if
  and only if it was retained by both of its incident nodes;
```

5.2.3. *Step 2: Bipartite spanner for each rectangle.* In this step, we define a map *color* : $V \to \mathbb{N}$. We also define the set $E' \subseteq E$. Note that after this step the map *color* will not necessarily be a valid coloring for $G' = (V, E')$.

From now on, we ignore all edges which are not part of G_k. We color the vertices of each connected component in each rectangle independently. For each connected component C in a rectangle R do the following: Compute a spanning tree T for C and compute a coloring $color_T$ for T which uses at most two colors. For the two-coloring of C we use colors 1 and 2 if $class(R) = A$ and colors 3 and 4 if $class(R) = B$. We define $color(v) := color_T(v)$. Assign the edges of T to the set E'. Do this for all connected components in all rectangles. Finally we add all edges to E' which connect two vertices which are in different rectangles. The above is presented in step 2 of Algorithm 6.

5.2.4. *Step 3: Resolving collisions.* Now $color(v)$ is well-defined for every vertex v. However, *color* might not be a valid coloring for G' yet as there might be collisions. By a collision we mean an edge $e = (v_1, v_2) \in E'$ with $color\,(v_1) = color\,(v_2)$. Since

Step 2 of Algorithm 6: Computation of bipartite spanners and two-colorings for all connected components in all rectangles

```
1  // Algorithm is executed independently by each node $v$;
2  // Consider only edges from $G_k$ as computed in step 1, ignore all edges which
   are not in $G_k$
3  Let $R$ be rectangle that $v$ has been assigned to;
4  Determine whether $R$ is of type A or B;
5  Explore the connected component $C$ in $R$ that $v$ belongs to;
6  Compute a spanning tree $T$ for $C$;
7  if $R$ is of type A then compute a bipartite coloring $color_T$ for $T$ which uses
   colors 1 and 2;
8  if $R$ is of type B then compute a bipartite coloring $color_T$ for $T$ which uses
   colors 3 and 4;
9  $color(v) := color_T(v)$;
10 Add all edges from $T$ to $E'$;
11 Add all edges to $E'$ which connect vertices in $C$ to vertices in rectangles other
   than $R$;
```

vertices in different rows are assigned different colors, from Proposition 4 and the way the plane is tiled we see that such a collision edge e has to cross exactly one glue.

For all pairs of adjacent rectangles of the same class R_1 and R_2 which contain vertices we do the following: Denote by V_1 the vertices in R_1 and by V_2 the vertices in R_2. Consider all vertices in V_1 that are adjacent to a vertex in V_2 and all vertices in V_2 that are adjacent to a vertex in V_1 (i.e. $N(V_1) \cap N(V_2) \cap (V_1 \cup V_2)$). Denote these vertices by V_{glue} and denote by G_{glue} the subgraph of G induced by V_{glue} (note that there is one glue which all edges in G_{glue} cross). For every connected component C_{glue} in G_{glue} we compute a spanning tree T_{glue}. As T_{glue} is bipartite, we can compute a coloring $color_T$ for T_{glue} that uses at most two colors. Denote these colors by $T1$ and $T2$. Now we recolor some of the vertices in V_{glue} as follows: If for a vertex v it holds that $color_T(v) = T1$ then $color(v) := 5$. We remove all edges in $E(G_{glue}) \setminus E(T_{glue})$ from E'.

We output $G' = (V, E')$ as the computed spanner and *color* as the computed coloring. The above description of step 3 of Algorithm 6. We summarize the whole computation in Algorithm 6.

5.3. **Proof of correctness.** We prove the correctness of Algorithm 6 and the properties of the computed spanner G' and the coloring *color* in Theorem 3.

Theorem 3. *Let $k \geq 2$ be the integer which was fixed at the beginning of Algorithm 6. For the computed spanner $G' = (V, E')$ and the computed coloring color it holds that*

(1) *G' is connected,*
(2) *color is a valid coloring for G',*
(3) *$|color| \leq 5$,*
(4) *G' is planar,*
(5) *$\triangle(G') \leq 5$,*
(6) *no angle between two edges in G' is smaller than $\pi/3$,*

Step 3 of Algorithm 6: Resolving collisions by using an additional 5th color.

```
// The Algorithm is executed independently by each node $v$;
Let $R_1$ the rectangle that contains $v$;
Let $V_1$ be the vertices in $R_1$;
Let $c$ be the class of $R_1$;
if there is a rectangle $R_2$ with $class(R_2) = c$ and $R_1 \neq R_2$ and there is a vertex $v' \in N(v) \cap V_2$ then
    Let $V_2$ be the vertices in $R_2$;
    Define $V_{glue} := N(V_1) \cap N(V_2) \cap (V_1 \cup V_2)$;
    Let $G_{glue}$ be the subgraph induces by $V_{glue}$;
    Determine the connected component $C_{glue}$ of $G_{glue}$ which contains $v$;
    Find a spanning tree $T_{glue}$ for $C_{glue}$;
    Find a two-coloring $color_T$ for $T_{glue}$;
    // We assume that $color_T$ uses the colors T1 and T2;
    if $color_T(v) = T1$ then
        $color(v) := 5$;
    end
    Remove all edges $E(G_{glue}) \setminus E(T_{glue})$ from $E'$;
end
```

Algorithm 6: Local Algorithm for computing a spanner and a 5-coloring for a unit disk graph

```
// The Algorithm is executed independently by each node $v$;
Fix an integer $k \geq 2$;
Compute the spanner $G_k$ as stated in step 1;
For all vertices $v$ compute $color(v)$ and compute the spanner $E'$ according to step 2;
For all vertices $v$ check whether $color(v)$ is changed in step 3 and what edges of $E'$ remain after step 3;
```

(7) *for a minimum spanning tree T for G it holds that $cost(G') \leq \frac{k+1}{k-1} \cdot cost(T)$ and*

(8) *the locality distance of Algorithm 6 is bounded by $34 + k$, i.e. Algorithm 6 is local.*

First we present a result from [5] for G_k in Lemma 3 and prove another property of G_k in Lemma 4. Then we give an upper bound for the diameter of a connected component similar to Lemma 5. Finally we prove the theorem.

Lemma 3. *Let T be a minimum spanning tree for G and G_k be the graph computed in step 1. Then it holds that*

- *G_k is connected,*
- *G_k is planar,*
- *$\triangle(G_k) \leq 5$ and*
- *$cost(G_k) \leq \frac{k+1}{k-1} \cdot cost(T)$*

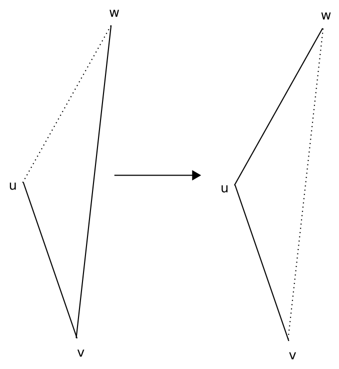

FIGURE 7. The angle $\angle uvw$ is smaller than $\pi/3$. The vertex v finds a smaller spanning tree by replacing the edge (v, w) by the edge (u, w).

Proof. In [5] the local algorithm which we employ in step 1 is presented and the above lemma is presented as Corollary 1. $\square$

Lemma 4. *Let $u, v, w \in V$ be vertices. If (u, v) and (v, w) are edges in G_k, then the angle $\angle u, v, w$ is at least $\pi/3$.*

Proof. Assume on the contrary that there are two edges (u, v), (v, w) in G_k that form an angle of less than $\pi/3$ (see Figure 7). In the triangle (u, v, w) one of the angles $\angle wuv$ and $\angle vwu$ must be greater than $\pi/3$ (as the sum of all angles is π). Let w.l.o.g. $\angle wuv$ be greater than $\angle vwu$. From the law of sines it holds that

$$\frac{sin\angle uvw}{|uw|} = \frac{sin\angle wuv}{|vw|}$$

As $\angle wuv > \angle uvw$ and the sum of angles in the triangle is π, it follows that $sin\angle wuv > sin\angle uvw$ and so $|vw| > |uw|$. As G is a unit disk graph and (v, w) is an edge in G, there must be an edge (u, w) as well. So when the vertex v computes the spanning tree T for $N^k(v)$, the edge (u, w) can ensure that there is a path from v to w in T and therefore (v, w) is not part of T. This contradicts the assumption that (v, w) is part of G_k.

$\square$

Lemma 5. *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 $diam(C) \leq 33$.*

Proof. We employ a similar argument as used in Lemma 1. 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 $(7+3\epsilon)\cdot(2+\epsilon) = (14+13\epsilon+3\epsilon^2)$. So we can fit at most $\left\lfloor \frac{14+13\epsilon+3\epsilon^2}{\pi/4} \right\rfloor$ centers of non-overlapping discs of radius 1/2 in R. As $\epsilon \leq \frac{1}{128}$ we compute that $\left\lfloor \frac{14+13\epsilon+3\epsilon^2}{\pi/4} \right\rfloor = 17$. It follows that the cardinality of a maximum independent set in $G[R]$ is at most 17. 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 vertice in p we get an independent set in R. As the size of such a set is bounded by 17, the length of p is bounded by 33 and therefore $diam(C) \leq 33$. $\square$

Proof. (of Theorem 3): As $E' \subseteq E$ and Lemma 3 shows that Properties 4 through 7 hold for G_k, it follows that they hold for $G' = (V, E')$ as well.

Now we prove that *color* is a valid coloring for G', i.e. no two vertices which are connected by an edge in $G' = (V, E')$ have the same color. Let $e = (u, v) \in E'$ be an edge. We distinguish three cases.

- Case 1: u and v are in rectangles of different classes.
  - Case 1a: u or v is colored in color 1,2,3 or 4. W.l.o.g. u is colored in 1 or 2. As v is in a rectangle of a different class, it must hold that $color(v) \in \{3, 4, 5\}$ and therefore $color(u) \neq color(v)$.
  - Case 1b: u and v are colored in color 5. It follows that u and v must have an Euclidean distance of at most 1 to a glue in their respective row. As u and v must be in different rows and two glues of different rows are at least $2 + \epsilon$ away from each other, u and v cannot be connected by an edge which is a contradiction.
- Case 2: u and v are in *different* rectangles of the same type. As the rectangles have a width of at least $6 + 3\epsilon$ and heigth of at least $1 + \epsilon$, the rectangles which contain u and v must be adjacent. Then in the third step u and v were part of the same graph G_{glue}. As $e \in E'$, it follows that e must have been part of T_{glue}. In the coloring of T_{glue}, u and v have different colors (T1 and T2) and therefore $color(u) \neq color(v)$.
- Case 3: u and v are in *the same* rectangle. As $e \in E'$, in step 2, u and v were given different colors among 1,2,3 and 4. In step 3 one of them might have been colored in color 5. If both of them were colored with color 5, e would have been eleminated from E' which would be a contradiction.

This proves that *color* is a valid coloring for G'. From the algorithm it follows directly that $|color| \leq 5$ since only the colors $\{1, 2, 3, 4, 5\}$ are used in the coloring.

Now we want to prove that G' is connected. From Lemma 3 it follows that G_k is connected. For each rectangle R consider its connected components $\mathcal{C}(R)$ in G_k. As for each connected component C in a rectangle R a spanning tree is computed and in step 3 only edges between connected components in different rectangles are removed, it holds that the vertices in C are connected in G' as well. After step 2, all connected components $C \in \mathcal{C}(R)$ and $C' \in \mathcal{C}(R')$ which are connected by edges in G_k are connected in $G' = (V, E')$ since all edges between vertices in different rectangles were added to E'. Now consider an edge $e = (u, v) \in E(G_k)$ which connects two connected components $C \in \mathcal{C}(R)$ and $C' \in \mathcal{C}(R')$. If $e \in E'$ there is nothing to prove. If $e \notin E'$ then in step 3 a spanning tree T_{glue} for a subgraph G_{glue} was computed such that $e \in E(G_{glue})$ but $e \notin E(T_{glue})$. But as T_{glue} is a spanning tree, there is a path in T_{glue} between u and v. It follows that C and C' are connected in G'. So after step 3, all connected components $C \in \mathcal{C}(R)$ and $C' \in \mathcal{C}(R')$ which were connected by an edge in G_k are connected in G' as well. This proves that G' is connected.

Finally we prove that the algorithm is local. Consider a vertex v and let R be the rectangle which v is assigned to. In step 1, we only need to explore the vertices which are at most $k + 1$ hops away from v. In step 2, we need to explore the connected component in $G_k[R]$ which contains v. From Lemma 5 it follows that for this we need to explore the vertices which are at most 33 hops away from v and check if their adjacent edges are in G_k. So in this step we need to explore the vertices which are at most $33 + k + 1$ hops away from v. In step 3, we might need to explore a graph G_{glue} (if v is adjacent to vertices in another rectangle of the same class as R). The vertices $V(G_{glue})$ are at most one unit away from the same glue and they are all in the same row. Therefore, they all fit in a rectangle of size $2 \cdot (1+\epsilon)$ which would fit in one of the other rectangles. So in order to explore it, it is sufficient to explore the vertices which are at most $33+k+1$ hops away from v. So in the entire algorithm, we only need to explore the vertices which are at most $34+k$ hops away from v. □

6. Local construction of 4 colorable spanner

In this section we present a local algorithm for finding a 4-colorable spanner of a given unit disk graph. It computes both the spanner and the 4-coloring for it. It needs one color less than the algorithm presented in Section 5 but has a higher locality distance.

Like in Algorithm 6, for preprocessing the graph we employ the local algorithm presented in [5, 13]. This ensures that the resulting spanner is planar, it does not contain any angle smaller than $\pi/3$ and the degree of any node is at most 5. For k arbitrarily large this subroutine can also guarantees the weight of the spanner to be at most $\frac{k+1}{k-1}$ times the weight of a minimum spanning tree. The locality distance of the algorithm is in $O(k)$.

6.1. **Outline of the algorithm.**

The concept of this algorithm uses the methodology of Algorithm 6. However, we are going to use a different tiling of the plane and we will only use four colors to color the vertices of the computed spanner.

We consider a unit disk graph $G=(V,E)$. We compute a set $E' \subseteq E$ such that $G'=(V,E')$ forms a spanner for G. We also compute a coloring $color : V \to \{1,2,3,4\}$ for G'. The algorithm has three steps:

(1) A planar spanner G_k of G is created using the algorithm presented in [5, 13].
(2) The plane is divided into rectangles. A bipartite spanner for the vertices in each rectangle is computed and the vertices are colored with colors 1 and 2 (note that this is different than in Algorithm 6 since there we used four colors in the step which corresponds to this step).
(3) Collisions (two adjacent vertices with the same color) between vertices in different rectangles are being resolved by using two more colors.

6.2. **The algorithm.**

We consider a unit disk graph $G=(V,E)$. First we present the tiling of the plane which we are going to use. Then we present the three steps of the algorithm as outlined above. We will compute a set $E' \subseteq E$ such that $G'=(V,E')$ forms a spanner for G with certain properties. We will also compute a coloring $color$ for G' with $|color| \leq 4$.

6.2.1. *Tiling of the plane.*

The plane is divided into tiles of rectangles. Figure 8 shows one tile. A class number between 1 and 3 is assigned to each rectangle as shown in Figure 8. The whole plane is tiled with such tiles, starting at an arbitrary position. Figure 9 shows an extract of this. The width of each rectangle is $6+\epsilon$, the height of each rectangle is $3+\epsilon$ for any fixed ϵ with $0 < \epsilon \leq \frac{1}{128}$. Each vertex is assigned to the rectangle which contains it. Ambiguities caused by vertices on the edge of rectangles are being resolved by assigning them to the rectangle with the smallest class number which contains them (any other resolving method works as

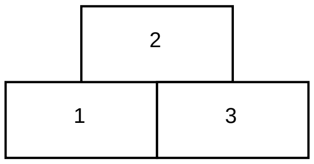

FIGURE 8. One tile for the tiling of the plane

well). We observe that two rectangles of the same class have a Euclidean distance of strictly more than three. So we conclude with the following propositions:

Proposition 5. *Two vertices in different rectangles of the same class are not adjacent.*

Proposition 6. *For a rectangle R let R^+ denote the area of R plus a surrounding belt of width one. Let R_1 and R_2 be two rectangles of the same class. Then there is no edge in G which connects two vertices in $G[R_1^+]$ and $G[R_2^+]$.*

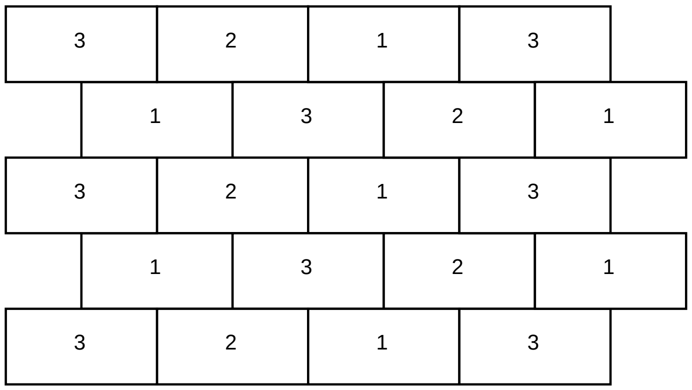

FIGURE 9. Tiling of the plane

Each vertex can compute the rectangle that it belongs to from its coordinates in the plane.

6.2.2. *Step 1: Computing the spanner G_k.* Like in Algorithm 6 we first fix an integer $k \geq 2$ and then run step 1 of Algorithm 6 in order to compute a spanner G_k. For a detailed description we refer to Section 5. This routine was originally presented in [5, 13].

6.2.3. *Step 2: Bipartite spanner for each rectangle.* We compute a set E' and a map $color : V \to \mathbb{N}$. Note that after this step the map *color* will not necessarily be a valid coloring for $G' = (V, E')$.

From now on, only edges that are part of G_k are considered. All other edges are ignored. Let R be a rectangle. Let $G[R]$ be the restriction of G to the vertices in R. For each connected component C in $G[R]$ we do the following: Compute a spanning tree T_C and a two-coloring $color_C$ for T_C which uses only colors 1 and 2. Assign all edges in T_C to E'. Define $color(v) := color_C(v)$ for all vertices v in C. Do this for all rectangles R which contain vertices of G. Finally we assign all edges to E' which connect vertices in different rectangles.

Step 2 of Algorithm 9: Finding a bipartite spanner for each rectangle

```
1 // Algorithm is executed independently by each node;
2 // Consider only edges from $G_k$, ignore all edges that are not in $G_k$;
3 Let $R$ be the rectangle which contains $v$;
4 Explore the connected component $C$ in $G[R]$ which contains $v$;
5 Compute a spanning tree $T_C$ for $C$;
6 Compute a coloring $color_C$ for $T_C$ which only uses colors 1 and 2;
7 Define $color(v) := color_C(v)$;
8 Add all edges of $T_C$ to $E'$;
9 Add all edges to $E'$ which connect vertices in $C$ to vertices in rectangles other
  than $R$;
```

6.2.4. *Step 3: Resolving collisions.* By a collision we denote an edge whose adjacent vertices have the same color. From the size of the rectangles in the tiling of the plane we conclude that such an edge must connect two vertices which are in adjacent rectangles (as the length of an edge is at most one). We first resolve all collisions where vertices in rectangles of class 1 are involved (step 3a). For that we need one additional color. Then we resolve collisions between vertices in rectangles of class 2 and 3 (step 3b). We use a fourth color for this.

We start with step 3a. Consider a rectangle R of class 1. Denote by V'' all vertices which are adjacent to at least one vertex in R. Denote by V' vertices in R which are adjacent to vertices in V''. Denote by $G_{coll}[R]$ the subgraph induced by $V' \cup V''$. For each connected component $C_{coll}[R]$ in $G_{coll}[R]$ we compute a spanning tree $T_{coll}[R]$. We compute a two-coloring $color_T$ for T_{coll}. Assume $color_T$ uses the colors $T1$ and $T2$. For all vertices v with $color_T(v) = T1$ we define $color(v) := 3$. Then we remove all edges $E(C_{coll}[R]) \setminus E(T_{coll}[R])$ from E'. Do this for all rectangles of class 1.

Resolving collisions between vertices in class 2 and 3 rectangles in step 3b works similarly: Consider a rectangle R of class 2. Denote by V'' all vertices in rectangles of class 3 which are adjacent to at least one vertex in R. Denote by V' vertices in R which are adjacent to vertices in V''. Denote by $G_{coll}[R]$ the subgraph induced by $V' \cup V''$. For each connected component $C_{coll}[R]$ in $G_{coll}[R]$ we compute a spanning tree $T_{coll}[R]$. We compute a two-coloring $color_T$ for $T_{coll}[R]$. Assume $color_T$ uses the colors $T1$ and $T2$. For all vertices v with $color_T(v) = T1$ we define $color(v) := 4$. Then we remove all edges $E(C_{coll}[R]) \setminus E(T_{coll}[R])$ from E'. Do this for all rectangles of class 2.

This ensures the connectivity of the spanner while only using four colors to color it. We summarize the whole algorithm in Algorithm 9.

6.3. **Proof of correctness.** We prove the correctness of Algorithm 9 and the properties of the computed spanner G' and the coloring *color* in Theorem 4.

Theorem 4. *Let $k \geq 2$ be the integer which was fixed at the beginning of Algorithm 9. For the computed spanner $G' = (V, E')$ and the computed coloring color it holds that*

(1) *G' is connected,*
(2) *color is a valid coloring for G',*
(3) *$|color| \leq 4$,*

Step 3 of Algorithm 9: Resolving collisions between vertices with the same color

```
// Step 3a;
forall rectangles $R$ of class 1 do
    compute $G_{coll}[R]$;
    forall connected components $C_{coll}[R]$ in $G_{coll}[R]$ do
        compute spanning tree $T_{coll}[R]$;
        compute coloring $color_T$ for $T_{coll}[R]$;
        // We assume that $color_T$ uses colors $T1$ and $T2$;
        forall vertices $v$ with $color_T(v) = T1$ do
            $color(v) := 3$;
        end
    end
    remove edges $E(C_{coll}[R]) \setminus E(T_{coll}[R])$ from $E'$;
end
// Step 3b;
forall rectangles $R$ of class 2 do
    compute $G_{coll}[R]$;
    forall connected components $C_{coll}[R]$ in $G_{coll}[R]$ do
        compute spanning tree $T_{coll}[R]$;
        compute coloring $color_T$ for $T_{coll}[R]$;
        // We assume that $color_T$ uses colors $T1$ and $T2$;
        forall vertices $v$ with $color_T(v) = T1$ do
            $color(v) := 4$;
        end
    end
    remove edges $E(C_{coll}[R]) \setminus E(T_{coll}[R])$ from $E'$;
end
```

Algorithm 9: Local Algorithm for computing a spanner and a 4-coloring for the spanner

```
// The Algorithm is executed independently by each node $v$;
Fix an integer $k \geq 2$;
Compute the spanner $G_k$ as stated in step 1 of Algorithm 6;
For all vertices $v$ compute $color(v)$ and compute the spanner $E'$ according to
    step 2;
For all vertices $v$ check whether $color(v)$ is changed in step 3 and what edges
    of $E'$ remain after step 3;
```

(4) *G' is planar,*
(5) $\triangle(G') \leq 5$,
(6) *no angle between two edges in G' is smaller than $\pi/3$,*
(7) *for a minimum spanning tree T for G it holds that $cost(G') \leq \frac{k+1}{k-1} \cdot cost(T)$ and*

(8) *the locality distance of Algorithm 9 is bounded by* $136 + k$*, i.e. Algorithm 9 is local.*

Before we can prove the theorem, we prove upper bounds for the diameter of a connected component in a rectangle R and the diameter of a graph $G_{coll}[R]$ which is computed in step 3. We observe that a graph $G_{coll}[R]$ fits into the area of R plus a surrounding belt of width one around it. Of course the same applies to a connected component in R.

Lemma 6. *Let* R^+ *be a rectangle plus a surrounding belt of width one around it. Let* $G[R^+]$ *the graph* G *restricted to* R^+*. For each connected component* C *in* $G[R^+]$ *it holds that* $diam(C) \leq 137$.

Proof. We employ a similar argument as used in Lemma 1. 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 $(9+\epsilon)\cdot(6+\epsilon) = (54+15\epsilon+\epsilon^2)$. So we can fit at most $\left\lfloor \frac{54+15\epsilon+\epsilon^2}{\pi/4} \right\rfloor$ centers of non-overlapping discs of radius 1/2 in R^+. As $\epsilon \leq \frac{1}{128}$ we compute that $\left\lfloor \frac{54+15\epsilon+\epsilon^2}{\pi/4} \right\rfloor = 68$. It follows that the cardinality of a maximum independent set in $G[R^+]$ is at most 68. 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 68, the length of p is bounded by 135 and therefore $diam(C) \leq 135$. □

Proof. (of Theorem 4): The properties 4 through 7 are inherited from the spanner G_k and for their proof we refer to the proof of Theorem 3.

As in step 2 only colors 1 and 2 are assigned to the vertices and in step 3 only colors 3 and 4 are assigned to vertices it follows that $|color| \leq 4$.

Now we want to prove that *color* is a valid coloring for G'. Let $e = (u,v) \in E'$ be an edge. We distinguish several cases:

- Case 1: u and v are in rectangles of different types.
  - Case 1a: u and v are in rectangles of class 1 and 2. Then e must have been part of a tree in step 3a, and therefore u and v must have been colored in different colors.
  - Case 1b: u and v are in rectangles of class 2 and 3. Then e must have been part of a tree in step 3b, and therefore u and v must have been colored in different colors.
  - Case 1c: u and v are in rectangles of class 1 and 3. Then e must have been part of a tree in step 3a, and therefore after step 3a u and v must have been colored in different colors. As one of the vertices is in a rectangle of type 1 (w.l.o.g. vertex v), it has not changed its color in step 3b. As in step 3b all recolored vertices are colored with color 4 it follows that after step 3b u has a different color than v.
- Case 2: u and v are in the same rectangle. So in step 1, u and v were colored in different colors.
  - Case 2a: At most one vertex among u and v was recolored in step 3. Then $color(u) \neq color(v)$ since all vertices which are recolored in step 3 are colored in colors 3 and 4.
  - Case 2b: Both u and v were recolored in step 3a. From Proposition 6 it follows that there is exactly only one rectangle R such that u and v

belong to $G_{coll}[R]$. Then $color(u) \neq color(v)$ since otherwise the edge e would have been removed from E'.
  - Case 2c: Both u and v were recolored in step 3b. Similar to case 2b.
  - Case 2d: One of the vertices was recolored in step 3a and the other one in step 3b. W.l.o.g assume u was recolored in step 3a and v was recolored in step 3b. Then $color(u) = 3 \neq 4 = color(v)$.
- Case 3: u and v are in different rectangles of the same class. From Proposition 5 it follows that they cannot be connected by an edge in G which is a contradiction.

Now we show that G' is connected. Denote by E'_i the set of edges which was computed after step i (so $E'_3 = E'$). From Lemma 3 follows that the spanner G_k is connected. Now let $e = (u, v)$ be an edge in E'_1. If $e \notin E'_2$ then there is still a path between u and v in $G'_2 = (V, E'_2)$ since in step 2 the edges of spanning trees for all connected components in all rectangles are assigned to E'_2 and all edges between different rectangles are assigned to E'_2. So it remains to show that the spanner does not get disconnected in step 3. Let $e = (u, v)$ be an edge in E'_2 . If in step 3a the edge e is not part of a graph $G_{coll}[R]$ it will not be removed and so there is still a connection between u and v in the spanner. Now assume that e is part of a connected component $C_{coll}[R]$ in a graph $G_{coll}[R]$. A spanning tree $T_{coll}[R]$ for $C_{coll}[R]$ is computed and all edges in $T_{coll}[R]$ remain in E' after step 3a. So after step 3a there is a path in the spanner between u and v. Since this holds for all edges $e \in E'_2$ the computed spanner is connected after step 3a. The same reasoning can be applied for step 3b. So in $G' = (V, E')$ there is a path between u and v. This proves that G' is connected.

Now we want to prove that Algorithm 9 is local. For computing whether an edge which is adjacent to a vertex $\bar{v}$ is in G_k we need to explore the vertices which are at most $k + 1$ hops away from $\bar{v}$. Now let v be a vertex in a rectangle R. In order to compute the color of v, in step 2 we need to explore the connected component of v in $G[R]$ and in step 3 we need to explore all connected components of graphs $G_{coll}[R']$ that v could possibly belong to. From Lemma 6 it follows that the diameters of the graphs $G[R']$ and $G_{coll}[R']$ are bounded by 135. So Algorithm 9 has a locality distance of $135 + k + 1 = 136 + k$. □

7. Impossibility results

In Sections 5 and 6 we presented local algorithms which computed 5 and 4 colorable spanners for a given unit disk graph and computed a coloring with the respective number of colors for the spanner. In this section we prove that this cannot be done locally for a bipartite spanner, even if we do not want to compute the coloring but only the spanner. We also prove that there is no local algorithm for 3-coloring unit disk graphs or 3-colorable spanners of unit disk graphs, even if the 3-colorability of the graph/spanner is guaranteed in advance. Finally we show a lower bound for the approximation ratio of a local algorithm for vertex coloring.

Theorem 5. *There is no local algorithm for computing connected bipartite spanners of unit disk graph.*

Proof. Assume on the contrary that there is a local algorithm $\mathcal{A}$ that computes a bipartite spanner for every unit disk graph G. Denote by k the locality distance of $\mathcal{A}$. Let $G = (V, E)$ be a unit disk graph with $2k + 3$ vertices that are distributed evenly in a circle such that all vertices have degree two (see Figure 10). Note that G is an odd cycle and therfore not bipartite.

Consider any edge $e = (u, v)$ and denote by v_e the vertex opposite to e (note that $2k + 3$ is odd and therefore v_e is uniquely defined). Now consider the subgraph G_e induced by the vertices $V \backslash \{v_e\}$ (see Figure 10). As $\mathcal{A}$ creates the bipartite spanner locally based on the information about the vertices which are at most k hops away from u and v, it must take the same decision for taking or not taking e into the spanner for both G and G_e. As not taking e for the spanner of G_e would result in a disconnected spanner for G_e, the edge e must be part of the spanners of both G and G_e.

By applying the same reasoning to all edges $e \in E$ it follows that $\mathcal{A}$ does not remove any edge from G when constructing the local spanner for it. But then $\mathcal{A}$ outputs G itself as its spanner which is an odd cycle and therefore not bipartite. This is a contradiction. □

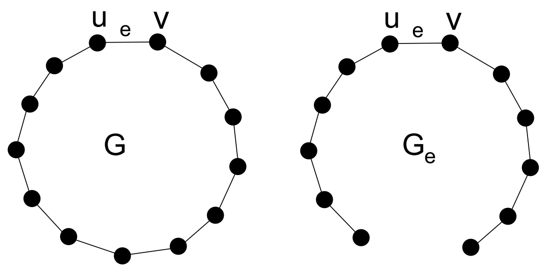

FIGURE 10. The graphs G and G_e for $k = 5$

Now we show that local algorithms cannot 3-color the vertices of a unit disk graph or a spanner of a unit disk graph.

Theorem 6. *There is no local algorithm for coloring 3-colorable unit disk graphs using at most 3 colors.*

Proof. Assume on the contrary that there is a local algorithm $\mathcal{A}$ that colors every three-colorable unit disk graph using at most three colors. Also assume that the color for each vertex v depends only on the vertices which are at most 2 hops away from v (i.e. the locality distance of $\mathcal{A}$ is 2). Consider the graphs G_1 and G_2 as shown in Figure 11. Consider the vertices which are labeled $u_1, u_2, v_1, v_2, w_1, w_2$. The 2-neighborhood of these vertices is the same in both G_1 and G_2. So $\mathcal{A}$ will color them with the same colors in G_1 and G_2. But once u_1, v_1 and w_1 are colored, the colors of the other vertices in both graphs are forced when only allowing three colors. In G_1 the vertices u_1 and v_2 must be colored in the same color, whereas in G_2 these vertices must be colored in different colors. As $\mathcal{A}$ colors the six indicated vertices with the same colors, either in G_1 or in G_2 there must be an edge whose adjacent vertices have the same color which is a contradiction. For any $k \in \mathbb{N}$ the graphs G_1 and G_2 can easily be enlarged so that they become counterexamples for any local algorithm that makes its decision based on the vertices that are at most k hops away from a given vertex. □

Corollary 1. *There is no local algorithm for coloring 3-colorable spanners of unit disk graphs with at most 3 colors.*

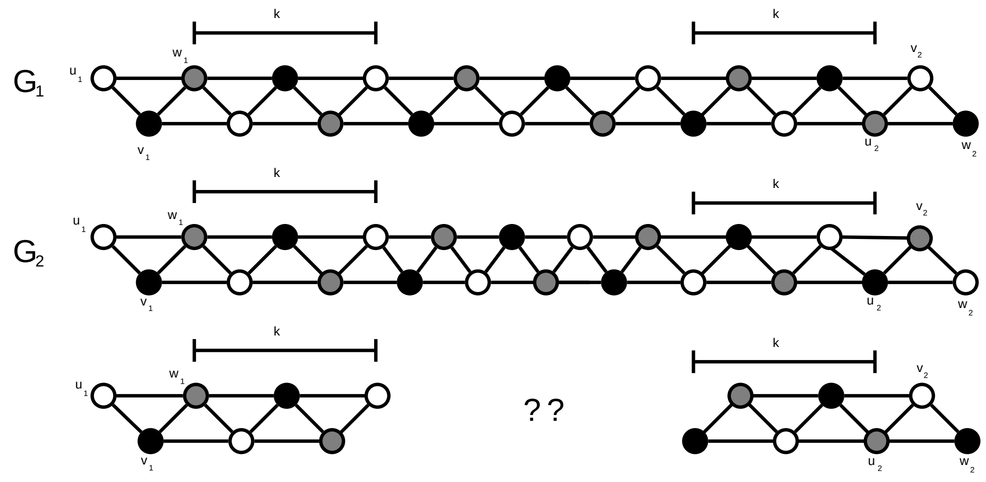

FIGURE 11. The graphs G_1 and G_2. On the bottom are the vertices which $u_1, u_2, v_1, v_2, w_1, w_2$ "see" when they explore the vertices 2 hops away from them in G_1 and G_2.

Corollary 2. *There is no local algorithm for coloring 3-colorable planar spanners of unit disk graphs with at most 3 colors.*

Now we give a lower bound for the approximation ratio which we can guarantee for a local approximation algorithm for vertex coloring. Our proof uses a similar technique as Linial in [14].

Theorem 7. *Let $\mathcal{A}$ be a local algorithm for vertex coloring. The approximation ratio of $\mathcal{A}$ is at least* $3/2$.

Proof. Let k be the locality distance of $\mathcal{A}$, i.e. the color of a vertex v depends only on the vertices which are at most k hops away from v. Let $G = (V, E)$ be a unit disc graph with $2k + 5$ vertices that are distributed evenly in a circle such that all vertices have degree two (see Figure 12). Note that G is an odd cycle and therfore not bipartite. It holds that $\chi(G) = 3$. Let *color* $: V \to \mathbb{N}$ be the coloring computed by $\mathcal{A}$.

Claim: There must be three vertices v_1, v_2, v_3 in a row in G such that $c(v_1) \neq c(v_2) \neq c(v_3) \neq c(v_1)$.

Proof of the claim: We take an arbitrary vertex v and its neighbor v'. As c is a valid coloring it holds that $c(v) \neq c(v')$. Now take the subgraph $\bar{G}$ induced by all vertices with colors $c(v)$ and $c(v')$. As G is not bipartite we know that $G \neq \bar{G}$. Let C be a connected component of $\bar{G}$ with at least two vertices (such a component must exist since v and v' are adjacent). As $G \neq \bar{G}$ there must be a vertex v_3 adjacent to C in G such that $c(v) \neq c(v_3) \neq c(v')$. We define two adjacent vertices in C where one of them is adjacent to v_3 as v_1 and v_2. This completes the proof of the claim.

We continue with the proof of the theorem. We construct a graph G' as follows: Starting with G, we identify the edge $e = (u, v)$ which is opposite to v_2 (note that G is an odd cycle and therefore e is well-defined) and remove u and v (see Figure 12). This implies that G' is bipartite. When coloring the vertices v_1, v_2, v_3, the algorithm $\mathcal{A}$ takes its decision based on the information of the vertices which are at most k hops away from them. So $\mathcal{A}$ must color the vertices v_1, v_2, v_3 with the same colors in G' and in G. Therefore, $\mathcal{A}$ colors G' with at least 3 colors even though G' is bipartite. □

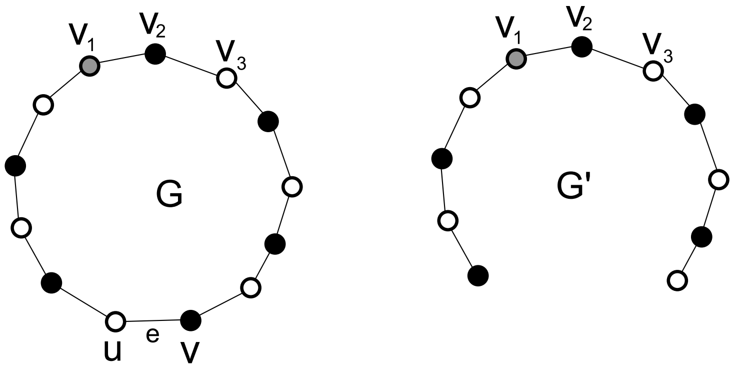

FIGURE 12. The graphs G and G' for $k = 4$

8. CONCLUSION

We presented a local approximation algorithm for the vertex coloring problem with polynomial processing time which achieves an approximation ratio of 4. Allowing exponential processing time we improved the approximation ratio to 3. The best known global polynomial time algorithm for this task achieves an approximation ratio of 3. It is an interesting open problem whether the approximation factor for local algorithms can be improved to less than 3. Also open is whether the approximation factor for local algorithms with polynomial processing time can be improved to less than 4. We showed that when the approximation factor of a local algorithm is given in the form $|color| \leq \alpha \cdot \chi(G)$ then α has to be at least $3/2$. Is it possible to prove or disprove that there are local approximation algorithms which color a unit disk graph G with at most $\beta \cdot \chi(G) + \gamma$ colors such that $\beta \leq 3/2$ and the constants being independent of the size of G?

We presented a local algorithm which computes k-partite spanners for unit disk graphs and k-colorings for these spanners in the case of $k = 5$ with a locality distance of 36. We also presented an algorithm which accomplishes this task for $k = 4$ with a locality distance of 138. It remains an open problem to improve these locality distances and to examine whether one can prove a lower bound for the locality of algorithms for these problems.

We showed that there is no local algorithm for the case of $k = 2$ even if we want to compute only the spanner and not the coloring for it. So our algorithm for $k = 4$ uses at most one color more than are being used by a local algorithm for the least possible k. It is an open question whether there is a local algorithm for $k = 3$. We also proved that no local algorithm can compute a 3-coloring for every 3-colorable unit disk graph. The same holds for 3-colorable spanners of unit disk graphs. So if there is a local algorithm for computing a 3-partite spanner and a 3-coloring for it the spanner must have more properties than just being 3-colorable globally.

References

[1] K. Appel and W. Haken. A proof of the four color theorem. *Discrete Math.*, 16:179–180, 1976.

[2] S. Arya, G. Das, D. M. Mount, J. S. Salowe, and M. Smid. Euclidean spanners: short, thin, and lanky. In ACM, editor, *Proceedings of the twenty-seventh annual ACM Symposium on Theory of Computing: Las Vegas, Nevada, May 29–June 1, 1995*, pages 489–498, pub-ACM:adr, 1995. ACM Press.

[3] P. Bose, J. Gudmundsson, and M. H. M. Smid. Constructing plane spanners of bounded degree and low weight. In Rolf H. Möhring and Rajeev Raman, editors, *ESA*, volume 2461 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2002.

[4] P. Bose, P. Morin, I. Stojmenovic, and J. Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[5] E. Chávez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Local construction of planar spanners in unit disk graphs with irregular transmission ranges. In José R. Correa, Alejandro Hevia, and Marcos A. Kiwi, editors, *LATIN*, volume 3887 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 2006.

[6] B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. *Discrete Math.*, 86(1-3):165–177, 1990.

[7] J. Czyzowicz, S. Dobrev, H. González-Aguilar, R. Kralovic, E. Kranakis, J. Opatrny, L. Stacho, and J. Urrutia. Local 7-coloring for planar subgraphs of unit disk graphs. *to appear*, 2007.

[8] D. Eppstein. Spanning trees and spanners. In Handbook of Computational Geometry, J.-R. Sack, and J. Urrutia, editors, *Elsevier*. 2000.

[9] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, 18:259–278, 1969.

[10] A. Gräf, M. Stumpf, and G. Weißenfels. On coloring unit disk graphs. *Algorithmica*, 20(3):277–293, 1998.

[11] X.-Y. Li, G. Calinescu, and P.-J. Wan. Distributed construction of planar spanner and routing for ad hoc wireless networks. In *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Society (INFOCOM-02)*, volume 3 of *Proceedings IEEE INFOCOM 2002*, pages 1268–1277, Piscataway, NJ, USA, June 23–27 2002. IEEE Computer Society.

[12] X.-Y. Li and Y. Wang. Efficient construction of low weight bounded degree planar spanner. In Tandy Warnow and Binhai Zhu, editors, *COCOON*, volume 2697 of *Lecture Notes in Computer Science*, pages 374–384. Springer, 2003.

[13] X.-Y. Li, Y. Wang, and W.-Z. Song. Applications of k-Local MST for Topology Control and Broadcasting in Wireless Ad Hoc Networks. *IEEE Transactions on Parallel and Distributed Systems*, 15(12):1057–1069, December 2004.

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

[15] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994.

[16] M. V. Marathe, H. Breu, H. B. Hunt III, S. S. Ravi, and D. J. Rosenkrantz. Simple heuristics for unit disk graphs. *Networks*, 25(1):59–68, 1995.

[17] R. H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In *Graphs and order (Banff, Alta., 1984)*, volume 147 of *NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci.*, pages 41–101. Reidel, Dordrecht, 1985.

[18] G. Narasimhan and M. Smid. Approximating the stretch factor of Euclidean graphs. *SIAM Journal on Computing*, 30(3):978–989, June 2000.

[19] R. Peeters. On coloring j-unit sphere graphs. Technical report, Department of Economics, Tilburg University, 1991.

[20] Y. Wang and X.-Y. Li. Localized construction of bounded degree and planar spanner for wireless ad hoc networks. In *DIALM-POMC*, pages 59–68. ACM, 2003.