Local Algorithms for Edge Colorings in UDGs

Iyad A. Kanj[1], Andreas Wiese[2], and Fenghui Zhang[3]

[1] School of Computing, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604, USA
ikanj@cs.depaul.edu
[2] Institute für Mathematik, Technische Universität Berlin, Germany
wiese@math.tu-berlin.de
[3] Department of Computer Science, Texas A&M University, College Station, TX 77843-3112, USA
fhzhang@cs.tamu.edu

Abstract. In this paper we consider two problems: the EDGE COLORING and the STRONG EDGE COLORING problems on unit disk graphs (UDGs). Both problems have important applications in wireless sensor networks as they can be used to model link scheduling problems in such networks. It is well known that both problems are NP-complete, and approximation algorithms for them have been extensively studied under the centralized model of computation. Centralized algorithms, however, are not suitable for ad-hoc wireless sensor networks whose devices typically have limited resources, and lack the centralized coordination.

We develop local distributed approximation algorithms for the EDGE COLORING and the STRONG EDGE COLORING problems on unit disk graphs. For the EDGE COLORING problem, our local distributed algorithm has approximation ratio 2 and locality 50. We show that the locality upper bound can be improved to 28 while keeping the same approximation ratio, at the expense of increasing the computation time at each node. For the STRONG EDGE COLORING problem on UDGs, we present two local distributed algorithms with different tradeoffs between their approximation ratio and locality. The first algorithm has ratio 128 and locality 22, whereas the second algorithm has ratio 10 and locality 180.

1 Introduction

The EDGE COLORING problem is to color the edges of a given graph G using the minimum number of colors so that no two edges of the same color are adjacent. The STRONG EDGE COLORING problem is to color the edges of a given graph G with the minimum number of colors so that no two edges with the same color are of distance less than 2. The EDGE COLORING and the STRONG EDGE COLORING problems are known to be NP-complete even on restricted classes of graphs [4,9]. Since both problems have numerous applications in networks where they model channel assignments/scheduling problems (see [1,2,3,7,11,12], among others), it is natural to seek approximation algorithms for them.

For the EDGE COLORING problem, Vizing's theorem [13] showed that any graph with maximum degree Δ has an edge coloring that uses at most $\Delta + 1$ colors;

C. Paul and M. Habib (Eds.): WG 2009, LNCS 5911, pp. 202–213, 2010.

however, his result was nonconstructive. Misra and Gries [10] gave a polynomial-time constructive proof of Vizing's theorem, thus showing that the problem can be approximated to within an additive constant of 1. Ramanathan [11] gave a very simple centralized greedy algorithm for the problem of ratio 2. Under the distributive model of computation, Gandam et al. [3] gave a distributed approximation algorithm based on Misra and Gries' [10] constructive proof of Vizing's theorem that approximates the problem to within an additive constant of 1. Kodialam and Nandagopal [7] gave a simple distributive algorithm of ratio 2, which was based on the centralized greedy algorithm of Ramanathan [11].

For the STRONG EDGE COLORING problem on planar graphs, Barrett el al. [1] gave a centralized algorithm that approximates the STRONG EDGE COLORING problem to ratio 17. This ratio has recently been improved to 2 by Ito et al. [5].

The assumed underlying graph model and the assumed computational model in the above results, however, do not seem appropriate for ad-hoc wireless sensor networks. In wireless sensor networks, devices can in principal communicate if they are in each other's transmission range. Therefore, a general graph model, or even a plane (embedded planar) graph model, is too flexible in the sense that it does not reflect the restrictions on the connectivity of such networks. Moreover, the topology of such networks undergoes constant change, and the devices in those ad-hoc networks have limited energy/power. Therefore, any assumed computational model should take into account the decentralized nature of such networks, and should be sensitive to issues such as scalability, robustness, and fault tolerance. In terms of the underlying graph model, when studying wireless sensor networks, it is natural to embed them in a Euclidean metric space. A common simple embedding assumes that the space is two dimensional, and that the transmission range of all devices is the same. In that case, the network is modeled as a *Unit Disk Graph*, abbreviated UDG henceforth, in the Euclidean plane: the nodes of the UDG correspond to the mobile wireless devices, and its edges connect pairs of nodes whose corresponding devices are in each other's transmission range equal to one unit. While this model is idealized, it has the advantage of being easier to work with. Meaningful theoretical and practical results can be derived under this model that, hopefully, will carry (at least partially) to more general models. Moreover, there are real examples where such models make sense: boats on water surfaces, vehicles in a relatively flat desert, etc.... In terms of the computational model, most of the above issues (scalability, robustness, fault tolerance) can be dealt with under the *local* distributed computational model, as defined by Linial [8]. A distributed algorithm is said to be *k-local* (where $k \geq 0$ is an integer) if the computation at each node of the graph depends solely on the initial state (in our case the ID and coordinates) of the nodes at distance (number of edges) at most k from the node (i.e., within k hops from the node). An algorithm is called *local* if it is k-local for some integer constant k. Efficient local distributed algorithms are naturally fault-tolerant and robust because faults and changes can be handled locally by such algorithms. These algorithms are also scalable because

the computation performed by a device is not affected by the total size of the network.

Local distributed algorithms for the EDGE COLORING problems on UDGs have been considered in [2]. However, the results in [2] only deal with a restricted subclass of UDGs called the "Yao-Like" subgraphs, and give an approximation algorithm for the EDGE COLORING problem within an additive constant of 1 from the optimal solution. For the STRONG EDGE COLORING problem on UDGs, we are only aware of the distributed approximation algorithm given by Barrett [1] which achieves an O(1) ratio; however, this algorithm is distributed but *not local*.

In this paper we develop local distributed approximation algorithms for the the EDGE COLORING and the STRONG EDGE COLORING problems on UDGs. For the EDGE COLORING problem, we present a local distributed algorithm of approximation ratio 2 and locality 50; this algorithm works for a generalization of UDGs, called *quasi-UDGs*. We show that the locality upper bound can be improved to 28, while keeping the same approximation ratio, at the expense of increasing the computation time at each node. For the STRONG EDGE COLORING problem on UDGs, we present two local distributed algorithms with different tradeoffs between their approximation ratio and locality. The first algorithm has approximation ratio 128 and locality 22, whereas the second algorithm has ratio 10 and locality 180.

2 Definitions and Notations

We assume familiarity with the basic graph-theoretic notations and terminologies.

Given a set of nodes S in the Euclidean plane, the Euclidean graph $\mathcal{E}$ on S is the complete graph whose node-set is S. The *unit disk graph*, shortly *UDG*, G on S is the subgraph of $\mathcal{E}$ with the same node-set as $\mathcal{E}$, and such that (u, v) is an edge of G if and only if $|(u, v)| \leq 1$, where $|(u, v)|$ is the Euclidean length of edge (u, v).

Let $0 < r \leq 1$ be a constant. The *quasi-UDG* on S with parameter r, is the subgraph G of $\mathcal{E}$ with the same node-set as $\mathcal{E}$, and such that for any two nodes u and v in G: if $|(u, v)| \leq r$ then $|(u, v)|$ is an edge of G, if $r < |(u, v)| \leq 1$ then (u, v) may or may not be an edge of G, and if $|(u, v)| > 1$ then (u, v) is not an edge of G. Clearly, a UDG is a quasi-UDG with $r = 1$.

Let H be a graph. We denote by $V(H)$ and $E(H)$ the set of nodes and the set of edges of H, respectively. The *length* of a path P in H, denoted $|P|$, is the number of edges in P. A *shortest path* between two nodes u and v in H is a path between u and v with the minimum length. A node v is said to be an *i-hop neighbor* of u in H, if the length of a shortest path between u and v in H is at most i. If u is an i-hop neighbor of v in H, we will say that the *hop distance* between u and v in H is at most i. For a node $u \in H$, and a natural number i, define $N_i[u]$ to be the set of i-hop neighbors of u in H.

For two edges e and e' in H, the *distance* between e and e' is the minimum length of a path, among all paths in H connecting an endpoint of e to an endpoint of e'. Two distinct edges are *adjacent* if their distance is 0, or equivalently, if they

share an endpoint. An *edge coloring* of H is an assignment of colors[1] to the edges in $E(H)$ such that no two adjacent edges in H are assigned the same color. A *strong edge coloring* of a graph H is an assignment of colors to the edges in $E(H)$ such that no two edges of distance at most 1 are assigned the same color. A *minimum edge coloring* of H is an edge coloring of H that uses the minimum number of colors. Similarly, a *minimum strong edge coloring* of H is a strong edge coloring of H that uses the minimum number of colors.

An *approximation algorithm* for a minimization problem $\mathcal{Q}$ is an algorithm that for each instance of $\mathcal{Q}$ computes a solution to the instance. The *ratio* of an approximation algorithm for a minimization problem is the maximum value, over all instances of the problem, of the size of the solution to the instance returned by the algorithm over the minimum-size solution to the instance.

The algorithms designed in this paper are k-local distributed algorithms. Each node in these algorithms starts by computing its k-hop neighbors, and performs only local computations afterwards. For a fixed k, it was shown in [6] that the k-hop neighborhoods of the nodes in a UDG (or a quasi-UDG) can be computed by a local distributed algorithm in which the total number of messages sent by all the nodes in the UDG is $O(n)$, where n is the number of nodes in the UDG. Therefore, the message complexity of each of the presented local distributed algorithms is $O(n)$.

3 Preliminaries

Let $\alpha > 2$ be a constant. Fix an infinite square tiling (i.e., a grid) $\mathcal{T}$ of the plane of tile dimensions $\alpha \times \alpha$.

Let T_1 be the translation with vector $(0, 0)$ (the identity translation), T_2 the translation of vector $(\alpha/2, 0)$ (horizontal translation), T_3 the translation of vector $(0, \alpha/2)$ (vertical translation), and T_4 the translation of vector $(\alpha/2, \alpha/2)$ (diagonal translation). We have the following simple lemma whose proof can be easily verified by the reader (note that $\alpha > 2$).

Fact 3.1. *Let G be a quasi-UDG, and let (u, v) be any edge in G. There exists a translation T in $\{T_1, T_2, T_3, T_4\}$ such that the translations of the nodes u and v under T, i.e., $T(u)$ and $T(v)$, reside in the interior of the same tile of $\mathcal{T}$.*

The following lemma uses a folklore packing argument to bound the length of a path between two nodes in a UDG that reside within a region of bounded area of the plane (see for example [14]).

Lemma 3.1. *Let G be a quasi-UDG of parameter $0 < r \leq 1$. Let H be a connected induced subgraph of G residing in a region R of the plane. Let R' be a region of area a' that contains R such that for any node p in R the disk centered at p and of radius $r/2$ is contained in R'. Then for any two nodes u and v of H, there exists a path in H between u and v of length at most $\lfloor 8a'/(\pi r^2) \rfloor$.*

[1] Without loss of generality, we shall assume that the colors are natural numbers.

4 Edge Coloring

In this section we present a local distributed algorithm that approximates the EDGE COLORING problem on quasi-UDGs which are a super class of UDGs. The idea behind the algorithm is to tile the plane as discussed in Section 3, and then to have the nodes residing in the same tile color the edges interior to their tile using the greedy algorithm given in [7,11]. This is a proper coloring since two edges contained in two distinct tiles are not adjacent. However, not every edge in the graph is interior to a tile because an edge may cross the horizontal or vertical (or both) boundary of a tile. To deal with this issue, we affect an appropriate set of translations to the nodes so that, for any edge in the graph, its translation under at least one of the translations is contained in some tile. This ensures that every edge of the graph will eventually be colored appropriately. Implementing this algorithm under a centralized model of computation is straightforward. However, implementing this algorithm under a localized distributed model poses some potential issues since the effect of the color of an edge over other edges needs to be limited, and some consensus problems need to be resolved.

We use the tiling $\mathcal{T}$ described in Section 3. Let G be a quasi-UDG with parameter r, where $0 < r \leq 1$. Each node $p \in G$ executes the algorithm **EdgeColoring-APX** given in Figure 1.

Lemma 4.1. *The algorithm* **EdgeColoring-APX** *is a k-local distributed algorithm, where $k = \lfloor (22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2) \rfloor$.*

1: p collects the coordinates of the nodes in $N_k[p]$ in G, where $k = \lfloor (22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2) \rfloor$
2: **for** round $i = 1, 2, 3, 4$ **do**
3: let $G_i(p)$ be a copy of the subgraph of G consisting of the set $E_i(p)$ of uncolored edges whose endpoints are in $N_k[p]$, and such that, for any edge $(u, v) \in E_i(p)$, $T_i(u)$ and $T_i(v)$ are in the same tile of $\mathcal{T}$
4: let $C_i^1(p), \ldots, C_i^\ell(p)$, where $\ell \geq 1$, be the connected components of $G_i(p)$
5: **for** $j = 1, \ldots, \ell$ **do**
6: p orders all the edges in $C_i^j(p)$ using the lexicographic order into the sequence of edges $E_i^j(p)$
7: **for** each edge e in $E_i^j(p)$ **do**
8: color e in $G_i(p)$ with the smallest available color, i.e., the smallest color that has not been used in the previous rounds to color any of the edges adjacent to e
9: **end for**
10: **end for**
11: **for** each edge $e \in G_i(p)$ incident on p **do**
12: p colors e in G with the same color in $G_i(p)$
13: **end for**
14: **end for**

Fig. 1. The algorithm **EdgeColoring-APX**

Proof. It is clear that the computation at each node depends solely on the coordinates of its k-hop neighbors, where $k = \lfloor (22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2) \rfloor$. □

For each $i \in \{1, 2, 3, 4\}$, let G_i be the subgraph of G consisting of the edges $(u, v) \in G$ such that $T_i(u)$ and $T_i(v)$ are in the same tile of $\mathcal{T}$; we call each connected component C in G_i an *i-cluster*, and we say that i is the *label* of C. Note that, by definition, any two distinct i-clusters are disjoint. A *cluster* is an i-cluster for some $i \in \{1, 2, 3, 4\}$. A sequence of clusters is said to be a *potential affecting sequence*, if the labels of the clusters on this sequence are strictly increasing, and each two consecutive clusters in the sequence are adjacent, i.e., share at least one node in G. Note that a potential affecting sequence of clusters has length at most 4. The notion of a potential affecting sequence will be used to confine the "effect" of the color of an edge on the color of another edge, as shown by the following lemma whose proof is omitted for lack of space:

Lemma 4.2. *Let $S = (C_1, C_2, C_3, C_4)$ be a potential affecting sequence of clusters (we allow C_i, $i \in \{1, 2, 3, 4\}$, to be empty). Then for any two nodes u and v in S, u is a k-hop neighbor of v in G, where $k = \lfloor (22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2) \rfloor$ and r is the parameter of the quasi-UDG G.*

Lemma 4.3. *The algorithm* **EdgeColoring-APX** *is an approximation algorithm of ratio 2 for the* EDGE COLORING *problem on quasi-UDGs.*

Proof. We first show that the algorithm computes an edge coloring of a given quasi-UDG G.

Let u be a node in G. By Fact 3.1, every edge incident on u belongs to one of the subgraphs $G_i(u)$, $i \in \{1, 2, 3, 4\}$, defined in line 3 of algorithm. Since u applies the greedy algorithm to the edges of $G_i(u)$ coloring an edge in $G_i(u)$ with a color that has not been used so far by an edge incident on it, node u will color its incident edges properly. Therefore, it suffices to show that for any edge (u, v), both u and v assign the *same* color to edge (u, v) to conclude that the coloring of G by the algorithm is consistent, and hence is an edge coloring of G.

For an edge $e \in G$, define $label(e)$ to be the minimum $i \in \{1, 2, 3, 4\}$ such that e is contained in an i-cluster. We say that an edge e *directly affects* another edge e' if e and e' are adjacent and either $label(e) < label(e')$ or $label(e) = label(e')$ and e comes before e' in the lexicographic order. We say that an edge e *affects* an edge e' if there exists an *affecting sequence* of edges $(e = e_0, e_1, \ldots, e_j = e')$ such that for $\ell = 0, \ldots, j-1$, e_ℓ directly affects $e_{\ell+1}$. Observe that the labels of the edges in any affecting sequence must be non-decreasing. Therefore, all the edges with the same label i in an affecting sequence form a connected subgraph of G, and hence are contained within a single i-cluster. It follows that, for any edge $e \in G$, any affecting sequence of edges containing e must be contained in some potential affecting sequence of clusters that contains e.

By looking at how the algorithm works, if the color of an edge e "influences" the color of an edge e', then edge e affects e'. For a potential affecting sequence S and an edge (u, v) in some cluster in S, both u and v in the algorithm collect the coordinates of all their k-hop neighbors, where $k = \lfloor (22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2) \rfloor$.

Therefore, by Lemma 4.2, both u and v have collected the coordinates of every node in S. It follows that both u and v must assign edge (u, v) the same color because both u and v have the coordinates of the endpoints of all edges affecting (u, v) and will color these edges in the same order using the same algorithm.

This shows that the algorithm computes a proper edge coloring of G.

To prove that the algorithm has approximation ratio 2, let apx_G be the number of colors used by the algorithm to color the edges of G, and let opt_G be the number of colors in a minimum edge coloring of G. Note that $opt_G \geq \Delta$, where Δ is the maximum degree of G. Let $e = (u, v)$ be the edge with the highest color number, i.e., $color(e) = \max_{e' \in E(G)} color(e')$. Let Δ_u and Δ_v be the degrees of nodes u and v. Since $color(e)$ is the smallest color number that is not used by any edge incident on u or v, it follows that $color(e) \leq (\Delta_u - 1) + (\Delta_v - 1) + 1$. Since e has the highest color number among all edges in G, we have $apx_G \leq (\Delta_u - 1) + (\Delta_v - 1) + 1 \leq 2 \cdot \Delta - 1 \leq 2 \cdot opt_G - 1$. □

Theorem 4.1. *The algorithm* **EdgeColoring-APX** *is a k-local distributed approximation algorithm for the* EDGE COLORING *problem on quasi-UDGs, where $k = \lfloor (22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2) \rfloor$, $0 < r \leq 1$ is the quasi-UDG parameter, and $\alpha > 2$ is a constant. For a UDG ($r = 1$), and by choosing α to be slightly larger than 2, the algorithm* **EdgeColoring-APX** *is a 50-local distributed approximation algorithm for* EDGE COLORING *of ratio 2.*

The above upper bound on the locality of the algorithm (i.e., k) can be improved by using smaller dimensions for the tiles; this will reduce the size of the region containing any affecting sequence, and hence decrease the upper bound on k. However, if we decrease the dimensions of the tiles, the above set of translations will no longer be sufficient to color all the edges in G (some edges may no longer reside in the interior of a tile under any of the above translations). To overcome this problem, we will need to use a family of translations, rather than a single translation, along each of the horizontal, vertical, and diagonal, directions. By fixing the dimensions of the tiles to be $(1 + \epsilon) \times (1 + \epsilon)$, where $\epsilon > 0$ is a constant, and picking an appropriate family of translations, we can prove that, in the worst case, any affecting sequence will be contained in a region whose area is at most $r^2 + (3\epsilon + 5)r + \epsilon^2 + 5\epsilon + 5$. This will give an upper bound of $8(r^2 + (3\epsilon + 5)r + \epsilon^2 + 5\epsilon + 5)/(\pi r^2)$ on k. In Table 1 we show the values of k corresponding to the values $\epsilon = 0.1, \ldots, 0.9$, and the asymptotic value of k when $\epsilon \to 0$. We note that, as ϵ decreases, the number of translations needed increases, and hence, the local computation time at the nodes increases.

Theorem 4.2. *For any constant $\epsilon > 0$, there exists a k-local distributed approximation algorithm of ratio 2 for the* EDGE COLORING *problem on quasi-UDGs,*

Table 1. Locality for different tile sizes

ϵ	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	$\to 0$
k	48	45	43	41	38	36	34	32	30	28

where $k = \lfloor 8(r^2+(3\epsilon+5)r+\epsilon^2+5\epsilon+5)/(\pi r^2)\rfloor$, *and* $0 < r \leq 1$ *is the quasi-UDG parameter.*

5 Strong Edge Coloring

In this section we present local distributed algorithms that approximate the STRONG EDGE COLORING problem on UDGs. Although the same approach used for the EDGE COLORING problem—in the previous section—works for the STRONG EDGE COLORING problem, this approach does not lead to good bounds on the locality of the algorithm. Therefore, we will adopt a different approach here. We note that the techniques in this section can be extended to quasi-UDGs; however, for simplicity, we restrict our attention to UDGs.

The local distributed algorithms we present use a centralized algorithm as a building block. We start by presenting this centralized algorithms.

5.1 The Centralized Algorithm

Barrett et al. [1] proposed a centralized greedy algorithm for approximating the STRONG EDGE COLORING problem on UDGs that works as follows. The nodes are first ordered using a lexicographic order. This lexicographic order on the nodes is used to induce a certain order on the edges (a bottom-up order). The edges are then considered with respect to this order, and an edge e is colored with the smallest color that has not been used to color any edge of distance at most 1 from e. If opt_G is the number of colors in a minimum strong edge coloring of G, then it was proved in [1] that the greedy algorithm computes a strong edge coloring of G that uses at most $8opt_G + 1$ colors. We will refer to the algorithm in [1] as the **Centralized-StrongEdgeColoring** algorithm.

We can show that, irrespective of the ordering in which the edges in G are considered, the algorithm **Centralized-StrongEdgeColoring** produces a strong edge coloring of G that uses at most $10opt_G$ colors. This property will be essential to bounding the approximation ratio of the algorithm we present in Subsection 5.3. The proof of this upper bound on the ratio is very similar to the proof given in [1] that the algorithm **Centralized-StrongEdgeColoring** has ratio $8opt_G + 1$ when the specific bottom-up ordering is used.

Theorem 5.1. *For any ordering $\mathcal{O}$ of the edges in G, the algorithm* **Centralized-StrongEdgeColoring**, *when it considers the edges in G with respect to the ordering $\mathcal{O}$, has approximation ratio 10.*

5.2 The Local Distributed Algorithm

In this subsection we present a local distributed algorithm that approximates the STRONG EDGE COLORING problem on UDGs. The approach is similar in flavor to the one used in Section 4. Using a different approach, we shall improve on the approximation ratio significantly at the expense of worsening the locality in Subsection 5.3.

Consider the same rectilinear tiling $\mathcal{T}$ of the plane discussed in Section 4 whose tiles are $\alpha \times \alpha$ squares, where $\alpha > 2$. We can label the tiles in $\mathcal{T}$ with the labels 1, 2, 3, 4, so that any two tiles with the same label are separated by at least one tile. We denote by $label(t)$ the label of a tile $t \in \mathcal{T}$.

Fact 5.1. *Let G be a UDG, and let e and e' be two edges in G such that the endpoints of e reside in the interior of a tile t, and the endpoints of e' reside in the interior of a tile t', where $t \neq t'$, and such that $label(t) = label(t')$. Then the distance between e and e' is at least 2.*

Proof. (Sketch) The statement follows from the facts that: (1) any two different tiles with the same label are separated by at least one tile, and (2) the dimension of a tile is greater than 1. □

Let T_1, T_2, T_3, and T_4, be the translations described in Section 4, and note that since $\alpha > 2$, Lemma 3.1 still holds true. Let C_i^1, C_i^2, C_i^3, and C_i^4, for $i = 1, 2, 3, 4$, be 16 mutually disjoint color classes. We assume that each of the color classes contains enough colors to color the edges of G, and that the colors in each class are ordered from smallest to largest.

Suppose that $\mathcal{A}$ is a centralized approximation algorithm of ratio $\rho_{\mathcal{A}}$ for the STRONG EDGE COLORING problem on UDGs. Intuitively, the algorithm can be summarized as follows. The algorithm runs in 4 rounds, each round corresponds to one of the above translations. Different color classes are used in different rounds to ensure that edges that are colored in different rounds do not conflict. In a given round i, translation T_i is applied to all the edges, and only the edges whose translations are interior to the tiles in $\mathcal{T}$ are colored as follows: the edges whose translations are in the same connected component of a tile of label j are colored with colors from class C_i^j, using the centralized algorithm $\mathcal{A}$. This ensures that edges whose translations end up in tiles of different labels are colored differently. Since different tiles of the same label are far enough from each other, and the centralized algorithm $\mathcal{A}$ is used to color the edges within the same tile, edges that are colored in the same round are colored properly.

More formally, each node p in G applies the algorithm **Strong-Edge-Coloring-APX** given in Figure 2.

Lemma 5.1. *The algorithm* **Strong-Edge-Coloring-APX** *is a k-local distributed algorithm, where $k = \lfloor 8(\alpha+1)^2/\pi \rfloor$.*

Lemma 5.2. *The algorithm* **Strong-Edge-Coloring-APX** *computes a valid strong edge coloring of G.*

Lemma 5.3. *The algorithm* **Strong-Edge-Coloring-APX** *approximates the* STRONG EDGE COLORING *problem on UDGs to a ratio $16 \cdot \rho_{\mathcal{A}}$, where $\rho_{\mathcal{A}}$ is the approximation ratio of $\mathcal{A}$.*

Proof. Let j be the round among the 4 rounds of the algorithm in which the maximum number of colors, apx_j, is used. It follows from the choice of j that the total number of colors used by the algorithm, call it apx_G, is at most $4 \cdot apx_j$.

1: p collects the coordinates of the nodes in $N_k[p]$ in G, where $k = \lfloor 8(\alpha+1)^2/\pi \rfloor$
2: **for** round $i = 1, 2, 3, 4$ **do**
3: p applies translation T_i and computes its virtual coordinates under T_i
4: if $T_i(p)$ is interior to some tile t_0 with label $\ell_0 \in \mathcal{T}$, where $\ell_0 \in \{1, 2, 3, 4\}$, p determines the set $S_i(p)$ of all the nodes in $N_k[p]$ whose translations under T_i reside in the same connected component as $T_i(p)$ in the interior of tile t_0; Let $H_i(p)$ be the subgraph of G induced by $S_i(p)$
5: p applies the algorithm $\mathcal{A}$ to the subgraph $H_j(p)$ to compute a strong edge coloring of $H_j(p)$, using only colors from the color class $C^j_{\ell_0}$, and starting with the smallest color in $C^j_{\ell_0}$; if an edge $e \in H_j(p)$ has already been colored in a previous round, p overwrites the previous color of e
6: **end for**

Fig. 2. The algorithm **Strong-Edge-Coloring-APX**

Let ℓ_j be the label of the color class from which the maximum number of colors, $apx^j_{\ell_j}$ is used in round j. Since there are 4 labels, it follows that $apx^j_{\ell_j} \leq 4 \cdot apx_j$, and hence, $apx_G \leq 16 \cdot apx^j_{\ell_j}$. Let opt_G be the number of colors in a minimum strong edge coloring of G.

From the way the algorithm works, in round j, every set of nodes S in G whose translations are in the same connected component in the interior of some tile with label ℓ_j, apply the algorithm $\mathcal{A}$ to compute a strong edge coloring of the edges of the subgraph of G induced by S, using the same set of colors $C^j_{\ell_j}$, and in the same order (all starting with the smallest color in $C^j_{\ell_j}$). Therefore, there exists a set of nodes S_j in G, whose translations reside in the same connected component in the interior of some tile, such that algorithm $\mathcal{A}$ uses $apx^j_{\ell_j}$ colors to properly color the edges of the subgraph H_j induced by S_j. Since $\mathcal{A}$ has approximation ratio $\rho_{\mathcal{A}}$, a minimum strong edge coloring of H_j requires at least $apx^j_{\ell_j}/\rho_{\mathcal{A}}$ colors. Since H_j is an induced subgraph of G, a minimum strong edge coloring of G requires at least $apx^j_{\ell_j}/\rho_{\mathcal{A}}$ colors. It follows that $opt_G \geq apx^j_{\ell_j}/\rho_{\mathcal{A}}$, and $16 \cdot apx_j \leq 16 \cdot \rho_{\mathcal{A}} \cdot opt_G$. This shows that the algorithm properly colors the edges of G using no more than $16 \cdot \rho_{\mathcal{A}} \cdot opt_G$ colors, and hence has ratio $16 \cdot \rho_{\mathcal{A}}$. □

Theorem 5.2. *There exists a 22-local distributed algorithm that, given a UDG G, computes a strong edge coloring of G using at most $128 \cdot opt_G + 16$ colors, where opt_G is the number of colors in a minimum strong edge coloring of G.*

Proof. Since a node p in the algorithm **Strong-Edge-Coloring-APX** can consider the edges in H_p in any order, p can order these edges according to the bottom-up ordering used in [1]. Under this specific ordering, as was mentioned before, the algorithm **Centralized-StrongEdgeColoring** computes a strong edge coloring of H_p using at most $8 \cdot opt_{H_p} + 1$ colors, where opt_{H_p} is the number of colors in a minimum strong edge coloring of H_p. Using the algorithm **Centralized-StrongEdgeColoring** as the subroutine $\mathcal{A}$ in the algorithm

Strong-Edge-Coloring-APX, and setting α to a value slightly larger than 2, the statement follows from Lemma 5.1, Lemma 5.2, and Lemma 5.3. □

5.3 The Improved Algorithm

In this subsection we present a local distributed algorithm for the STRONG EDGE COLORING problem on UDGs with a smaller approximation ratio, but larger locality, than the algorithm presented in Subsection 5.2. The algorithm uses the same tiling $\mathcal{T}$, but we require that $\alpha > 3$. The tiles are labeled with the labels $1, 2, 3, 4$ as in Subsection 5.2.

Each node is assigned to the tile which contains it. Ambiguities caused by nodes on the boundaries of tiles are resolved by assigning them to the tile with the smallest label which contains them (any other resolving method works as well). We observe that two tiles of the same label have a Euclidean distance more than 3. Therefore, if we place a bounding square box of dimensions $(\alpha + 1) \times (\alpha + 1)$ centered at each tile, two bounding boxes of two tiles with the same label have a Euclidean distance larger than 1. Consequently, two edges contained in different bounding boxes of two tiles with the same label have distance at least 2, and can be colored in the same round. The improved algorithm is given in Figure 3.

1: p collects the coordinates of the nodes in $N_k[p]$ in G, where $k = \lfloor (32\alpha^2 + 80\alpha + 40)/\pi \rfloor$
2: **for** round $i = 1, 2, 3, 4$ **do**
3: let $G_i(p)$ be a copy of the subgraph of G consisting of the set $E_i(p)$ of uncolored edges whose endpoints are in $N_k[p]$, and such that, for any edge $(u, v) \in E_i(p)$, u and v are in the bounding box of some tile of label i
4: p colors all the uncolored edges in $G_i(p)$ using the algorithm **Centralized-StrongEdgeColoring**
5: **for** each edge $e \in G_i(p)$ incident on p **do**
6: p colors e in G with the same color in $G_i(p)$
7: **end for**
8: **end for**

Fig. 3. The algorithm **Improved-StrongEdgeColoring-APX**

Lemma 5.4. *The algorithm is a k-local distributed algorithm, where $k = \lfloor (32\alpha^2 + 80\alpha + 40)/\pi \rfloor$, that computes a strong edge coloring of a given UDG.*

Lemma 5.5. *The algorithm is an approximation algorithm of ratio 10 for the* STRONG EDGE COLORING *problem on UDGs.*

Proof. By Lemma 5.4, the algorithm **Improved-StrongEdgeColoring-APX** is an approximation algorithm for the STRONG EDGE COLORING problem on UDGs. To prove that the algorithm has ratio 10, note that the algorithm **Improved-StrongEdgeColoring-APX** is equivalent to the algorithm **Centralized-StrongEdgeColoring** applied to the edges of G in the order they were colored by the algorithm **Improved-StrongEdgeColoring-APX**. It follows from Theorem 5.1 that the algorithm has ratio 10. □

Theorem 5.3. *Given a UDG G and a constant $\alpha > 3$, the algorithm* **Improved-StrongEdgeColoring-APX** *is a a k-local distributed algorithm, where $k = \lfloor (32\alpha^2 + 80\alpha + 40)/\pi \rfloor$, that computes a strong edge coloring of G using at most $10opt_G$ colors, where opt_G is the number of colors in a minimum strong edge coloring of G. By choosing α to be slightly larger than 3, the algorithm* **Improved-StrongEdgeColoring-APX** *is a 180-local distributed algorithm of ratio 10.*

References

1. Barrett, C., Kumar, V., Marathe, M., Thite, S., Istrate, G.: Strong edge coloring for channel assignment in wireless radio networks. In: PERCOMW 2006, pp. 106–110 (2006)
2. Czyzowicz, J., Dobrev, S., Kranakis, E., Opatrny, J., Urrutia, J.: Local edge colouring of yao-like subgraphs of unit disk graphs. In: Prencipe, G., Zaks, S. (eds.) SIROCCO 2007. LNCS, vol. 4474, pp. 195–207. Springer, Heidelberg (2007)
3. Gandham, S., Dawande, M., Prakash, R.: Link scheduling in sensor networks: distributed edge coloring revisited. In: INFOCOM, pp. 2492–2501 (2005)
4. Holyer, I.: The NP-completeness of edge-coloring. SIAM J. Comput. 10(4), 718–720 (1981)
5. Ito, T., Kato, A., Zhou, X., Nishizeki, T.: Algorithms for finding distance-edge-colorings of graphs. J. Discrete Algorithms 5(2), 304–322 (2007)
6. Kanj, I., Wiese, A., Zhang, F.: Computing the k-hop neighborhoods locally. Technical report # 08-007 at:
   `http://www.cdm.depaul.edu/research/Pages/TechnicalReports.aspx`
7. Kodialam, M., Nandagopal, T.: Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels. IEEE/ACM Trans. Netw. 13(4), 868–880 (2005)
8. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. 21(1), 193–201 (1992)
9. Mahdian, M.: On the computational complexity of strong edge coloring. Discrete Applied Mathematics 118(3), 239–248 (2002)
10. Misra, J., Gries, D.: A constructive proof of vizing's theorem. IPL 41 (1992)
11. Ramanathan, S.: A unified framework and algorithm for channel assignment in wireless networks. Wirel. Netw. 5(2), 81–94 (1999)
12. Ramanathan, S., Lloyd, E.L.: Scheduling algorithms for multihop radio networks. IEEE/ACM Trans. Netw. 1(2), 166–177 (1993)
13. Vizing, V.: On the estimate of the chromatic class of p-graphs. Diskret. Analiz 3, 25–30 (1964)
14. Wiese, A., Kranakis, E.: Local construction and coloring of spanners of location aware unit disk graphs. Technical report # 07-18 at,
   `http://www.scs.carleton.ca/~kranakis/Papers/TR-07-18.pdf`