Theoretical Computer Science 412 (2011) 4704–4714

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Local algorithms for edge colorings in UDGs☆

Iyad A. Kanj [a,*], Andreas Wiese [b], Fenghui Zhang [c]

[a] School of Computing, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604-2301, USA
[b] Institute für Mathematik, Technische Universität Berlin, Germany
[c] Google Kirkland, 747 6th Street South, Kirkland, WA 98033, USA

ARTICLE INFO

Article history:
Received 20 December 2010
Received in revised form 22 April 2011
Accepted 4 May 2011
Communicated by G. Ausiello

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. 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 [1–3,7,11,12], among others), it is natural to seek approximation algorithms for them.

For the EDGE COLORING problem, Vizing's theorem [13] shows that every graph with maximum degree Δ has an edge coloring that uses at most $\Delta + 1$ colors; however, his result is 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, Gandham 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].

☆ A preliminary version of this paper appears in proceedings of the *35th International Workshop on Graph-Theoretic Concepts in Computer Science* (WG), volume 5911 of *Lecture Notes in Computer Science*, (2009), pages 202–213, Springer.

* Corresponding author. Tel.: +1 312 362 5558; fax: +1 312 362 6116.
E-mail addresses: ikanj@cs.depaul.edu (I.A. Kanj), wiese@math.tu-berlin.de (A. Wiese), fhzhang@gmail.com (F. Zhang).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.05.005

For the STRONG EDGE COLORING problem on planar graphs, Barrett et 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 others 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 others transmission range equal to one unit. While this model seems too ideal, 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] deal only 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 aware only 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 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*. 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.

We simulated the algorithms in the current paper to get some data on how they perform in practice. The simulation results we obtained indicate that, empirically, the upper bounds on the ratio and the locality of these algorithms are much smaller than the theoretical upper bounds derived in the current paper. For example, for the EDGE COLORING problem, the simulations consistently showed an approximation ratio that is upper bounded by 1.2 and a locality that is upper bounded by 11. For the STRONG EDGE COLORING problem, the simulation results showed an approximation ratio that is upper bounded by 7.5 and a locality that is upper bounded by 10.

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. A *quasi-UDG* on S with parameter r is a 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 of 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 also 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

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

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 afterward. (We assume that each node knows its coordinates.) 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 fact:

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 $\mathcal{T}$.*

Proof. Note that $\alpha > 2$. If the edge (u, v) is interior to some tile in $\mathcal{T}$ then translation T_1 serves the purpose. If (u, v) crosses only the horizontal boundary of some tile in $\mathcal{T}$ then T_2 serves the purpose. If (u, v) crosses only the vertical boundary of some tile in $\mathcal{T}$ then T_3 serves the purpose. Finally, if (u, v) crosses both the horizontal and the vertical boundary of some tile in $\mathcal{T}$ then T_4 serves the purpose. □

The proof of 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:

Lemma 3.2. *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 $\lfloor 8a'/(\pi r^2) \rfloor$.*

Proof. Since u and v are connected in H, there exists a shortest path $P_{min} = (u = p_0, p_1, \ldots, p_\ell = v)$ between u and v in H. Let D_j, for $j = 0, \ldots, \ell$, be the disk centered at p_j and of radius $r/2$. By the hypothesis, all the disks D_j are contained in the region R'. Observe that by the minimality of P_{min}, the disks D_j for even j are mutually disjoint. Therefore, the area a of the region determined by the union of the disks D_j for even j is the sum of the areas determined by these individual disks. The value of a is precisely $(\pi r^2/4) \cdot \lceil \ell/2 \rceil$, and is at most the area of R'. It follows that $(\pi r^2/4) \cdot \lceil \ell/2 \rceil \leq a'$. Solving for the integer ℓ in the previous inequality we obtain $\ell \leq \lfloor 8a'/(\pi r^2) \rfloor$. This completes the proof. □

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 an edge coloring since two edges contained in the interior of 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 issues 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 Fig. 1. We intuitively describe the main steps performed by a node p in the algorithm. First, p starts by collecting the coordinates of its k-hop neighbors, where $k = \lfloor (22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2) \rfloor$; those are all the nodes that can contribute to the computation performed by p. Then p iterates over the four different translations described in the previous section; this ensures that all edges incident to p will be considered (by Fact 3.1). For each translation T_i, p considers a copy G_i of the subgraph consisting of the uncolored edges in its k-hop neighborhood whose endpoints reside in the same tile under the current translation. Since p has all the information of its k-hop neighborhood,

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**.

p applies a standard greedy (centralized) algorithm to color the uncolored edges in G_i. Finally, p "copies" only the coloring of its incident edges in G_i that it computed back to the original graph G. The correctness of the algorithm and its properties are proved below.

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

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 every two consecutive clusters in the sequence 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.

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.*

Proof. Consider the case when $C_i \neq \emptyset$, for $i = \{1, 2, 3, 4\}$, and note that the subgraph of G consisting of the edges in the sequence of clusters S forms a connected subgraph of G.

By definition, the edges in C_1 reside in some tile, say tile A depicted in Fig. 2. The edges in C_2 must reside in either the region $R_1 = A \cup B \cup C$, or the region $R_2 = A \cup E \cup F$. This is true because an edge in C_2 must cross a horizontal boundary of tile A (and only a horizontal boundary). Since the horizontal dimension of the tile is α, and the shift of the horizontal translation is $\alpha/2$, an edge in C_2 must reside in either R_1 or R_2. Note that, because each i-cluster, for $i = \{1, 2, 3, 4\}$, is contained in a single tile under translation T_i, if an edge $e \in C_2$ is in one of the two regions R_1, R_2, then all the edges in C_2 have to be in the same region. Therefore, by symmetry, we can assume that all the edges of C_2 reside in one of these two regions, say R_1. By the same token, the edges in C_3 are either contained in the region $R_3 = B \cup O \cup M \cup N$, or in the region $R_4 = C \cup D \cup I \cup J$, or in the region $R_5 = A \cup K \cup L$, or in the region $R_6 = A \cup G \cup H$. For each of the regions R_3, R_4, R_5, and R_6, we can check all the possible regions in which C_4 is contained. It can be easily verified that the worst case (region of maximum area) happens when C_3 is contained in R_4, and C_4 is contained in the region $D \cup J \cup P \cup Q$.

Therefore, when all the clusters C_i, $i \in \{1, 2, 3, 4\}$, are non-empty, we can assume, without loss of generality, that all the edges in the potential affecting sequence S are contained in a region whose shape is identical to the region $R = A \cup B \cup C \cup D \cup I \cup J \cup P \cup Q$, which is depicted as the blue region (light gray) in Fig. 2. This case (all the clusters are non-empty), as we argue a little bit later, corresponds to the worst upper bound on the value of k. We analyze this case next.

Let u and v be any two nodes in S. Since the subgraph G' of G induced by the edges in S is connected, u, v are connected within region R. Observe that, for any point in R, the disk centered at the point and of radius $r/2$ is contained within the region R' consisting of R plus a bounding frame of thickness $r/2$, depicted as the green (dark gray) region in Fig. 3. It is straightforward to verify that the area a' of R' is $11\alpha^2/4 + r^2 + 4\alpha r$. By Lemma 3.2, the hop distance between u and v is at most $(22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2)$.

The case considered above (all C_i's are non-empty) corresponds to the worst upper bound on k. We list below the regions obtained for all other cases.

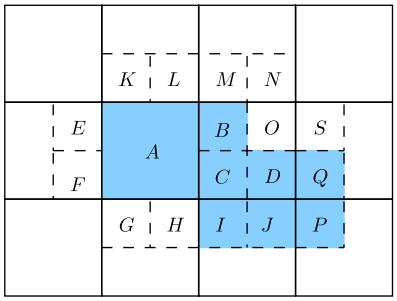

Fig. 2. Illustration for the worst-case region containing an affecting sequence.

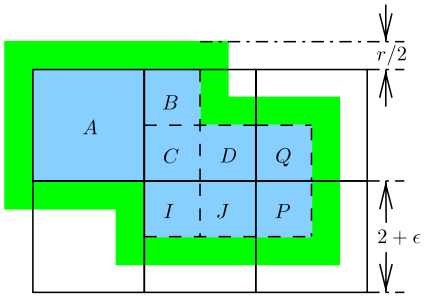

Fig. 3. Illustration for the bounding frame.

If exactly three of the clusters C_i ($i \in \{1, 2, 3, 4\}$) are empty, then S is contained within a region that is identical to region A in Fig. 2. If only C_1 and C_2 are empty, then the region obtained is identical to region $A \cup B \cup C$. If only C_1 and C_3 are empty, then the region is identical to region $A \cup G \cup H$. If only C_1 and C_4 are empty, then the region is identical to region $O \cup S \cup D \cup Q \cup I \cup J$. If only C_2 and C_3 are empty, then the region is identical to region $A \cup I \cup C \cup H$. If only C_2 and C_4 are empty, then the region is identical to region $A \cup G \cup H$. If only C_3 and C_4 are empty, then the region is identical to region $A \cup B \cup C$. If only C_1 is empty, then the region is identical to region $A \cup B \cup C \cup D \cup I \cup J \cup P \cup Q$. If only C_2 is empty, then the region is identical to region $A \cup G \cup C \cup H \cup I$. If only C_3 is empty, then the region is identical to region $A \cup B \cup C \cup H \cup I$. If only C_4 is empty, then the region is identical to region $A \cup B \cup C \cup D \cup I \cup J$.

Each of the above listed cases corresponds to an area R' that is not larger than that when all the sequences are non-empty, and hence, to a value of k that is not larger than $(22\alpha^2 + 8r^2 + 32\alpha r)/(\pi r^2)$. This completes the proof. □

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 $apx_G \leq (\Delta_u - 1) + (\Delta_v - 1) + 1 \leq 2 \cdot \Delta - 1 \leq 2 \cdot opt_G - 1$. □

The theorem below follows from Lemmas 4.1–4.3 above:

Theorem 4.4. *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.*

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. 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 algorithm that we present uses a centralized algorithm as a building block. We start by presenting this centralized algorithm.

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 will show next 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 Section 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.

Proof. Consider the algorithm **Centralized-StrongEdgeColoring** with respect to an arbitrary ordering on the edges of G. Let $e = (u, v)$ be the edge in G with the maximum color number. Let D_u and D_v be the disks centered at u and v, respectively, and of radius 1. The region $D_u \cup D_v$ can be partitioned in the worst case (when $|(u, v)| = 1$) into at most 10 sectors/triangles, as shown in Fig. 4, each of diameter (i.e., longest distance between any two points in the sectors/triangles) at most 1. Therefore, the nodes in G in any of the sectors/triangles form a clique. Note that any edge in G within distance 1 from e must have at least one endpoint in one of these sectors/triangles. Moreover, all edges with at least one endpoint in the same sector/triangle must all be colored differently in any strong edge coloring of G. If the color of e is equal to the value apx_G, then apx_G many colors were used by the algorithm to color the edges of distance at most 1 from e. At least $apx_G/10$ of these edges have one of their endpoints in the same sector/triangle, and hence must receive different colors in any strong edge coloring of G. In particular, a minimum strong edge coloring of G must use at least $apx_G/10$ many colors. It follows that the algorithm **Centralized-StrongEdgeColoring** computes a strong edge coloring of G that uses at most $10opt_G$ many colors. □

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 in Section 5.3, we shall improve on the approximation ratio significantly at the expense of worsening the locality.

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 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; see Fig. 5 for an illustration. We denote by $label(t)$ the label of a tile $t \in \mathcal{T}$.

Fact 5.2. *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.

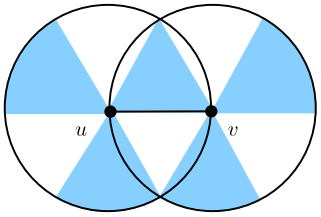

Fig. 4. Illustration of the disks D_u and D_v and their worst-case partitioning into 10 sectors.

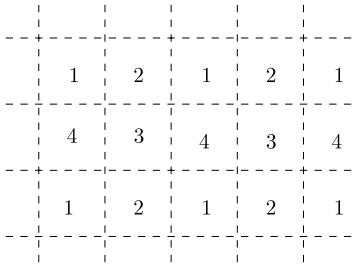

Fig. 5. Illustration of the tiling $\mathcal{T}$.

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 $H_i(p)$ to compute a strong edge coloring of $H_i(p)$, using only colors from the color class $C^i_{\ell_0}$, and starting with the smallest color in $C^i_{\ell_0}$; if an edge $e \in H_i(p)$ has already been colored in a previous round, p overwrites the previous color of e
6: **end for**

Fig. 6. The algorithm **Strong-Edge-Coloring-APX**.

Proof. 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 2. □

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^1_i, C^2_i, C^3_i, and C^4_i, 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^j_i, 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.

Each node p in G applies the algorithm **Strong-Edge-Coloring-APX** given in Fig. 6. We intuitively describe the steps performed by the algorithm. Recall that the plane is tiled with tiles of labels $\{1, 2, 3, 4\}$, and that different tiles with the same label are of distance at least 2. Note also that the sixteen color classes $C^j_i, i, j = 1, 2, 3, 4$, are pairwise disjoint. This will ensure that the coloring is a proper coloring. Each node p starts by collecting the coordinates of its k-hop neighborhood, where $k = \lfloor 8(\alpha+1)^2/\pi \rfloor$. Node p then iterates over the four different translations. For each translation T_i, p computes its translation under T_i, $T_i(p)$. Suppose that $T_i(p)$ resides in a tile of label ℓ_0. Node p considers all nodes in its k-hop neighborhood whose translations under T_i reside in the same tile as $T_i(p)$, and applies a centralized algorithm to color the edges of the graph induced by those points using the color class $C^i_{\ell_0}$; this ensures that for different translations and different tile labels, disjoint color classes are used, and thus guarantees that the coloring is proper. The correctness and properties of the algorithm are proved below.

Lemma 5.3. *The algorithm* **Strong-Edge-Coloring-APX** *is a k-local distributed algorithm, where $k = \lfloor 8(\alpha+1)^2/\pi \rfloor$.*

Proof. This follows from the fact that the computation at each node in the algorithm depends only on the coordinates of its k-hop neighbors, where $k = \lfloor 8(\alpha+1)^2/\pi \rfloor$. □

Lemma 5.4. *The algorithm* **Strong-Edge-Coloring-APX** *computes a valid strong edge coloring of G.*

Proof. Let p be a node in G, and consider the set $\Gamma_i(p)$ of all nodes in G whose translations under T_i reside in the same tile as $T_i(p)$ (note that $p \in \Gamma_i(p)$). Since a translation is isometric, there exists a square R in the plane of dimensions $\alpha \times \alpha$ containing all the nodes in $\Gamma_i(p)$. Let R' be the square whose center coincides with that of R and of dimensions $(\alpha+1) \times (\alpha+1)$. Then the area a' of R' is $(\alpha+1)^2$, and for any point in R, the disk centered at the point and of radius $1/2$ is contained in R'. By Lemma 3.2, any two nodes in $\Gamma_i(p)$ are $k = \lfloor 8(\alpha+1)^2/\pi \rfloor$-hop neighbors in G. This shows that for any node p and a translation T_i, all nodes in $S_i(p)$ will apply the same centralized algorithm to the edges in $H_i(p)$, and hence each edge in $H_i(p)$ will be assigned the same color by all the nodes in $S_i(p)$. Since in round i, only the nodes in $S_i(p)$ can color the edges of $H_i(p)$, every edge in $H_i(p)$ will be colored in round i. By Lemma 3.1, every edge of G will be colored in some round. It suffices, therefore, to show that the algorithm colors the edges of G properly. Let e and e' be two edges in G of distance at most 1. We will show that e and e' are colored with different colors at the end of the algorithm.

Proceed by contradiction. Assume that e and e' are assigned the same color at the end of the algorithm. Since for any two rounds j and j', where $j \neq j'$, the color classes used in round j are mutually disjoint from those used in round j', edges e and e' must have been assigned their color in the same round; let this round be round j. In addition, the translations of the endpoints of e in round j must reside in the interior of a tile with the same label as that in which the translations of the endpoints of e' reside. Since a translation is isometric, and since the distance between e and e' is at most 1 in G, by Fact 5.2, it follows that the translations of the endpoints of e and e' reside in the same tile t. Since the distance between e and e' in G is at most 1, the translations of the endpoints of e and e' reside in the same connected component in tile t. This, however, is a contradiction because the algorithm $\mathcal{A}$, applied in round j in the algorithm **Strong-Edge-Coloring-APX**, computes a strong edge coloring of the edges whose translations reside in the same connected component of tile t that contains e and e'. □

Lemma 5.5. *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 $4 \cdot apx_j$. 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 5.6. *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 Lemmas 5.3–5.5. □

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 Section 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 Section 5.2 (see Fig. 5).

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 Fig. 7. The algorithm is somehow similar to the algorithm **EdgeColoring-APX**, except that the current algorithm uses bounding boxes instead of

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. 7. The algorithm **Improved-StrongEdgeColoring-APX**.

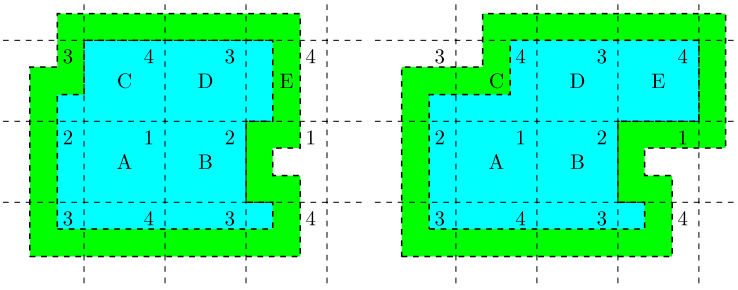

Fig. 8. In the worst case an affecting sequence starts from a type-1 tile, goes through a type-2 tile, then a type-3 tile, and finally ends up in a type-4 tile. Therefore, in the worst case, an affecting sequence is contained within one of the two regions depicted above.

translations to ensure coordination. We intuitively describe the main steps performed by a node p in the algorithm. First, p starts by collecting the coordinates of its k-hop neighbors, where $k = \lfloor(32\alpha^2 + 80\alpha + 40)/\pi\rfloor$; those are all the nodes that can contribute to the computation performed by p. Then p iterates for four iterations $i = 1, 2, 3, 4$. For each iteration i, p considers a copy G_i of the subgraph consisting of the uncolored edges in its k-hop neighborhood whose endpoints reside in the bounding box of some tile of label i. Node p applies a centralized algorithm to color the uncolored edges in G_i. Finally, p "copies" only the coloring of its incident edges in G_i that it computed back to the original graph G. The correctness of the algorithm and its properties are proved below.

Lemma 5.7. *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.*

Proof. Let G be a UDG. First, since the computation at each node in the algorithm **Improved-StrongEdgeColoring-APX** depends only on the coordinates of its k-hop neighbors, where $k = \lfloor(32\alpha^2 + 80\alpha + 40)/\pi\rfloor$, the algorithm is a k-local distributed algorithm for the aforementioned value of k. Therefore, it suffices to show that the algorithm computes a strong edge coloring of G.

For each $i \in \{1, 2, 3, 4\}$, let G_i be the subgraph of G consisting of all the edges $(u, v) \in G$ such that u and v are in the bounding box of the same tile of label i in $\mathcal{T}$; 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 of distance at least 2, that is, are disjoint and there is no edge connecting a node in the first i-cluster to a node in the second i-cluster. 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 of distance at most 1, i.e., either they share at least one node in G or there exists a node in the first cluster that is adjacent to a node in the second cluster. Note that a potential affecting sequence has length at most 4.

For an edge $e = (u, v)$, we define *label*(e) to be the smallest $i \in \{1, 2, 3, 4\}$ such that u or v (or both) resides in a tile of label i. Now that the notion of the label of an edge has been defined, we can define the notion of the affecting sequence of edges, and subsequently, of an edge affecting another edge, in a similar fashion to that in the proof of Lemma 4.3. The only difference now is that the distance between two consecutive edges in the sequence can be at most 1. Similarly, it can be proved that any affecting sequence of edges is contained in some potential affecting sequence of clusters.

Now we show that any two nodes u and v in a potential affecting sequence of clusters are at most k hops away from each other, where $k = \lfloor(32\alpha^2 + 80\alpha + 40)/\pi\rfloor$. Let $S = (C_1, C_2, C_3, C_4)$ be a potential affecting sequence of clusters. Similarly to the proof of the Lemma 4.2, it can be shown that the worst upper bound on k corresponds to the case where all the C_i's are non-empty. Note that in round 1, all edges within the blue bounding boxes around any type-1 tile are colored, i.e., any edge with at least one vertex in a type-1 tile has been colored in round 1. Hence, in an affecting sequence, the blue bounding boxes around a type-2 tile do not extend to the neighboring type-1 tiles. Similarly the blue bounding boxes around a type-3 tile in an affecting sequence do not extend to the neighboring type-1 or type-2 tiles, and a bounding box for a type-4 tile does not extend to any of the neighboring tiles.

Assuming the worst case, there are two possible regions for the bounding box of an affecting sequence; they are depicted in Fig. 8. The case when the shape contains tiles $\{A, B, D, E\}$ is the worst case because the area is larger. In that case the

affecting sequence must reside in a region R whose shape is depicted as the blue (dark gray) region in the right figure of Fig. 8.

By placing R within a frame of thickness 1/2, we obtain a region R' depicted as the green (dark gray) plus the blue (light gray) regions in the right figure of Fig. 8 of area $a' = 4\alpha^2 + 10\alpha + 5$. Using Lemma 3.2, any two nodes in R are at most $k = \lfloor (32\alpha^2 + 80\alpha + 40)/\pi \rfloor$ hops away from each other.

To prove that the algorithm computes a strong edge coloring of G, observe that every edge of G must be contained in at least one bounding box, and hence will be considered by the algorithm. Let $e = (u, v)$ and $e' = (u', v')$ be two edges in G, and assume, without loss of generality, that e' affects e. Since e' affects e, any potential affecting sequence of clusters for e' is contained in some potential affecting sequence of clusters for e. Since every node in the algorithm collects the coordinates of all its k-hop neighbors where $k = \lfloor (32\alpha^2 + 80\alpha + 40)/\pi \rfloor$, from the above discussion it follows that both nodes u and v know the coordinates of all nodes in any potential affecting sequence of clusters of both e' and e. Consequently, nodes u and v will assign the edge e' the same color assigned by its endpoints u' and v'. It follows from the previous argument that, when u and v color the edge e, both u and v will assign e the same color, and will assign it a color that does not conflict with the color of any edge within distance at most 1 from e. This shows that the algorithm computes a strong edge coloring of G. □

Lemma 5.8. *The algorithm is an approximation algorithm of ratio* 10 *for the* STRONG EDGE COLORING *problem on UDGs.*

Proof. By Lemma 5.7, 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 **Improved-StrongEdgeColoring-APX** has ratio 10. □

The theorem below follows directly from the above two lemmas:

Theorem 5.9. *Given a UDG G and a constant $\alpha > 3$, the algorithm* **Improved-StrongEdgeColoring-APX** *is a k-local distributed algorithm, where $k = \lfloor (32\alpha^2 + 80\alpha + 40)/\pi \rfloor$, that computes a strong edge coloring of G which uses at most $10opt_G$ colors, where opt_G is the number of colors in a minimum strong edge coloring of G. In particular, by choosing α to be slightly larger than* 3*, the algorithm* **Improved-StrongEdgeColoring-APX** *is a* 180*-local distributed algorithm of ratio* 10*.*

6. Empirical results and discussions

We conducted extensive simulations to study the behavior of the presented algorithms empirically.

We randomly deployed N UDG nodes in a square sensor field. The average degree d of the networks generated is ranged over the set of values {5, 7, 9, 11, 13}. In order to achieve the expected average degree, we adjusted the area of the sensor field appropriately. For each configuration (N, d), we generated 100 networks and ran the algorithms on them. For each run of the algorithms, we measured the maximum locality bound, and we upper bounded the approximation ratio. Then we averaged these values over the 100 generated networks.

For the EDGE COLORING problem, note that the maximum node degree of the network G is a lower bound on the number of colors in a minimum edge coloring of G. We upper bounded the actual approximation ratio of the algorithm **EdgeColoring-APX** by dividing the number of colors used by the algorithm by the maximum node degree in G. For the STRONG EDGE COLORING problem, observe that the number of edges adjacent to an edge in G, denoted by n_e, is a lower bound on the number of colors in a minimum strong edge coloring of G. Therefore, we can use the value $max\{n_e \mid e \in G\}$ as a lower bound on the number of colors in a minimum strong edge coloring of G. By dividing the number of colors used in the algorithms for the STRONG EDGE COLORING problem by $max\{n_e \mid e \in G\}$, we obtain an upper bound on the approximation ratio of the algorithms.

Fig. 9 shows a typical set of simulation results for the approximation ratio and the locality of the algorithms. The curves in these two figures correspond to the results when $N = 5000$. We remark that we did not notice much change in the results for different values of N. This is possibly due to the fact that the algorithm are local.

For the **EdgeColoring-APX** algorithm, the theoretical upper bound on the approximation ratio derived in the current paper is 2, whereas in the empirical results the ratio was always upper bounded by 1.2. On the other hand, the theoretical upper bound on the locality of the algorithm is 50, whereas the simulations showed that this number is smaller than 11, even when the diameter of the network is as large as 136.

For the **ImprovedStrongEdgeColoring-APX** algorithm, the theoretical upper bounds on the locality seem to be impractical (180). However, the simulation results show that the locality of the algorithm is almost always upper bounded by 37. On the other hand, the approximation ratio in the simulation results was always upper bounded by 4.5, which is also much smaller than the derived theoretical upper bound of 10.

Surprisingly, the **StrongEdgeColoring-APX** algorithm performed very well empirically. Although the theoretical upper bound on the approximation ratio is as large as 128, the empirical results showed that the approximation ratio of the algorithm is smaller than 7.5. The locality of the algorithm is always less than 9.5, in spite of the obtained theoretical upper bound of 22. This suggests that the **StrongEdgeColoring-APX** algorithm may be more suitable for practical use when compared to the **ImprovedStrongEdgeColoring-APX** algorithm, as the trade off between its approximation ratio and locality is more reasonable and favorable.

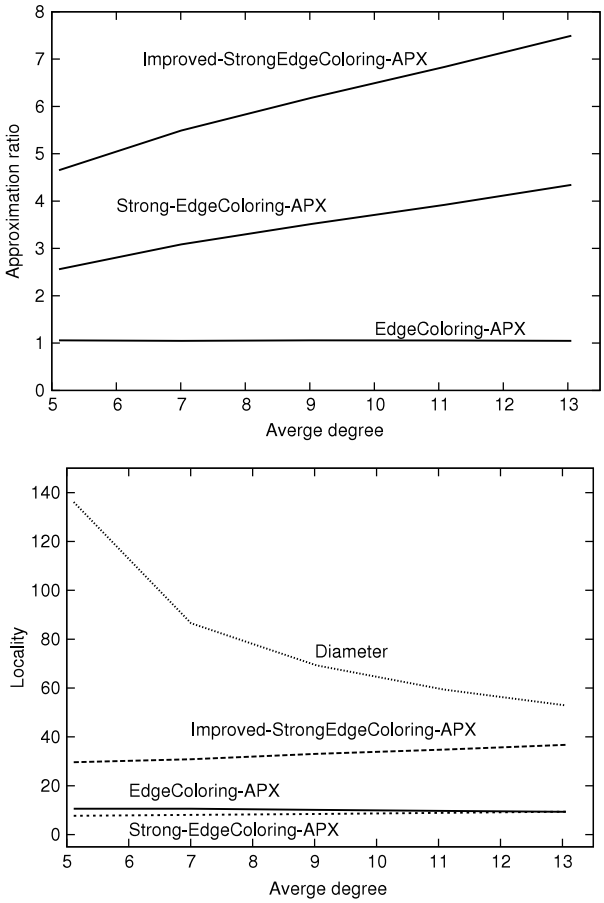

Fig. 9. The simulation results.

We conclude that the algorithms we presented in the current paper are not only of theoretical importance, but are also of practical relevance.

Acknowledgement

The first author was supported in part by a DePaul University Competitive Research Grant.

References

[1] C. Barrett, V. Kumar, M. Marathe, S. Thite, G. Istrate, Strong edge coloring for channel assignment in wireless radio networks, in: PERCOMW'06, 2006, pp. 106–110.
[2] J. Czyzowicz, S. Dobrev, E. Kranakis, J. Opatrny, J. Urrutia, Local edge colouring of yao-like subgraphs of unit disk graphs, in: SIROCCO, in: Lecture Notes in Computer Science, vol. 4474, Springer, 2007, pp. 195–207.
[3] S. Gandham, M. Dawande, R. Prakash, Link scheduling in sensor networks: Distributed edge coloring revisited, in: INFOCOM, 2005, pp. 2492–2501.
[4] I. Holyer, The NP-completeness of edge-coloring, SIAM J. Comput. 10 (4) (1981) 718–720.
[5] T. Ito, A. Kato, X. Zhou, T. Nishizeki, Algorithms for finding distance-edge-colorings of graphs, J. Discrete Algorithms 5 (2) (2007) 304–322.
[6] I. Kanj, A. Wiese, F. Zhang, Computing the k-hop neighborhoods locally, Available at: http://www.cdm.depaul.edu/SoC/research/Documents/TechnicalReports/2008/TR08007.pdf.
[7] M. Kodialam, T. Nandagopal, Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels, IEEE/ACM Trans. Netw. 13 (4) (2005) 868–880.
[8] N. Linial, Locality in distributed graph algorithms, SIAM J. Comput. 21 (1) (1992) 193–201.
[9] M. Mahdian, On the computational complexity of strong edge coloring, Discrete Appl. Math. 118 (3) (2002) 239–248.
[10] J. Misra, D. Gries, A constructive proof of vizings theorem, Inform. Process. Lett. 41 (3) (1992) 131–133.
[11] S. Ramanathan, A unified framework and algorithm for channel assignment in wireless networks, Wirel. Netw. 5 (2) (1999) 81–94.
[12] S. Ramanathan, E. Lloyd, Scheduling algorithms for multihop radio networks, IEEE/ACM Trans. Netw. 1 (2) (1993) 166–177.
[13] V. Vizing, On the estimate of the chromatic class of p-graphs, Diskret. Analiz 3 (1964) 25–30.