Location Oblivious Distributed Unit Disk Graph Coloring

Mathieu Couture Michel Barbeau Prosenjit Bose
Paz Carmi Evangelos Kranakis

School of Computer Science
Carleton University
January 25 2006

Abstract

We present the first location oblivious distributed unit disk graph coloring algorithm having a provable performance ratio of three (i.e. the number of colors used by the algorithm is at most three times the chromatic number of the graph). This is an improvement over the standard sequential coloring algorithm since we present a new lower bound of 10/3 for the worst-case performance ratio of the sequential coloring algorithm. The previous greatest lower bound on the performance ratio of the sequential coloring algorithm was 5/2. Using simulation, we also compare our algorithm with other existing unit disk graph coloring algorithms.

1 Introduction

A unit disk graph is a graph that admits a representation where nodes are points in the plane and edges join two points whose distance is at most one unit. In wireless ad hoc networks, communicating nodes are sometimes assumed to have the same communication range. For this reason, unit disk graphs are used to model wireless ad hoc networks. Breu and Kirkpatrick [1] showed that determining if an abstract graph is a unit disk graph is an NP-hard problem, which implies that finding a unit disk graph representation is

also NP-hard. This difficulty has led to the development of two varieties of algorithms on unit disk graphs depending on how the graphs are represented. If the unit disk graph representation is given (i.e. vertices are points in the plane and edges join pairs of points whose distance is at most one unit) then this situation is referred to as *location-aware* since each node is aware of its geometric location. On the other hand, if one is simply given an abstract graph (i.e. that a valid representation exists), then this situation is referred to as *location oblivious*. Location oblivious algorithms are desirable because they can be implemented without the use of a GPS (Global Positioning System).

A *coloring* of a graph G is a function c mapping vertices of G to a set of colors (which can be thought of as a set of integers) such that adjacent vertices are assigned different colors. The graph coloring problem is to find a coloring which uses the minimum number of colors. The minimum number of colors needed to color a graph G is called its *chromatic number* and is denoted by $\chi(G)$. It has been pointed out by Hale [5] that the problem of assigning different frequencies to nodes which are within communication range from each other can be formalized as a graph coloring problem. Algorithms using a small number of colors are desirable because they allow the use of fewer frequencies. However, the graph coloring problem is NP-complete [6], even for unit disk graphs [4].

The *performance ratio* of a coloring algorithm is defined as the ratio of the number of colors it uses over the chromatic number of the input graph. Approximation algorithms have been proposed to address the unit disk graph coloring problem (see Erlebach and Fiala [3] for a survey), but there exists no coloring algorithm that is

1. distributed,
2. location oblivious, and
3. has a performance ratio of three.

In this paper, we introduce the first distributed unit disk graph coloring algorithm that has all these three properties.

A standard approach used in the context of coloring graphs is the *sequential coloring algorithm*. The sequential coloring algorithm is the algorithm that colors the nodes of a graph in an arbitrary order, assigning to each node the lowest color that has not been assigned to one of its neighbors. In the

literature, the greatest lower bound on the worst-case performance ratio of the sequential coloring algorithm over unit disk graphs is 5/2, by Caragiannis *et al.* [2]. Therefore, it was unclear whether one slightly more complex algorithm with a performance ratio of three is better than the trivial sequential algorithm. In this paper, we show that algorithms having a performance ratio of three outperform the sequential coloring algorithm in the worst-case by providing an example where the performance ratio of the sequential coloring algorithm is exactly 10/3.

The rest of this paper is organized as follows: In Section 2, we review related work on coloring unit disk graphs. In Section 3, we give our coloring algorithm. We prove its termination, correctness and performance properties in Section 4. In Section 5, we give new lower bounds on the worst-case performance ratio of sequential coloring of unit disk graphs. In Section 6, using simulation, we compare the average performance ratio of our algorithm with other algorithms. In Section 7, we discuss some optimization techniques we used to speed-up the simulation. We conclude in Section 8.

2 Related Work

A *sequential* coloring algorithm takes a graph as input, computes some ordering on the nodes, and greedily assigns colors to nodes according to that order. Each node is assigned the lowest color that has not been assigned to any of its neighbors. We denote the maximum degree of a graph G by $\Delta(G)$, and the size of the largest clique in G (the *clique number* of G) by $\omega(G)$. Since the number of colors used by a sequential coloring algorithm cannot exceed $\Delta(G)+1$, we have that $\chi(G) \leq \Delta(G)+1$. On the other hand, since no two nodes in a clique can have the same color, we have that $\chi(G) \geq \omega(G)$. For unit disk graphs, Marathe *et al.* [8] pointed out the following relation: $\Delta(G) \leq 6\omega(G) - 6$. This implies that all sequential unit disk graph coloring algorithms have a performance ratio of at most six. In fact, a minor adjustment of that proof shows that the performance ratio is no greater than five [3].

What distinguishes sequential coloring algorithms from each other is the order in which they color the nodes. When an arbitrary order is used, we will simply refer to it as *the* sequential coloring algorithm. For graphs embedded in the plane, the *lexicographic* ordering is the one induced by the (x, y) coordinates of the nodes (nodes with smaller x-coordinate are colored first,

	distributed	location oblivious	worst-case perf. ratio
sequential	yes	yes	5
lexicographic	yes	no	3
smallest-last	no	yes	3

Table 1: Summary of Unit Disk Graph Coloring Algorithms Properties.

with ties broken according to the y-coordinate). For the case of unit disk graphs, Peeters [10] showed that the lexicographical ordering achieves a performance ratio of three. Note that this approach can be easily implemented in a distributed manner provided the nodes are aware of their location.

The *smallest-last* coloring algorithm [9] computes the following ordering over the nodes of a graph G: a node v of minimum degree is colored last (ties are broken arbitrarily). The rest of the ordering is computed recursively on the graph $G \setminus \{v\}$. For an ordering $<$ of the nodes of a graph G, we introduce the following notation:

1. G_v is the subgraph of G induced by every node u of G where $u \leq v$;
2. $deg(u, G_v)$ is the degree of node u in G_v;
3. $span(<)$ is the maximum value of $deg(v, G_v)$ according to the order $<$.

Sequentially coloring the nodes of a graph according to $<$ leads to a coloring using at most $span(<) + 1$ colors. Lexicographical orderings have span no greater than $3\omega(G) - 3$. Matula and Beck [9] showed that the smallest-last ordering has minimum span. As pointed out by Gräf *et al.* [4], this implies that the smallest-last coloring algorithm achieves a performance ratio of at most three over unit disk graphs. However, this algorithm is not distributed. Table 1 summarizes unit disk graph coloring algorithms properties. As one can see, there seems to be a trade-off between being distributed, location oblivious, and having a worst-case performance ratio of three. We show that in fact, no such trade-off exists.

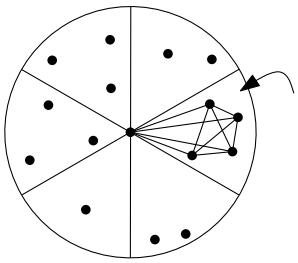

Figure 1: The neighborhood of a node does not contain more than $6\omega(G)-6$ nodes.

3 Location Oblivious Distributed Algorithm

Before giving the details of our algorithm, we first remind the reader why it is the case that for all unit disk graphs G, the following relation holds:

$$\Delta(G) \leq 6\omega(G) - 6.$$

To see this, divide the neighborhood of a node u in six sectors as shown in Figure 1. Since each sector has diameter one, the nodes located within each sector, including u, form a clique. Therefore, u has at most $6\omega(G) - 6$ neighbors. As we mentioned in Section 2, since any coloring must use at least $\omega(G)$ colors, this implies that any sequential coloring algorithm has a performance ratio of at most six over unit disk graphs.

Lexicographic coloring achieves a performance ratio of three because for every node u, no more than $3\omega(G) - 3$ neighbors of u will choose their color before u. The key of our algorithm is to show how to compute an ordering that has this property in a distributed manner when the nodes do not know their position in the plane (i.e. in a location oblivious manner). The main observation is the following: in every unit disk graph G, there is at least one node that has at most $3\omega(G) - 3$ neighbors.

We denote by $\omega(u)$ the size of a largest clique in which node u belongs. If the neighborhood of a node u has size at most $3\omega(u) - 3$, we say that it has the *small neighborhood* property. Lexicographic coloring exploits the fact that the leftmost node has this property. In fact, all nodes on the convex hull of the nodes also have this property. Since the size of a maximum clique in a unit disk graph can be computed in polynomial time, even without the unit disk representation [11], each node can locally determine whether or not

Algorithm 1 RankingPhase(id, G_{id})

Input: id, a node identifier
N, a list containing the identifiers of the neighbors of node id
G_{id}, the subgraph of G induced by N
Output: $ranks$, a table containing the neighbors ranks

```
1: max_clique ← ω(G_id)
2: while {u ∈ N : rank[u] = 0} ≠ ∅ do
3:    if rank[id] = 0 and |{u ∈ N : rank[u] = 0}| ≤ 3 * max_clique − 3
      then
4:       ranks[id] ← max{u ∈ N : ranks[u]} + 1
5:       send RANK(id, ranks[id])
6:    else
7:       receive RANK(u, r)
8:       ranks[u] ← r
9:    end if
10: end while
```

it has the small neighborhood property. Notice that since $\omega(u) \leq \omega(G)$ for every node u, if a node has the small neighborhood property, then it also has at most $3\omega(G) - 3$ neighbors.

The intuition behind our algorithm is the following: in order to reach a performance ratio of three, nodes having the small neighborhood property can pick their colors *after* their neighbors. We then remove all these nodes from the graph, recursively color the remaining subgraph, put the removed nodes back in, and then sequentially color them. Recursion is guaranteed to make progress because there are always nodes having the small neighborhood property. What remains to be shown is how this can be done in a distributed manner.

The distributed algorithm works in two phases. In the first phase, the nodes establish a local order by each selecting a rank. The ranks, together with the identifier, determine the local order in which they will decide their color. The second phase is the actual coloring.

The underlying idea of the ranking algorithm is the following: we want to make sure that for every node u of a unit disk graph G, no more than $3\omega(u) - 3 \leq 3\omega(G) - 3$ nodes pick their color before u. In order to ensure this, each node u collects the connectivity information of its distance one neighborhood and computes $\omega(u)$.

Algorithm 2 ColoringPhase($id, N, ranks$)

Input: id, a node identifier
N, a list containing the identifiers of the neighbors of node id
$ranks$, a table containing the neighbors' ranks
Output: $colors$, a table containing the node's colors (initial values are all 0)

```
1: while {u ∈ N : colors[u] = 0} ≠ ∅ do
2:   if colors[id] = 0 and
       //if the node has not yet chosen its color and,
       //in its neighborhood, it has maximum rank among
       //the nodes which have not yet chosen their color
       {u ∈ N : colors[u] = 0 and ⟨ranks[id], id⟩ < ⟨ranks[u], u⟩} = ∅ then
3:     colors[id] ← min{i > 0 : {u ∈ N : colors[u] = i} = ∅}
4:     send COLOR(id, colors[id])
5:   else
6:     receive COLOR(u, c)
7:     colors[u] ← c
8:   end if
9: end while
```

A node u having a total number of neighbors less than or equal to $3\omega(u)-3$ (i.e. having the small neighborhood property) selects rank one and informs its neighbors of its decision. A node u having more than $3\omega(u)-3$ neighbors must wait. Ranking information from neighbors is recorded in a table. When the number of neighbors of a node u with undetermined rank becomes less than or equal to $3\omega(u)-3$, node u takes a rank that is one more than the maximum rank among its neighbors. Node u then informs its neighbors about its decision. This process continues until all nodes have chosen their rank. Algorithm 1 gives the details of the ranking phase.

When all neighbors have chosen their ranks, a node may start the coloring phase. Note that two neighbors may have chosen the same rank. Locally, nodes then choose their color according to the order induced by the pair $\langle rank, id\rangle$. Nodes with higher rank pick their color first, and ties are broken according to their identifier. Algorithm 2 gives the details of the coloring phase.

4 Theoretical Properties

We now prove the termination and correctness of our algorithm. We also show that it has a performance ratio of at most three.

Proposition 4.1 *After Algorithm 1 terminates, all nodes have selected a rank.*

Proof: Suppose after Algorithm 1 terminates, there is a set of nodes S of G which have not yet chosen their rank. This means that every node $u \in S$ has more than $3\omega(u) - 3$ neighbors which have not yet chosen their rank (i.e. which are in S). In particular, this is true for a node v which is on the convex hull of S. Also, since v is on the convex hull of S, all of its neighbors which are in S are located on a half-plane whose boundary passes through v. Thus, v cannot have more than $3\omega(v) - 3$ neighbors in S, which is a contradiction. Therefore, when no more messages are being sent, all nodes have chosen their ranks. □

Proposition 4.2 *Algorithm 2 produces a valid coloring.*

Proof: First of all, Algorithm 2 terminates. The reason for this is that, among the nodes which have not yet chosen their color, there is always a node which is a global maximum according to the ordered pair $\langle rank, id \rangle$. In particular, this node is a local maximum which will pick its color. Also, no two neighbors can pick their color at the same time. This is because the ordered pair $\langle rank, id \rangle$ induces a total order on the nodes. Therefore, of two neighbor nodes which have not picked their color, at most one of them can satisfy the condition on line 2. Finally, no two neighbors can pick the same color. This is because the second one will only pick a color which is still available (line 2). □

Lemma 4.3 *For a node u of a unit disk graph G, let $h(u)$ denote the number of neighbors of u with higher rank than the rank of u. Then $|h(u)| \leq 3\omega(u) - 3$.*

Proof: In Algorithm 1, a node u will choose its rank only when fewer than $3\omega(u) - 3$ of its neighbors have undetermined rank (line 1). Also, when u chooses its rank, it chooses it such that it is greater than all ranks that have been chosen in its neighborhood. Therefore, only less than $3\omega(u) - 3$ nodes could potentially choose rank greater than the one chosen by u. □

Proposition 4.4 *Using the order computed by Algorithm 1, the color chosen by a node u in Algorithm 2 is less than or equal to $3\omega(u) - 2$.*

Proof: In the neighborhood of u, only nodes with rank greater than u can choose their color before u. By Lemma 4.3, there are no more than $3\omega(u) - 3$ such nodes. Therefore, the color chosen by u is no greater than $3\omega(u) - 2$. □

Theorem 4.5 *Using the order computed by Algorithm 1, the number of colors used by Algorithm 2 to color a unit disk graph G is no greater than three times the optimal. During the execution of these algorithms, each node sends exactly two messages, each one having size $O(\log n)$.*

Proof: By Proposition 4.4, all nodes u are assigned color at most $3\omega(u) - 2 \leq 3\omega(G) - 2$. The performance ratio follows from the fact that at least $\omega(G)$ colors are needed to color G. In Algorithm 1, each node sends exactly one RANK message. In Algorithm 2 each node sends exactly one COLOR message. This is why each node sends exactly two messages. Each message only carries a type, the rank or color of the sender, as well as its identifier. The number of different ranks and colors is bounded by the number of nodes. Therefore, the size of a message is bounded by the size of the greatest identifier, which is $O(\log n)$. □

5 Lower Bounds

We now give new lower bounds on the worst-case performance ratio of the sequential coloring algorithm for unit disk graphs. The currently greatest lower bound is 5/2, given by Caragiannis *et al.* [2]. To prove a lower bound of b, we have to show that there exists a unit disk graph G for which there exists an ordering $<$ of the nodes such that the number of colors used by the sequential coloring algorithm is at least $b \cdot \chi(G)$. The construction of such a unit disk graph proceeds as follows: first, decide what the chromatic number of the graph will be. Then, at least one node must pick color $b \cdot \chi(G)$. In order to ensure this, it must have at least $b \cdot \chi(G) - 1$ neighbors, picking all colors ranging from 1 to $b \cdot \chi(G) - 1$. The construction of the graph then continues recursively in order to force these nodes to pick these colors. To force the sequential algorithm to use many colors, one needs to construct a graph with vertices of high degree. The difficulty lies in increasing the degree of vertices without increasing the chromatic number of the graph.

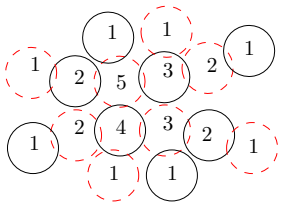

Figure 2: Lower Bound of 5/2.

Figure 2 reproduces the lower bound of 5/2 of Caragiannis *et al.* [2]. The chromatic number of that graph is two (dashed and solid circles form a bipartition) but there exists an ordering of the nodes such that the sequential coloring algorithm uses five colors.

We now prove a lower bound of three. A case where a performance ratio of three can be reached is shown in Figure 3. This graph is bipartite (no two dashed nodes touch each other, and no two solid nodes touch each other). Hence, it can be colored with two colors. However, it is also possible to order the nodes in such a way that the sequential coloring algorithm uses six colors. In order to do this, first take all disks with label 1 in an arbitrary order. Since they form an independent set, the sequential coloring algorithm will assign color 1 to all of them. Then, take all disks with label 2. Since they form an independent set and each of them touches a disk with label 1, the sequential coloring algorithm will assign them color 2. Repeat this procedure until the only disk with label 6 has been colored with color 6. Using this ordering, the sequential coloring algorithm will then use six colors whereas only two colors are necessary, which means that in that case, it achieves a performance ratio of three. Table 4 (page 19) gives the exact positions of the points generating the unit disk graph of Figure 3. Column *seq* gives the colors assigned by the sequential coloring algorithm, while column *opt* gives an optimal coloring of the graph. Thus, since the graph shown in Figure 3 is triangle-free, we have proved the following:

Proposition 5.1 *For triangle-free unit disk graphs, the worst-case performance ratio of the sequential coloring algorithm is at least three.*

The reason why it is interesting to restrict the preceding proposition to triangle-free unit disk graphs is that the bound is tight for that class of graphs, as shown below.

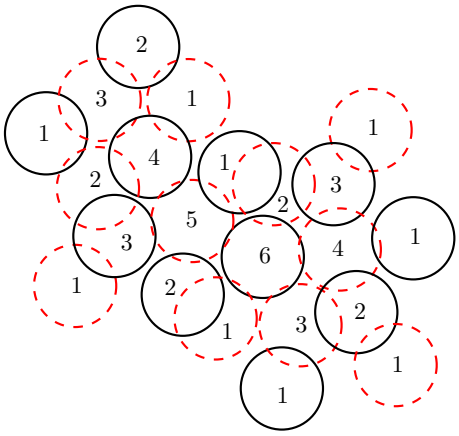

Figure 3: Lower Bound of 3.

Proposition 5.2 *For triangle-free unit disk graphs, the worst-case performance ratio of the sequential coloring is at most three.*

Proof: Suppose that there exists a node u of a unit disk graph G such that the color attributed to u by the sequential coloring algorithm is 7. This means that u has degree at least 6. Since no node of a unit disk graph can have more than five independent neighbors [8], at least two of these six neighbors, say v and w, are neighbors of each other. Therefore, v, w and u form a triangle, which means that G is not triangle-free. □

For graphs that are not necessarily triangle-free, we show a worst-case lower bound of 10/3. The construction is depicted in Figure 4. As one can see, this graph can be colored using only three colors (solid, dashed and dotted nodes form a 3-partition of the graph). However, there exists an ordering of the nodes such that sequentially coloring the graph in that order uses ten colors, leading to a performance ratio of 10/3. In order to force the sequential coloring algorithm to use ten colors, the solid bold node which has degree nine is colored last. Its nine neighbors are forced to take all colors ranging from one to nine thereby forcing color ten on the solid bold node. The coloring of its neighbors is forced in a similar fashion. Exact location of the points are given in Table 5 (page 20), as well as the three-partition of that graph. Nodes are listed in the order that the sequential coloring algorithm needs to use in order to use ten colors. The existence of this graph allows us to conclude the following:

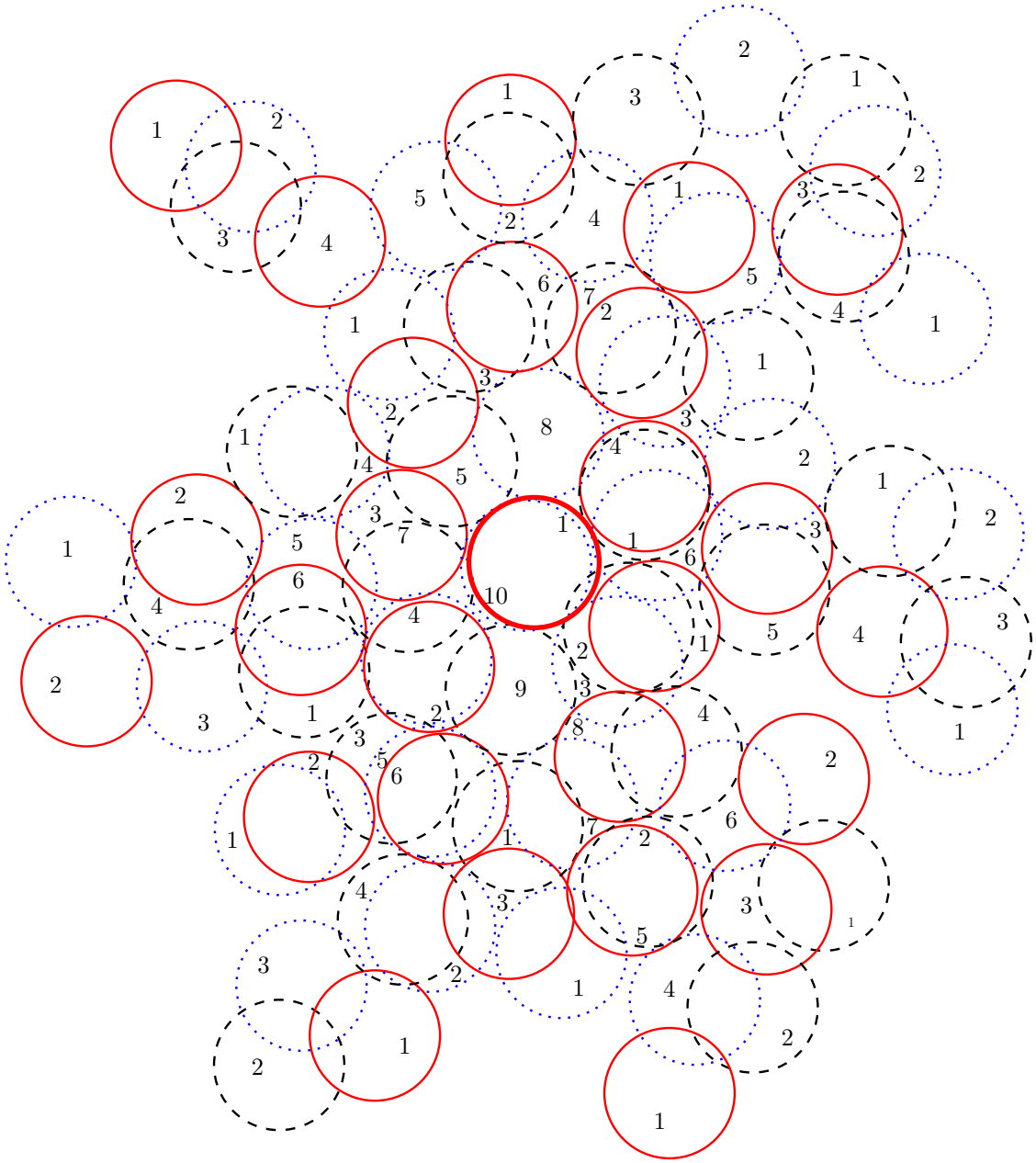

Figure 4: Lower Bound of 10/3.

Proposition 5.3 *For unit disk graphs, the worst-case performance ratio of the sequential coloring algorithm is at least 10/3.*

The importance of this last result is that we now have a confirmation that there exist cases where it is worth making the effort of computing an ordering which is guaranteed to achieve a performance ratio of three. However, as we see from our simulations, when nodes are randomly and uniformly placed in a unit square, all strategies are equally good on average.

6 Simulation Results

In the preceding section, we saw that it is fairly complicated to build an example where the sequential coloring algorithm achieves a performance ratio worse than three. Here, using simulation, we compare the coloring algorithm introduced in this paper with other existing coloring algorithms. Using a fixed radius of 0.05, we first randomly generated 400 unit disk graphs of 200 nodes each. Nodes have been placed on a unit square and their x and y-coordinates have been chosen following a uniform distribution. We also generated unit disk graphs having up to 2000 nodes, by incrementally adding 100 nodes to each of the 400 unit disk graphs.

We then colored each of these unit disk graphs using five different coloring algorithms. Using the heuristic described in the next section, we also computed a lower bound on the size of the maximum clique for each of these unit disk graphs. In order to optimize the running time of the simulation, the same heuristic has also been used to simulate the three-cliques-last coloring algorithm.

The five coloring algorithms we have used are the following: sequential (nodes are colored in the order induced by their identifier), three-cliques-last (the algorithm introduced in this paper), lexicographic (nodes are colored from left to right), smallest-last (nodes of small degree are colored last) and largest-first (nodes of large degree are colored first).

The difference between smallest-last and largest-first is the following: in smallest-last, a node u with minimum degree in a graph G is colored last, and the order in which the other nodes are colored is computed recursively on the graph $G \setminus \{u\}$. In largest-first, a node u with maximum degree is colored first, and the order in which the other nodes are colored is computed recursively on the same graph. Although the largest-first ordering is easier to compute in a distributed manner, it is not known whether it provides a

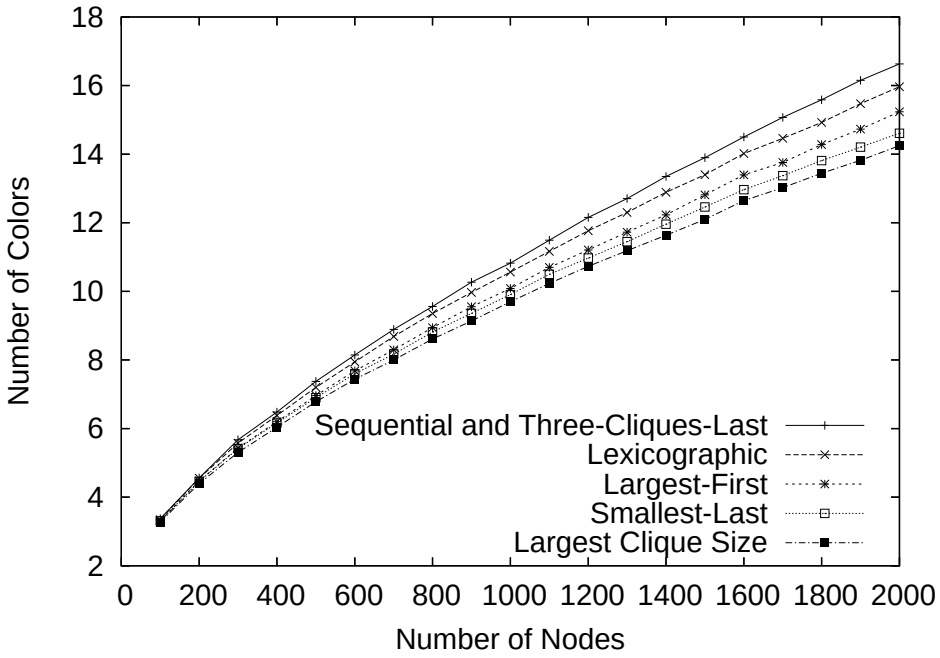

Figure 5: Simulation Results.

better upper bound than five on the performance ratio on unit disk graphs. Smallest-last ordering, on the other hand, is non-trivial to compute in a distributed manner but is known to provide an upper bound of three on the performance ratio when coloring unit disk graphs.

Figure 5 shows the simulation results we obtained. It displays the average number of colors used by each algorithm as a function of the number of nodes in the graph. It also plots the average estimated value of the size of a maximum clique. As explained in the next section, this estimated value is a lower bound on the actual size of a maximum clique. Therefore, it is also a lower bound on the chromatic number. Table 3 (page 19) gives the exact values of our simulation results. On all these values, the maximum standard deviation was 1.2. Therefore, the 95% confidence interval is at most $\pm 2\frac{1.2}{\sqrt{400}} = 0.12$.

The first observation that can be made by looking at the simulation results is that the algorithm we proposed in this paper (three-cliques-last) provides almost no significant improvement over sequential coloring. In fact, the difference between values obtained for the two algorithms is less than the width of the 95% confidence interval. However, this does not really mean

that our algorithm performs badly. What it really means is that sequential coloring performs better than expected. Looking at Table 3, we can see that the ratio of the average number of colors used over maximum clique size is always below 1.17. This means that the sequential coloring algorithm performs quite well although the only known upper bound on the performance ratio is five.

Also, it is not surprising to see that the algorithm which performed the best is the smallest-last coloring. As discussed in Section 2, smallest-last ordering attains minimum span. Since the span of an ordering provides an upper bound on the number of colors that will be used, smallest-last coloring can be expected to provide good results.

What is really interesting is to see is that largest-first coloring provided better results than both three-cliques-last and lexicographic. There is no known proof that largest-first has a performance ratio better than five, and still it performs better than algorithms which have an upper bound of three on the performance ratio. Since largest-first is distributed, location oblivious and simpler to implement than three-cliques-last, looking at the simulation results allows us to conclude that it is preferable to use largest-first even though there is no proof that it performs better.

7 Simulation Optimization

Since computing the maximum clique in the neighborhood of a node can be quite time consuming for simulation purposes, we used some heuristics to compute a lower bound on the size of a largest clique. The main idea of our heuristic is the following: the size of the largest clique is the maximum number of nodes contained in a subset of the plane whose diameter is at most one. Since the geometric shape maximizing an area of fixed diameter is the circle, it is reasonable to hope that the maximum number of nodes contained in a disk of radius one is a good approximation of the size of a maximum clique. Since it is sufficient to look at disks having two nodes on their boundaries, there are only $2\binom{n}{2}$ such disks to look at. Since counting the number of nodes in such a disk can be done in linear time, the maximum number of points contained in a disk of radius one can be computed in time $O(n^3)$.

For a node u, let $C(u)$ be the maximum number of nodes contained in a disk of radius one which also contains u, $\omega(u)$ be the size of a maximum

clique containing u, and $N(u)$ be the set containing u and its neighbors. The heuristic we used is the following: if $|N(u)| \leq 3C(u) - 3$, then use $C(u)$ as an estimate for $\omega(u)$. Otherwise, compute the exact value of $\omega(u)$. Using this estimate instead of computing the exact value of $\omega(u)$ does not affect the simulation results. If a node u is such that $|N(u)| \leq 3C(u) - 3$, then it is also the case that $|N(u)| \leq 3\omega(u) - 3$ and therefore it will be assigned rank one in Algorithm 1 anyway.

nodes	%	nodes	%	nodes	%	nodes	%
100	0.9999	600	0.9977	1100	0.9968	1600	0.9957
200	0.9996	700	0.9976	1200	0.9966	1700	0.9953
300	0.9991	800	0.9975	1300	0.9964	1800	0.9950
400	0.9987	900	0.9971	1400	0.9962	1900	0.9946
500	0.9982	1000	0.9971	1500	0.9960	2000	0.9943

Table 2: Percentages of nodes u such that $|N(u)| \leq 3C(u) - 3$.

Table 2 shows the proportion of nodes u which were such that $|N(u)| \leq 3C(u) - 3$. On all these values, the maximum standard deviation was 0.0020. Therefore, the 95% confidence interval is at most $\pm 2\frac{0.0020}{\sqrt{400}} = 0.0002$. The first observation to be made is that the heuristic allowed us to accelerate the simulation in more than 99% of the cases. This means that the heuristic was worth using it. The second observation to be made is that the percentages diminish as the graph becomes denser. This makes sense, because the area of a disk of diameter one is only 1/4 the area of the unit disk around a node.

The most important observation to be made is that all nodes such that $|N(u)| \leq 3C(u) - 3$ are assigned rank one in Algorithm 1. Therefore, Table 2 also gives a lower bound on the proportion of nodes which are assigned rank one. Since this proportion is always higher than 99%, the order used by Algorithm 2 in the second phase is almost the same as the one used by the sequential algorithm, and this gives an intuition of why the simulation results are so similar for these two algorithms.

8 Conclusion

We presented the first distributed location oblivious coloring algorithm which achieves a performance ratio of three on unit disk graphs. However, simula-

tion results showed that this algorithm does not provide a significant improvement over the algorithm which sequentially colors the nodes in an arbitrary order. Simulation results also showed that, in the average case, largest-first (which is also distributed and location oblivious) performs better than the algorithm we proposed. It also performs better than lexicographic coloring, which also has a worst-case performance ratio of at most three. However, no one has shown whether largest-first has a better worst-case performance ratio than five. In fact, it is also an open question whether coloring the nodes of a unit disk graph in an arbitrary order can, on the worst case, use less than five or more than 10/3 times the minimum number of colors that are necessary.

Acknowledgment

The authors graciously acknowledge the support received from the following organizations: Natural Sciences and Engineering Research Council of Canada (NSERC), Mathematics of Information Technology and Complex Systems (MITACS) and High Performance Computing Virtual Laboratory (HPCVL).

References

[1] H. Breu and D. G. Kirkpatrick, Unit disk graph recognition is np-hard. *Comput. Geom. Theory Appl.*, **9(1-2)**:3–24, 1998.

[2] I. Caragiannis, A. V. Fishkin, C. Kaklamanis, and E. Papaioannou, A tight bound for online coloring of disk graphs. In A. Pelc and M. Raynal, eds., *SIROCCO*, vol. 3499 of *Lecture Notes in Computer Science*, pp. 78–88, Springer, 2005.

[3] T. Erlebach and J. Fiala, Independence and coloring problems on intersection graphs of disks. In *Approximation Algorithms in Combinatorial Optimization*, no. 3484 in Lecture Notes in Computer Science, pp. 135–155, Springer Verlag, 2006.

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

[5] W. K. Hale, Frequency assignment: theory and applications. In *Proceedings of the IEEE*, vol. 68, pp. 1497–1514, 1980.

[6] R. M. Karp, Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, eds., *Complexity of Computer Computations*, pp. 85–103, Plenum Press, 1972.

[7] M. Kubale and L. Kuszner, A better practical algorithm for distributed graph coloring. In *PARELEC*, pp. 72–75, IEEE Computer Society, 2002.

[8] M. Marathe, H. Breu, S. Ravi, and D. Rosenkrantz, Simple heuristics for unit disk graphs. *Networks*, **25**:59–68, 1995.

[9] D. W. Matula and L. L. Beck, Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, **30(3)**:417–427, 1983.

[10] R. Peeters, On coloring j-unit sphere graphs. Tech. Rep. FEW 512, Department of Economics, Tilburg University, Tilburg, The Netherlands, 1991.

[11] V. Raghavan and J. Spinrad, Robust algorithms for restricted domains. *J. Algorithms*, **48(1)**:160–172, 2003.

Appendix 1: Tables

nodes	lgst-cl	sml-last	lgst-first	lex	seq	3-cl-last
100	3.275	3.31	3.31	3.345	3.3775	3.3775
200	4.4	4.46	4.4725	4.5525	4.56	4.56
300	5.305	5.4175	5.4375	5.575	5.675	5.675
400	6.04	6.1625	6.2025	6.3825	6.4825	6.4825
500	6.7825	6.9	6.9625	7.2	7.375	7.375
600	7.44	7.5975	7.6825	7.9425	8.145	8.145
700	8.0025	8.1725	8.2975	8.6875	8.8925	8.8925
800	8.6125	8.815	8.9425	9.355	9.5575	9.56
900	9.1375	9.3575	9.5525	9.97	10.27	10.27
1000	9.695	9.9075	10.085	10.5625	10.825	10.8225
1100	10.2325	10.4975	10.7	11.1675	11.4875	11.4925
1200	10.7325	10.97	11.21	11.7675	12.1575	12.1575
1300	11.1875	11.45	11.7275	12.3025	12.7075	12.705
1400	11.6375	11.96	12.23	12.8875	13.35	13.3525
1500	12.1	12.46	12.815	13.4	13.9	13.8975
1600	12.6475	12.965	13.3975	14.0175	14.4925	14.5025
1700	13.0225	13.3725	13.76	14.4625	15.0675	15.0725
1800	13.445	13.815	14.28	14.925	15.595	15.5875
1900	13.8225	14.205	14.73	15.4725	16.1575	16.155
2000	14.25	14.61	15.235	15.9675	16.6225	16.6325

Table 3: Simulation Results.

seq	opt	x	y	seq	opt	x	y	seq	opt	x	y
1	1	12910	3765	1	2	9460	4875	3	1	5775	2970
1	1	13570	9955	2	1	10360	5190	3	2	11935	5190
1	1	5130	7870	2	1	5730	5295	3	2	6165	6555
1	1	8110	2977	2	2	12535	8560	4	1	12100	6910
1	1	8830	8725	2	2	6790	1575	4	2	7105	4470
1	2	10575	10570	2	2	7965	8100	5	1	8215	6160
1	2	14035	6615	3	1	11065	8905	6	2	10072	7104
1	2	4365	3850								

Table 4: Lower bound of 3 (radius = 2168).

seq	opt	x	y	seq	opt	x	y	seq	opt	x	y
1	1	10154	14461	2	2	6773	8419	5	2	6825	10271
1	1	10428	2425	2	2	6825	12147	5	2	6905	2140
1	1	3293	1290	2	3	11310	13269	5	3	11474	7463
1	1	6057	13660	2	3	4726	14069	5	3	7130	5679
1	1	7942	1209	2	3	7915	1738	5	3	9849	11523
1	1	9814	6022	2	3	9584	7993	6	1	5026	8021
1	1	9943	7968	3	1	11500	11905	6	1	7004	10370
1	2	13717	3696	3	1	11511	6890	6	1	7964	3530
1	2	14087	9128	3	1	12490	2455	6	2	10927	10465
1	2	1831	7075	3	1	6429	6703	6	2	9981	6703
1	2	4737	10801	3	1	7920	11967	7	2	8797	10439
1	2	6257	3913	3	2	10091	4570	7	3	6515	7422
1	2	8160	7126	3	2	3651	8807	7	3	9337	3827
1	2	8652	12511	3	2	5037	12966	8	1	9463	9783
1	3	11249	4475	3	2	9430	8462	8	2	8330	5298
1	3	12298	11575	3	3	14277	8193	9	3	7946	8853
1	3	12601	936	3	3	4124	2140	10	1	8270	7085
1	3	13222	6370	3	3	6289	10084				
1	3	4904	5544	3	3	7367	3809				
1	3	5075	8610	3	3	9718	931				
1	3	8044	10751	4	1	13113	8044				
2	1	12023	10094	4	1	5295	2621				
2	1	2048	8734	4	1	6811	8534				
2	1	3577	6765	4	2	10507	13161				
2	1	5142	10620	4	2	5351	5546				
2	1	6586	4865	4	2	9011	2275				
2	1	9636	11642	4	3	10253	9712				
2	1	9767	4171	4	3	12577	2834				
2	2	11127	250	4	3	3471	7387				
2	2	11550	5714	4	3	6447	12048				
2	2	13015	1641	4	3	9804	6144				
2	2	14171	6687	5	2	10792	2852				
2	2	4330	1582	5	2	5177	7384				

Table 5: Lower bound of 3.3 (radius = 1812).