APPROXIMATION ALGORITHMS FOR GEOMETRIC NETWORKS

by
Mathieu Couture

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
May, 2008

Your file Votre référence
ISBN: 978-0-494-40515-4
Our file Notre référence
ISBN: 978-0-494-40515-4

À Catherine et Maxime:

mes amours, mes deux plus grandes richesses.

Table of Contents

Abstract

We study various combinatorial and geometric problems that have applications in ad hoc wireless networks. From a combinatorial point of view, we are particularly interested in dominating set and coloring problems. We study the impact of solving these problems in a distributed manner on unit disk graphs. We also discuss the importance for the nodes in the network to know their location in the plane. More specifically, we provide algorithms that take advantage of the geometric properties of the model without knowing the actual geometry of a given instance.

From a geometric point of view, we are interested in computing graphs that approximate shortest paths in geometric networks. Such a graph is called a *spanner*. The geometric networks under consideration include unit disk graphs, complete k-partite graphs and complete graphs on additively weighted point sets. For each of these three types of graphs, we provide algorithms that compute spanners that have a linear number of edges and a constant spanning ratio. The spanner we propose for unit disk graphs has bounded out-degree and, when applied to complete graphs, admits a local routing strategy. The spanner we propose for complete k-partite graphs has a spanning ratio of at most $5+\epsilon$, while $3-\epsilon$ is a lower bound for any solution to that problem. For additively weighted point sets, we study two spanners. One has a spanning ratio of $(1+\epsilon)$ and the other has constant spanning ratio and admits a plane embedding.

Finally, we are interested in the problem of computing geometric spanners under the combinatorial constraint that the output graph must have bounded chromatic number k. When $k = 2, 3, 4$, we provide algorithms that are optimal in the sense that the upper bound on the spanning ratio of the output graph is also a lower bound for any given algorithm computing a spanner with the given chromatic numbers. We also give upper and lower bounds for greater values of k. Additionally, we consider an on-line variant of the problem where the points are given one after another, and the color of a point must be assigned at the moment when the point is given; thus, later on, the color of a point cannot be changed. This makes the problem more difficult. Consequently, the bounds are higher, but still tight for $k = 2, 3, 4$.

Acknowledgements

Je désire tout d'abord remercier mon épouse, Catherine, qui m'a supporté et encouragé tout au long de la réalisation de ce projet. Merci à Maxime pour toute la joie qu'il m'a apporté. Merci aussi à ma mère, Nicole, qui m'a donné le goût des études et à mon père, Renaud, qui m'a inculqué l'importance d'être authentique, intègre, et d'avoir confiance en ses idées.

Je remercie mon directeur de recherches, Prosenjit Bose, pour toutes les discussions que nous avons eues. Je lui suis particulièrement reconnaissant pour le souci constant qu'il a eu de voir à mon bien-être. I am also thankful to Paz Carmi, with whom I spent countless hours discussing and debating about my research. Thanks also to the numerous professors of the School of Computer Science who played an important role in the achievement of this research: Michel Barbeau, Jean-Pierre Corriveau, Evangelos Kranakis, Anil Maheshwari, Pat Morin and Michiel Smid. Merci aussi à mes collègues: Paul Boone, Christine Laurendeau, Michel Paquette et Stefanie Wuhrer pour toutes les discussions que nous avons eues et les cafés que nous avons bus. Merci à Lucie, Frédéric, Alexandre, Catherine, François et Jean-François pour votre appui, vos encouragements, et toutes les soirées passées en votre compagnie.

Finalement, je tiens à remercier les organismes suivants pour leur support financier: le CRSNG, MITACS, l'Université Carleton, le Ministère de la recherche et de l'innovation de l'Ontario, et le HPCVL.

Chapter 1

Introduction

An important difference between wired and wireless networks is that in the latter, the physical location of the communicating entities have a direct influence on the possibility to establish a direct link one with another. When direct communication is possible, the distance between the communicating entities also has an influence on the quality and the cost of a communication. Over the years, these observations led researchers to study geometric properties of wireless networks. When communicating nodes are simply described as points in the plane, then the term *geometric network* or *geometric graph* is used in the literature [65]. Unless otherwise specified, in this thesis, a geometric graph is a graph whose vertices are points in the plane and whose edges are straight line segments weighted by their length. Geometric graphs need not be complete graphs. An important case of geometric graph that is not complete is the *unit disk graph.*

A unit disk graph is a (generally unweighted) 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 analysis purposes, the range can be normalized to one unit. For this reason, unit disk graphs have been used since the early 80's [42] to model wireless ad hoc networks. For example, the minimum coloring problem has been used to address frequency assignment [42] and the minimum connected dominating set problem has been used to address routing [6, 27, 87].

In 1990, Clark *et al.* [21] showed that several NP-complete problems, including minimum coloring and minimum (connected) dominating set, remain NP-complete in the special case of unit disk graphs. An exception to that is the problem of finding the size of a largest clique, which can be solved in polynomial time. Five years later, Marathe *et al.* [61] gave heuristics to find constant approximations to

several combinatorial problems on unit disk graphs, including minimum coloring and minimum (connected) dominating set.

In 1998, Breu and Kirkpatrick [16] showed that determining if an abstract (unweighted) graph is a unit disk graph is a 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. nodes 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 given an abstract unit disk graph (i.e. a valid representation exists but is not given), 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). Also in 1998, Hunt *et al.* [44] provided location-aware polynomial time approximation schemes to solve the maximum independent set problem, the minimum vertex cover problem, and the minimum dominating set problem. No polynomial time approximation schemes have been proposed to address these problems in a location-oblivious manner. On the other hand, in 2003, Raghavan and Spinrad [72] showed that the location-aware algorithm proposed by Clark *et al.* [21] to solve the maximum clique problem can be adapted to the location-oblivious setup. Therefore, although it is clearly useful to know the location of the nodes, there is no clear result about how necessary it is.

Approximation algorithms are often used to address NP-hard problems, for which there exists no known polynomial time algorithm. However, this fact does not tell the whole story about approximation in computer science. For example, in the field of computational geometry, geometric spanners are approximating structures that have been widely studied (see Narasimhan and Smid [65] for a survey). A geometric spanner is a subgraph of a geometric graph that approximates shortest paths. Geometric spanners are typically used to reduce the structural complexity of the original graph. Therefore, it is also desirable to construct geometric spanners that have significantly fewer edges than the original graph.

Geometric spanners also have applications in wireless ad hoc networks [35]. They

are used to answer the following question: how many links can a node drop without introducing too much routing overhead? Most of the time, geometric spanners are defined as subgraphs of the complete Euclidean graph of a set of points in the plane. In wireless ad hoc networks, nodes are often considered as having a uniform limited communication range. Therefore, in order to apply theoretical results on geometric spanners to wireless ad hoc networks, it is more realistic to consider geometric spanners that are subgraphs of the unit disk graph.

The problem of choosing a subset of the available physical links to perform routing is only one of many topological problems that are encountered when designing wireless ad hoc networks. Other important problems are assigning roles to nodes in the network and assigning communication channels to nodes. As we discussed, the latter problem can be formalized in terms of graph coloring [42]. When Frequency Division Multiple Access (FDMA) is used, the term *channel* designates a frequency. When Time Division Multiple Access (TDMA) is used, the term *channel* designates a set of time slots during which a node is allowed to transmit. Since it is physically impossible for a node to transmit and receive at the same time [74], assigning the same time slots to adjacent nodes forces them to use an intermediate node to communicate with each other. Therefore, when there is a limited number of channels, the channel assignment problem involves the dropping of certain links and should be done carefully with that fact in mind.

In this thesis, we address the problem of approximating geometric networks, both from a combinatorial and a geometric point of view. The application background we keep in mind is the one of wireless networks.

1.1 Overview of Results

The rest of this thesis is structured as follows: in Chapter 2, we address a generalized version of the dominating set problem: the k-dominating set problem. We provide a location-oblivious distributed k-dominating set algorithm whose output size is guaranteed to be at most six times the size of a minimum one when the input graph is a unit disk graph. Our algorithm works in an incremental manner, i.e. it constructs a monotone family of dominating sets $D_1 \subseteq D_2 \ldots \subseteq D_i \ldots \subseteq D_k$ such that each

D_i is an i-dominating set. We also show that there exist cases where our algorithm can construct a k-dominating set whose size is five times the size of a minimum one. These results were presented at OPODIS 2006 [23] and an extended version has been accepted to the International Journal of Ad Hoc & Sensor Wireless Networks [24].

In Chapter 3, we provide a unit disk graph coloring algorithm that is 1) distributed 2) location-oblivious and 3) has a provable worst-case performance ratio of 3 (i.e. the number of colors used by the algorithm is at most 3 times the chromatic number of the graph). Prior to our work, no algorithm existed that had all these three properties simultaneously, leaving the door open to an apparent tradeoff. Our algorithm allows to conclude that no such tradeoff is necessary. These results were presented at SIROCCO 2007 [22].

In Chapter 4, we introduce a new spanner of the unit disk graph. It is a directed graph where each node is guaranteed to have at most six outgoing edges. The current best upper bound on the out-degree of the nodes of a spanner of the unit disk graph is 9. The spanner we introduce also has a nice property when comes time to use it in an ad hoc network: it allows local routing when applied to a complete graph. This means that once the graph is constructed, it is sufficient to know the coordinates of the destination node in order to deliver a message. Moreover, the path that will be used to forward the message is guaranteed to be at most a constant times longer than the Euclidean distance between the originator of the message and its destination. These results were presented at WADS 2007 [11].

In Chapter 5, given an integer $k \geq 2$, we consider the problem of computing the smallest real number $t(k)$ such that for each set P of points in the plane, there exists a $t(k)$-spanner for P that has chromatic number at most k. We prove that $t(2) = 3$, $t(3) = 2$, $t(4) = \sqrt{2}$, and give upper and lower bounds on $t(k)$ for $k > 4$. We also show that for any $\epsilon > 0$, there exists a $(1 + \epsilon)t(k)$-spanner for P that has $O(|P|)$ edges and chromatic number at most k. Finally, we consider an on-line variant of the problem where the points of P are given one after another, and the color of a point must be assigned at the moment the point is given. In this setting, we prove that $t(2) = 3$, $t(3) = 1 + \sqrt{3}$, $t(4) = 1 + \sqrt{2}$, and give upper and lower bounds on $t(k)$ for $k > 4$. These results were presented at WAOA 2007 [10].

In Chapter 6, we consider a variant of the problem studied in Chapter 5 where the k-partition is given and we want to compute a spanner of the complete k-partite graph K induced by that partition. We present two algorithms for this problem. The first algorithm computes a $(5+\epsilon)$-spanner of K with $O(n)$ edges in $O(n \log n)$ time. The second algorithm computes a $(3+\epsilon)$-spanner of K with $O(n \log n)$ edges in $O(n \log n)$ time. The latter result is optimal: We show that for any $2 \leq k \leq n - \Theta(\sqrt{n \log n})$, spanners with $O(n \log n)$ edges and spanning ratio less than 3 do not exist for all complete k-partite geometric graphs. These results were presented at LATIN 2008 [9].

In Chapter 7, we study the problem of computing geometric spanners for (additively) weighted point sets. A weighted point set is a set of pairs (p, r) where p is a point in the plane and r is a real number. The distance between two points (p_i, r_i) and (p_j, r_j) is defined as $|p_i p_j| - r_i - r_j$. We show that in the case where all r_i are positive numbers and $|p_i p_j| \geq r_i + r_j$ for all i, j (in which case the points can be seen as non-intersecting disks in the plane), a variant of the Yao graph is a $(1+\epsilon)$-spanner that has a linear number of edges. We also show that the Additively Weighted Delaunay graph (the face-dual of the Additively Weighted Voronoi diagram) has constant spanning ratio. The straight line embedding of the Additively Weighted Delaunay graph may not be a plane graph. We show how to compute a plane embedding that also has a constant spanning ratio. These results will be presented at SWAT 2008 [8].

Chapter 2

Incremental Construction of k-Dominating Sets

2.1 Introduction

Constructing and maintaining a structure that allows nodes to communicate with each other is one of the main challenges of ad hoc and sensor networks (see [2] for a survey). A *dominating set* of a graph is a subset of its nodes where each node of the graph is either in the dominating set or has at least one neighbor in the dominating set. Nodes in a dominating set can be used in ad hoc and sensor networks to perform routing by acting as gateways [6, 27, 87]. In sensor networks, dominating sets also help the sensing task itself. Since nodes located close to each other sense similar values, only a dominating set of the nodes is needed to monitor an area. Moscibroda and Wattenhofer [64] showed how this helps prolonging the network's lifetime by turning off the nodes that are not in the dominating set, thereby extending the battery life of these nodes.

Sensor networks typically contain more nodes and each node has less memory than in general ad hoc networks [1]. Therefore, it is important to design algorithms with low memory complexity. One way to achieve this goal is to design distributed algorithms whose memory complexity is only a function of the size of a node's neighborhood as opposed to the total number of nodes.

Sensor nodes are also more error-prone than nodes in general ad hoc networks [1]. They have limited energy resources and need to be periodically redeployed by adding new nodes to the network. The fact that they are error-prone calls for *fault-tolerance* in the design of algorithms for such networks. In algorithms using dominating sets, several authors [26, 55, 90] proposed to address fault-tolerance by using k-dominating sets. In the literature, there are several different definitions of a k-dominating set. Each different definition presents its own computational challenges. To avoid any misunderstanding, we define what we mean by a k-dominating set in this thesis.

Definition 2.1 *A* k-dominating set *of a graph is a subset* S *of its nodes where each node of the graph is either in* S *or is adjacent to at least* k *nodes in* S.

We address the problem of distributively constructing k-dominating sets of unit disk graphs. Unit disk graphs are structures that are used to model wireless networks [42]. Kuhn *et al.* [55] introduced the idea of exploiting clique properties to construct a k-dominating set from a 1-dominating set. The *performance ratio* of a k-dominating set algorithm is defined as the ratio of the size of the k-dominating set it produces over the size of an optimal (minimum) k-dominating set. Kuhn *et al.* [55] claim that their algorithm has an expected performance ratio of $O(1)$ for unit disk graphs.

We generalize dominating set algorithms based on the idea of maximal independent sets [6, 61, 84] to obtain k-dominating sets. A subset of the nodes of a graph is said to be *independent* if it does not contain two adjacent nodes. It is *maximal* if it does not have a proper independent superset. A maximal independent set is also a dominating set. Our algorithm is distributed and, on unit-disk graphs, has a deterministic performance ratio of six. It is also *location-oblivious.* This means that nodes do not need to know their coordinate in the plane. In addition, it constructs the k-dominating set incrementally. More specifically, it constructs a monotone family of dominating sets $D_1 \subseteq D_2 \ldots \subseteq D_i \ldots \subseteq D_k$ such that each D_i is an i-dominating set. Incremental construction of k-dominating sets is helpful when redeploying sensor networks. When sensor nodes in the k-dominating set run out of batteries or experience failure for diverse reasons, the k-dominating set has to be reconstructed. With an incremental algorithm, reconstruction of a k-dominating set can be done by keeping the current dominators.

The problem of computing a k-dominating set of a unit disk graph in a distributed manner has been previously addressed by Kuhn *et al.* [55] and Dai and Wu [26]. We present a comparison of these algorithms with our proposed algorithm where we highlight the strengths and weaknesses of each approach. Our comparison is restricted solely to the above mentioned problem as we note that the algorithms proposed by Dai and Wu [26] address other variants (such as the case where the k-dominating set needs to be k-connected) that are not discussed in our comparison.

The algorithm presented in [55] needs to compute a 1-dominating set that plays a crucial role. Should nodes in that 1-dominating set fail (i.e. run out of energy during the execution of the algorithm), the whole algorithm could fail. Such a failure is likely to happen when reconstructing the k-dominating set after network redeployment. In such a situation, it may happen that nodes from the original 1-dominating set have drained their battery. Our algorithm does not require any node to play such a central role. The algorithm presented in [26] elects all nodes at the same time. It does not allow the incremental augmentation of a j-dominating set to a k-dominating set with $k \geq j$. Our algorithm has this capability (i.e. it provides an explicit incremental construction). The algorithm presented in [55] has an expected performance ratio, and the one in [26] offers probabilistic guarantees both on the performance ratio and the correctness of the output. Our algorithm is guaranteed to produce a k-dominating set with a deterministic performance ratio of at most six. Finally, the algorithm presented in [55] requires nodes to either know their geographic position or be able to modify their communication range as the algorithm runs. Our algorithm is location-oblivious and does not have any requirement on the ability to dynamically modify the communication range.

The rest of this chapter is organized as follows: in Section 2.2, we review related work on the dominating set problem in the context of unit disk graphs. In Section 2.3, we present our algorithm. In Section 2.4, we analyze the performance ratio of our algorithm. In Section 2.5, we compare our algorithm with the one proposed by Kuhn *et al.* [55]. In Section 2.6, we present some simulation results. We draw conclusions in Section 2.7.

2.2 Related Work

2.2.1 Dominating Set

A subset S of the nodes of a graph G is *dominating* G if every node of G is either in S or has at least one neighbor in S. When S is dominating G, we also say that S is a *dominating set* of G. The dominating set problem consists of finding a dominating set of minimum size. It is a special case of the set-cover problem [47].

In the set-cover problem, the input is a set of sets $\mathcal{S}$, and the output is a smallest subset $\mathcal{S}'$ of $\mathcal{S}$ such that the union of the sets in $\mathcal{S}'$ is the same as the union of the sets in $\mathcal{S}$. Set-cover is one of the first problems that have been shown to be NP-complete [47]. The dominating set problem itself has been shown to be NP-complete by a reduction from the vertex-cover problem. The *performance ratio* of a dominating set (or a set-cover) approximation algorithm is defined as the ratio of the size of the dominating set it produces over the size of a minimum one. Johnson [45] proposed a greedy approximation algorithm for the set-cover problem. Its performance ratio is $H(\max\{|S| : S \in \mathcal{S}\})$, where H is the harmonic function[1]. When applied to an instance of the dominating set problem, its performance ratio is then $H(\Delta+1)$, where Δ is the maximum degree in the graph.

For unit disk graphs, the dominating set problem is also NP-complete [21, 62]. For this problem, Marathe *et al.* [61] gave a simple approximation algorithm that achieves a performance ratio of five. It is based on the concept of maximal independent set. A subset S of the nodes of a graph G is said to be *independent* if it does not contain two adjacent nodes. It is *maximal* if it does not have a proper independent superset. A maximal independent set is also a dominating set. Since a node in a unit disk graph has at most five neighbors that are independent of each other, a simple amortized analysis allows to conclude that in a unit disk graph, a maximal independent has at most five times the size of a minimum dominating set. The main advantage of that approximation algorithm is that it is easily implementable in a distributed manner [6].

2.2.2 k-Dominating Set

Several authors [26, 55, 90] proposed to address fault-tolerance in clustering algorithms by ensuring that every node that is not an aggregation point is within range of at least k aggregation points. In other words, they proposed to use a k-dominating set of the network. The k-dominating set problem [26, 55, 64] is a variant of the dominating set problem. A subset S of the nodes of a graph G is k-dominating if every node of G is either in S or has at least k neighbors in S. A node is said to be k-dominated by S if it is either in S or has at least k neighbors in S. The k-dominating

[1] $H(n) := \sum_{i=1}^{n} 1/i$

set problem, as addressed in this thesis, is formally defined as follows:

Problem 2.2 *Given a graph G, find a k-dominating set of G whose size is minimum.*

Other variants of this problem are the k-tuple dominating set problem [50], distance-k dominating set problem [56], max k-cover problem [31] and connected dominating set problem [41]. In the k-tuple dominating set problem, the goal is to find a subset S of the nodes of a graph G such that every node of G has at least k neighbors in S. In contrast to the k-dominating set problem, the constraint of being k-dominated is on all nodes of G as opposed to only nodes in $G \setminus S$. While the k-dominating set problem always has a solution, the k-tuple dominating set problem has a solution if and only if every node has degree at least $k-1$. In the distance-k dominating set problem, the goal is to find a subset S of the nodes of a graph G such that every node of G is either in S or is at at most k hops away from a node in S. In the max k-cover problem, the goal is to find a subset of size k of S that covers the maximum number of nodes. In the connected dominating set problem, one needs to find a dominating set that is also connected.

Note that there is often confusion between the distance-k dominating set problem and the k-dominating set problem. In the literature, both problems have been referred to as *the* k-dominating set problem [56, 64]. To add to the confusion, a k-dominating set has also been defined as a dominating set of size k [3]. The k-dominating set problem discussed in this chapter has also been referred to as the *k-fold* dominating set problem [55].

A p-claw free graph is a graph that does not have a $K_{1,p}$ (the complete bipartite graph having one node on one side and p nodes on the other side) as an induced subgraph. For example, unit disk graphs are 6-claw free graphs. For p-claw free graphs, Klasing and Laforest [50] proposed an approximation algorithm for the k-tuple dominating set problem. Their approximation algorithm works in two phases. In the first phase, they construct k disjoint maximal independent sets. This gives a k-dominating set. In the second phase, they select additional dominators in order to ensure that nodes in the k-dominating set are also k-dominated. They showed that the approximation ratio of their algorithm is at most $\frac{(p-1)}{2}(k-1+\frac{2}{k})$.

Work on the k-dominating set problem includes results from Dai and Wu [26], Moscibroda and Wattenhofer [64] and Kuhn *et al.* [55]. Dai and Wu [26] proposed three different algorithms. The first one is randomized and gives a k-connected[2] k-dominating set with high probability. The second algorithm is another generalization of Wu *et al.* [89]. It is deterministic and is proven to always give a k-connected k-dominating set should one exist. In both cases, the decision is based on local or k-local information, which means that a node needs to be aware of the nodes that are within k hops. The third algorithm proceeds as follows: first, each node randomly and independently picks a color from 1 to k. This partitions the graph into k subgraphs. Second, an existing connected dominating set algorithm is applied on each color.

Moscibroda and Wattenhofer [64] addressed the problem of finding k *disjoint* dominating sets (the union of which makes a k-dominating set). Their algorithms are based on coloring schemes. Note that the problem they addressed is slightly different since the parameter k is not given as input. Instead, they find the largest k such that it is possible to find k disjoint dominating sets.

Kuhn *et al.* [55] proposed two algorithms. The first one is for general graphs and uses a distributed rounding scheme. The second one is for unit disk graphs and works in two phases. The first phase reuses an algorithm by Gao *et al.* [34] to construct a 1-dominating set. In the second phase, the 1-dominators select the other dominators. The authors give a proof that the expected performance ratio of their algorithm is $O(1)$.

2.2.3 Connected Dominating Set

To compute a minimum connected dominating set is NP-complete [47], even for unit disk graphs [21]. Guha and Khuller [41] gave the first approximation algorithms for the connected dominating set problem on general graphs. They proposed three greedy algorithms. Two algorithms are based on growing a tree. The third algorithm consists of growing a forest. The algorithms have a performance ratio of $O(\log n)$, where n is the number of nodes in the graph. Das and Bharghavan [27] implemented these global algorithms in a distributed framework.

[2]A graph is *k-connected* if it remains connected after the removal of any $k-1$ nodes.

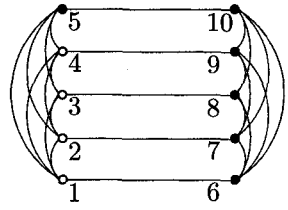


Figure 2.1: Performance ratio of $\frac{n}{4}$ for the algorithm presented in [88].

Wu and Li [87] introduced a local algorithm based on the notion of a replacement path: *If all of your neighbors form a clique, then do not elect yourself as a dominator. Otherwise, elect yourself as a dominator.* Wan *et al.* [84] showed that the worst case performance ratio of that algorithm is at least $\frac{n}{2}$. Figure 2.1 depicts the worst case. Nodes 1 to 5 and 6 to 10 form two cliques, and nodes with the same y-coordinate are connected. In this graph, there is a connected dominating set of size two (nodes 1 and 6), but the execution of the algorithm of Wu and Li would elect every node as a dominator. On the other hand, by their construction, two nodes are guaranteed to be connected by a path of dominators with a minimum number of hops [87]. Wu and Li [88] further improved their algorithm by observing that if a node u has two connected neighbors v and w such that the neighborhood of u is included in the union of the neighborhoods of v and w, and if the identifier of u is smaller than the identifiers of v and w, then u should not be elected as a dominator. The proof in [84] can be adjusted to show that the worst case performance ratio in that case is at least $n/4$. Again in Figure 2.1, all the black nodes get elected while only two nodes suffice. Instead of using only two connected neighbors, Dai and Wu [25] then proposed the use a replacement path of length at most k. They called this improved version of the algorithm *Rule k*. Hansen and Schmutz [43] analyzed the expected size of the connected dominating set produced by Rule k. The *Rule k* algorithm has been further generalized by Wu and Dai [86] in the form of a generic coverage condition. The generic coverage condition is the following: *If for any two of your neighbors v and w, there exists a replacement path between v and w such that all nodes on that path have identifiers higher than yours, then do not elect yourself as a dominator. Otherwise, elect yourself as a dominator.* It is more general than *Rule k* because there is no bound on the size of the replacement path and nodes on the replacement

path need not be neighbors of node u. Finally, Stojmenovic *et al.* [79] proposed to improve the algorithm presented in [87] by giving some priority to nodes having higher degree.

Another stream of research in connected dominating sets uses the notion of independent set. Marathe *et al.* [61] pointed out that using maximal independent sets in unit disk graphs provides a simple way to approximate a minimum dominating set within a factor of five. Alzoubi *et al.* [6] and Wan *et al.* [84] showed that a similar strategy leads to a performance ratio of 8 in the case of connected dominating sets. Cardei *et al.* [19] obtained similar results. Li *et al.* [59, 60] successively reduced the performance ratio to $5.8 + \ln 4$ and $4.8 + \ln 5$, respectively.

2.3 Incremental Algorithm

Alzoubi *et al.* [6] and Wan *et al.* [84] addressed the construction of a *connected* dominating set. Their algorithm consists of two phases. The first phase constructs a maximal independent set. A maximal independent set is also a dominating set. In this section, we generalize the first phase of the algorithm presented in [6, 84] to obtain a k-dominating set. Our generalization augments a $(k-1)$-dominating set in order to obtain a k-dominating set. More specifically, we construct a monotone k-dominating family.

Definition 2.3 *A* k-dominating family *is a sequence* $D_1, D_2, \ldots, D_k$ *of subsets of nodes of a graph such that for all* $i = 1, 2, \ldots, k$, D_i *is an* i*-dominating set. A* k*-dominating family is* monotone *provided that the sequence of dominating sets is monotonically increasing under inclusion, i.e.* $D_1 \subseteq D_2 \subseteq \cdots \subseteq D_k$.

The key idea of our algorithm is to first produce a 1-dominating set by constructing a maximal independent set. Then, we build a maximal independent set of the nodes that are not 2-dominated. Adding this set to the 1-dominating set gives a 2-dominating set. We repeat the procedure until we have a k-dominating set. The construction of each dominating set is similar to the approach in [6, 84].

We now present our algorithm in detail. We assume that every node has a unique identifier. In an initialization phase, each node sends its identifier to its immediate

Algorithm 2.1: DOMINATING SET(id, N, k)

Input: id, a node identifier
N, a list containing the identifiers of the neighbors of node id
k, the required number of dominators for a non-dominating node
Output: *dominator*, a Boolean indicating whether the node id is a dominator or not
Local Variables: *round*, the current round
candidate, a lookup table indicating whether or not a node n is a candidate to be a dominator in round r (all initial values are **true**)
domcount, a lookup table counting the number of dominating neighbors for each round (all initial values are zero)

```
1  dominator ← false;
2  round ← 1;
3  if id < min(N) then
4  |  dominator ← true;
5  |  send JOIN(id, 1);
6  |  exit;
7  end
8  while round ≤ k do
9  |  receive message;
10 |  if message is GIVE-UP(n, r) then
11 |  |  candidate[n, r] ← false;
12 |  end
13 |  if message is JOIN(n, r) then
14 |  |  for i = r to k do
15 |  |  |  candidate[n, i] ← false;
16 |  |  |  domcount[i] ← domcount[i] + 1;
17 |  |  end
18 |  |  while domcount[round] ≥ round do
19 |  |  |  send GIVE-UP(id, round);
20 |  |  |  round ← round + 1;
21 |  |  end
22 |  end
23 |  if round ≤ k and id < min{n ∈ N | candidate[n, round] = true} then
24 |  |  dominator ← true;
25 |  |  send JOIN(id, round);
26 |  |  round ← k + 1;
27 |  |  exit;
28 |  end
29 end
```

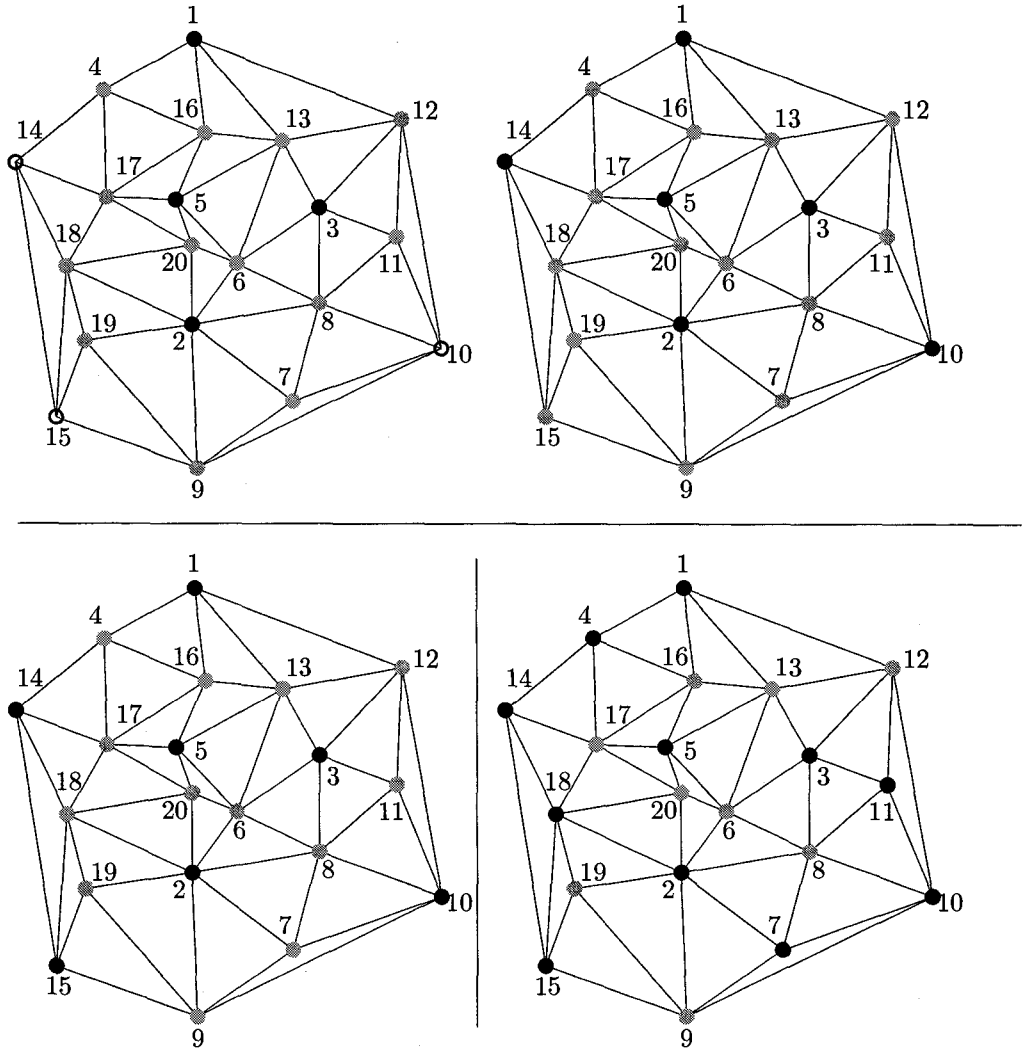


Figure 2.2: Marking Process Example for $k = 1$ (above) and $k = 2$ and 3 (below).

neighbors. After initialization, two types of messages are used: JOIN(id, i) and GIVE-UP(id, i), where id is the identifier of the sending node and $i = 1, \ldots, k$ identifies a round. These messages are only sent to immediate neighbors. The JOIN(id, i) message means that for $j = i, \ldots, k$, the sender joins the j-dominating set. A node sending the JOIN(id, i) message is said to be *marked* in round i. The GIVE-UP(id, i) message means that the sender is excluded of the i-dominating set. After transmitting a JOIN message (lines 5 and 25), the sender remains silent. A node is said to be a *candidate* for round i if it is not part of the $(i-1)$-dominating set and it has never sent the GIVE-UP(id, i) message (line 19). Following the completion of the initialization

phase, every node that has a lower identifier than that of all its immediate neighbors sends the JOIN$(id, 1)$ message (line 5). The rest of the algorithm is message driven. Algorithm 2.1 specifies how each node should behave. Note that different nodes may simultaneously execute in different rounds.

Figure 2.2 illustrates the marking process for $k = 1$, 2 and 3. Nodes in black are dominators. Nodes in grey are k-dominated. Nodes in white are not k-dominated. For $k = 1$, nodes 1, 2, 3 and 5 have the smallest identifier among their 0-dominated neighbors and thus declare themselves dominators. Initially, node 10 can not declare itself a dominator because of nodes 7, 8 and 9. However, after node 2 has declared itself a dominator, nodes 7, 8 and 9 become 1-dominated. Node 10 is then allowed to declare itself a dominator. The same reasoning applies to node 14. The 1-dominating set is then $\{1, 2, 3, 5, 10, 14\}$. For $k = 2$, there is only one new dominator, node 15. For $k = 3$, the new dominators are nodes 4, 7, 11 and 18.

2.4 Theoretical Properties

In this section, we review the theoretical properties of our algorithm. We first show that our algorithm computes a valid k-dominating set and a monotone k-dominating family. Then, we analyze the worst case performance ratio of our algorithm. In the latter part, we follow the general idea of Kuhn *et al.* [55]. More precisely, we first show that no unit disk can contain more than a given number of dominators (i.e. $5k$). Then, we use properties of k-dominating sets to show that this leads to a constant performance ratio.

Proposition 2.4 *For all nodes v, the value of the round variable eventually reaches $k + 1$.*

Proof: Suppose there are nodes for which this is not true, and let v be the node that has minimum identifier among those having minimum value of *round* when no more messages are sent. Let $i \leq k$ be the value of *round* for v at this moment. This means that v has not sent a JOIN(id, i) message. Consequently, it must have a neighbor u with a smaller identifier that is still a candidate for round i (line 23). Since u is still a candidate for round i, the value of its *round* variable is at most i. But this contradicts

the fact that v has minimum identifier among the nodes having minimum value of *round*. Therefore, the value of *round* eventually reaches $k+1$ for all nodes v. □

Proposition 2.5 *Let S_i $(i \leq k)$ be the set of nodes that are marked in rounds $j = 1, \ldots, i$ of Algorithm 2.1. Then S_i is an i-dominating set.*

Proof: Suppose a node v is not i-dominated by S_i after round i. By Proposition 2.4, the value of *round* for v is eventually at least i. The value of *round* for v is updated either at line 20 or line 26. If it is updated in line 26, then $v \in S_i$ which is a contradiction. Therefore, the only place it can be updated is in line 20. However, this implies that $domcount[i]$ has value at least i, which is only true if v has received at least i JOIN(n, r) messages with $r \leq i$. This means there are at least i nodes that are adjacent to v and have been marked in a round with index at most i (i.e. there are at most i nodes in S_i that are adjacent to v), which is a contradiction. □

In order to claim that Algorithm 2.1 produces a monotone k-dominating family, we also have to show the monotonicity property.

Proposition 2.6 *For $i = 1, \ldots, k$, let S_i be defined as above. Then for all $i = 0, \ldots, k-1$, $S_i \subseteq S_{i+1}$.*

Proof: Let $n \in S_i$. It has been marked in a round $j \leq i < i+1$. By construction of S_{i+1}, we have $n \in S_{i+1}$. □

For unit disk graphs, there is an upper bound on the cardinality of the k-dominating set computed by Algorithm 2.1. We first need to show that at each round, the elected nodes form an independent set.

Proposition 2.7 *In any given round, the nodes marked by Algorithm 2.1 form an independent set.*

Proof: Suppose that in a given round, two adjacent nodes v_1 and v_2 declare themselves dominators. Without loss of generality, suppose v_1 has a lower identifier than

v_2. This means that as long as v_1 did not send a give-up message, v_2 cannot elect itself a dominator. But since v_1 never sends such a message (a node cannot both send a give-up message and a join message), v_2 can never declare itself a dominator. This means that two adjacent nodes cannot declare themselves dominators in the same round. □

For $k = 1$, on unit disk graphs, we can use the above property to show that the set of marked nodes is not larger than five times the size of an optimal dominating set [61]. For $k > 1$, it is not the case that the set of elected nodes is independent.

Proposition 2.8 *Let $G = (V, E)$ be a unit disk graph, C be a unit disk and $S \subseteq V$ be the set of nodes marked by Algorithm 2.1. Then $|S \cap C| \leq 5k$.*

Proof: By Proposition 2.7, S is the union of k independent sets. Since a unit disk cannot contain more than 5 independent nodes [61], $S \cap C$ cannot contain more than $5k$ nodes. □

Proposition 2.9 *Let $G = (V, E)$ be a graph, S a subset of V, t an integer and $OPT_k = \{v_1, \ldots, v_{|OPT_k|}\}$ an optimal k-dominating set of G. If $|S| > t|OPT_k|$, then there is at least one node $v \in OPT_k$ such that $|N(v) \cap S| > k(t-1)$, where $N(v)$ is the set formed by v and its neighbors.*

Proof: Let S' be $S \setminus OPT_k$. Since $|S'| \geq |S| - |OPT_k| > t|OPT_k| - |OPT_k|$, we have $|S'| > (t-1)|OPT_k|$. For each $v_i \in OPT_k$, define S_i as $N(v_i) \cap S'$. Since each node in S' is adjacent to at least k nodes in OPT_k, we have that

$$\sum_{i=1}^{opt_k} |S_i| \geq k|S'| > k(t-1)|OPT_k|$$

Therefore, by the pigeonhole principle, one of the S_i contains more than $k(t-1)$ nodes. The result follows from the fact that $S_i \subseteq N(v_i) \cap S$. □

The two last propositions allow to prove the following:

Theorem 2.10 *Let $G = (V, E)$ be a unit disk graph, $S \subseteq V$ the set of nodes marked by Algorithm 2.1 and OPT_k an optimal k-dominating set. Then $|S| \leq 6|OPT_k|$ (i.e. the performance ratio is at most six). During the execution of the algorithm, each node sends at most k messages on each link, each one having size $O(\log k + \log n)$.*

Proof: Suppose $|S| > 6|OPT_k|$. By Proposition 2.9, there is at least one node $v \in V$ such that $|N(v) \cap S| > 5k$. But this contradicts Proposition 2.8, and therefore $|S| \leq 6|OPT_k|$.

In each of the k rounds, a node either remains silent or broadcasts one message (either a JOIN(id, i) or a GIVE-UP(id, i)). Therefore, during the execution of the whole algorithm, it sends at most k messages on each link. Each message only carries a type, a round identifier and a node identifier. Thus, the size of a message is bounded by the sum of the size of the greatest node identifier and the size of the greatest round identifier, which is $O(\log k + \log n)$. □

As stated before, for $k = 1$ it is known that no independent set can be larger than five times the size of an optimal dominating set [61]. We then have a leap from five to six when we generalize the maximal independent set algorithm for dominating sets to k-dominating sets. At first sight, it may be surprising but the leap actually comes directly from the definition of a k-dominating set: the only nodes that need to have at least k neighbors in the dominating set are the ones that are not in the dominating set. Nodes that are in the k-dominating set do not need to have k neighbors in the dominating set.

As we see in the next section, it is that property that motivates our concerns regarding the proof of Kuhn *et al.* [55] of the performance ratio of their algorithm.

We now show that for any k, there exist graphs for which our algorithm has a performance ratio of five. It is an open question whether or not it is possible to bridge the gap between five and six. First, we need the following lemma:

Lemma 2.11 *Let $\triangle ABC$ be an isosceles triangle such that $\angle BAC = \angle ACB = \phi$, p be a point located on the line AB such that A is between p and B, and q be a point located on the line CB such that C is between q and B. Then $|pq| > |AC|$.*

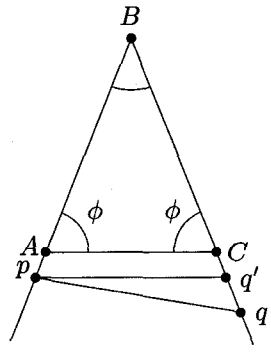


Figure 2.3: Lemma 2.11.

Proof: If $|pB| = |qB|$, then $\triangle pBq$ is similar to $\triangle ABC$, and $|pB| > |AB|$ implies $|pq| > |AC|$. Suppose now that $|pB| < |qB|$, and let q' be the point located on the line CB such that C is between q' and B and $|q'B| = |pB|$. By the first case, $|pq'| > |AC|$. Now, since $\triangle ABC$ is isosceles, $\phi < \frac{\pi}{2}$, and since $\angle pq'q = \pi - \phi$, we have that $\angle pq'q > \frac{\pi}{2}$. Therefore, $\angle pq'q$ is the largest angle of $\triangle pq'q$, meaning that its opposite side, pq, is the largest side. In particular, we have $|pq| > |pq'| > |AC|$. The case where $|pB| > |qB|$ is equivalent. □

Proposition 2.12 *The worst case performance ratio of Algorithm 2.1 is at least five.*

Proof: For $k = 1$ and $n = 6$, place five nodes equally spaced on the boundary of a disk of radius 1, and place a node in the center of that disk. Since the disk has radius 1, the center node shares an edge with all the other nodes. Also, since the distance between every pair of nodes on the circle is at least $2\sin\frac{\pi}{5} > 1$, there is no other edge in the unit disk graph. In the remainder of the proof, this basic structure is referred to as a *star*, the node placed in the center of the disk is referred to as the *center* of the star (pictured as squares) and the five nodes on the boundary of the circle are referred to as the *branches* of the star (pictured as filled circles). The center node forms a dominating set of the star. However, if the center happens to have an identifier higher than the ones of the branches, all branches are marked as dominators, leading to a performance ratio of five.

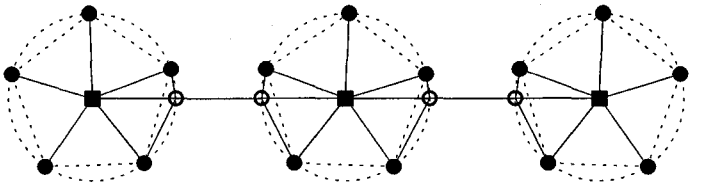


Figure 2.4: Lower bound of five for Algorithm 2.1.

Figure 2.4 depicts the interconnection of several stars. We show how to construct examples of size $14 + 8m$, for any given m (in Figure 2.4, $m = 1$). The construction goes as follows: place $m + 2$ stars along the x-axis such that their center is placed at x-coordinates $0, 3, 6, \ldots, 3(m + 1)$ and no branch lies on the x-axis. Since the circles in which the stars are inscribed are at distance at least 1 from each other, the only edges of the graph are the ones linking the star branches to their respective center. To connect the graph, add nodes on the intersection of the disks with the x-axis. These nodes are referred to as *bridging nodes* (pictured as hollow circles). Since the centers of two consecutive stars are at distance 3 from each other, the two bridging nodes between them are at distance 1 from each other. Therefore, there is an edge between two bridging nodes located between the centers of two consecutive stars, making the whole graph connected. To see how a performance ratio of five can be reached, notice that the star centers form a dominating set of size $m+2$. However, since the set of all branches form an independent set, these nodes may be marked as dominators. This situations leads to a dominating set of size $5(m+2)$, which gives a performance ratio of five.

For $k > 1$, Figure 2.5 shows a generalization of the star structure. The goal is to map each node of the star to a set of k nodes such that:

1. nodes mapped to the center share an edge with every other node and
2. nodes mapped to a branch only share edges with nodes mapped to the center and nodes mapped to the same branch.

In order to achieve this, draw a regular pentagon having side length of 1. Let C be the inscribing circle of that pentagon and $r = \frac{1}{2\sin(\frac{\pi}{5})}$ be the radius of C. Now, let C_1 and C_2 respectively be the circles having the same center as C and radii $r_1 = \frac{1-r}{2}$ and

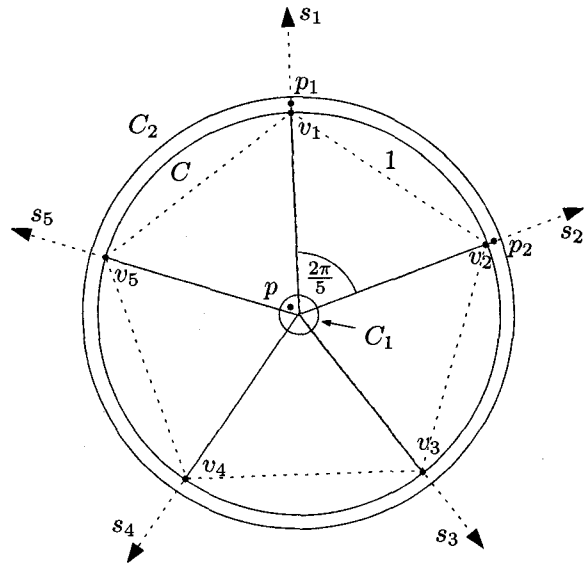


Figure 2.5: Widget for $k > 1$.

$r_2 = r + r_1$. For each node v_i of the pentagon (i from 1 to 5), let s_i be the half-line from the center of C through v_i. Now, let p be a point located inside C_1, and p_1 and p_2 be two points located on s_i and s_j ($i \neq j$), between C_2 and C. Then, Lemma 2.11 tells us that

$$|p_1, p_2| > |v_1, v_2| \geq 1$$

and from the triangle inequality, we have

$$|p, p_1| \leq r_1 + r_2 = 2(\frac{1-r}{2}) + r = 1.$$

Similarly, $|p, p_2| \leq 1$. The construction we need is then the following: place k points inside C_1 and k points on every s_i between C and C_2. We call the result of that construction a *generalized star*. The points located inside C_1 form a k-dominating set, but the algorithm may mark all nodes located on the s_i's. Since there are $5k$ such points, the performance ratio is five in that case. To construct a lower bound example with $k > 1$ for large n, we link the generalized stars in a way similar to the case $k = 1$. □

2.5 Comparison with Kuhn et al.'s algorithm

To the best of our knowledge, only Kuhn *et al.* [55] proposed an algorithm producing a k-dominating set with probability 1. However, their performance ratio is not deterministic. In this section, we compare their algorithm with ours. The most important property in [55] about the proposed algorithm is Lemma 5.6, which states that in a disk of radius $\frac{1}{2}$, their algorithm elects at most $O(k)$ leaders in the expected case. In the deterministic case, our algorithm elects at most k leaders.

Proposition 2.13 *Let $G = (V, E)$ be a unit disk graph, C a disk of radius $\frac{1}{2}$ and $S \subseteq V$ the set of nodes marked by Algorithm 2.1. Then $|S \cap C| \leq k$.*

Proof: The proof goes the same way as for Proposition 2.8. By Proposition 2.7, S is the union of k independent sets. Since the maximum number of independent nodes a unit disk can contain is 1, $S \cap C$ can not contain more than k nodes. □

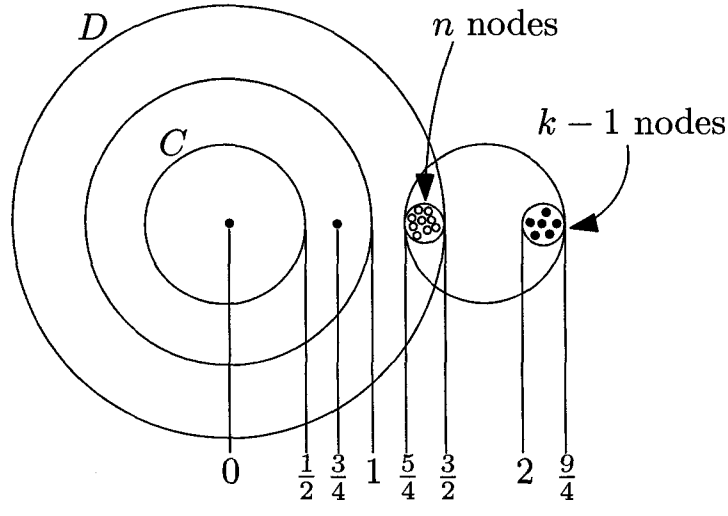


Figure 2.6: Counter-example to the proof of Theorem 5.7 of [55].

Kuhn *et al.* [55] actually showed that the performance ratio of the proposed algorithm is $O(1)$. However, their proof uses the fact that, in order to k-dominate the nodes in a disk C of radius $\frac{1}{2}$, every optimal k-dominating set must elect at least k dominators in a disk of radius $\frac{3}{2}$ having the same center as C. However, this is not

always the case. Figure 2.6 shows how to construct a counter-example for any $k \geq 2$ (on the figure, $k = 7$). Black nodes are the dominators and white nodes are the non-dominators. The node at $\frac{3}{4}$ is only there to make the network connected. The only node in C is 7-dominated but only two of the $n + 2$ nodes in D are selected as dominators. We believe their theorem is still correct. However, we did not investigate how to adjust their proof in order to account for the above situation.

2.6 Simulation Results

We generalized an existing independent set-based algorithm [6, 61, 84] in order to incrementally construct a k-dominating set. We chose to generalize this specific algorithm because it is distributed and has constant performance ratio. By simulation, we compare our algorithm with k-generalized versions of other available algorithms. Stojmenovic *et al.* [79] suggested the following heuristic to improve the independent set algorithm of [6, 61, 84]: instead of ordering the nodes according to their identifier, order them by their degree first and then by their identifier. Higher priority is granted to nodes having higher degree. The performance ratio is still at most five, but the case of Figure 2.4 is avoided. However, it has not been proven that the worst-case performance ratio is strictly smaller than five when using this heuristic. For the k-dominating set problem, it is not desirable to favor higher degree nodes. The reason is that nodes having degree less than k cannot have k dominating neighbors, so they must necessarily be in the k-dominating set.

Selecting nodes of higher degree is the same idea that is behind the greedy set-cover algorithm. The greedy set-cover algorithm first favors nodes that dominate the largest number of nodes not yet dominated. Although this is a global selection criterion, it still has to be examined. At first sight, since it does not have constant performance ratio (recall that it is $H(\Delta + 1)$, where Δ is the maximum degree of a node in the network and H is the harmonic function), one would believe that it would not perform as well as our algorithm. However, it turns out that in order to have $H(\Delta + 1) > 5$, we need Δ to be at least 82, and to reach six, we need Δ to be at least 225. If we assume that it is not likely to have nodes having that many neighbors, this algorithm still deserves attention.

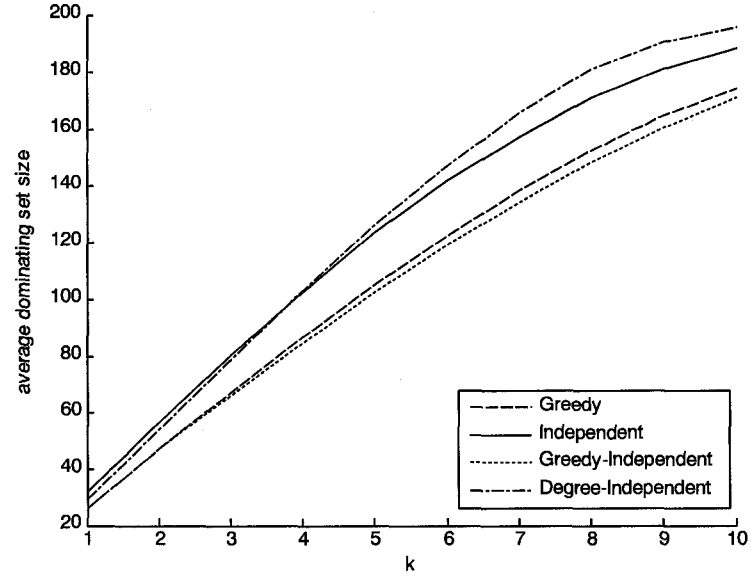
In this section, we discuss simulation results comparing Algorithm 2.1 with k-generalized versions of both the algorithm presented in [79] and the greedy algorithm. We also compare it with the greedy construction of a maximal independent set. The k-generalized versions of these algorithms work the same way we generalized the maximal independent set algorithm: for $k = 1$, we run the standard algorithm on all nodes. For $k \geq 2$, we run the standard algorithm on nodes that are not yet k-dominated. We ran our simulations 200 times for networks of 200 nodes. Nodes have been placed on a unit square and their x and y-coordinates have been chosen following a uniform distribution. We chose a communication range such that with high probability, the network is connected. According to Penrose [70, 71], for any integer $k \geq 0$ and real constant c, if the nodes have identical radius r given by the formula:

$$r = \sqrt{\frac{\ln n + k \ln \ln n + \ln(k!) + c}{n\pi}}$$

then the network is $(k+1)$-connected with probability $e^{-e^{-c}}$ as n goes to infinity. For $n = 200$, choosing $k = 1$ and $c = 5$, we then obtain that for a radius of $r \approx 0.138$, the network is 2-connected with probability 0.99.

Figure 2.7 shows the simulation results we obtained. The 95% confidence interval for these values is ± 0.67 node. The algorithm that performs the best is the one that greedily constructs an independent set (greedy-independent). Not far behind is the greedy algorithm. It is worth noting that even if those algorithms perform slightly better, they are not distributed. This is because the greedy choice of the next node to be marked is based on a global criterion. For the two distributed algorithms, it is interesting to note that the one using the ordered pair degree-id (degree-independent) only performs better for small values of k (5 and less). For $k > 5$, it is the one simply based on identifiers (independent) that performs better.

Unfortunately, Figure 2.7 does not show the optimal solution. Since the dominating set problem is NP-complete, only exponential time algorithms are known to solve the problem exactly. Therefore, only small instances of the problem can be simulated. Figure 2.8 compares the same algorithms with the optimal solution for a network of 35 nodes. We ran over 200 simulation cases. The 95% confidence interval for these values is ± 0.28 node. Figure 2.9 shows the average performance ratio we obtained



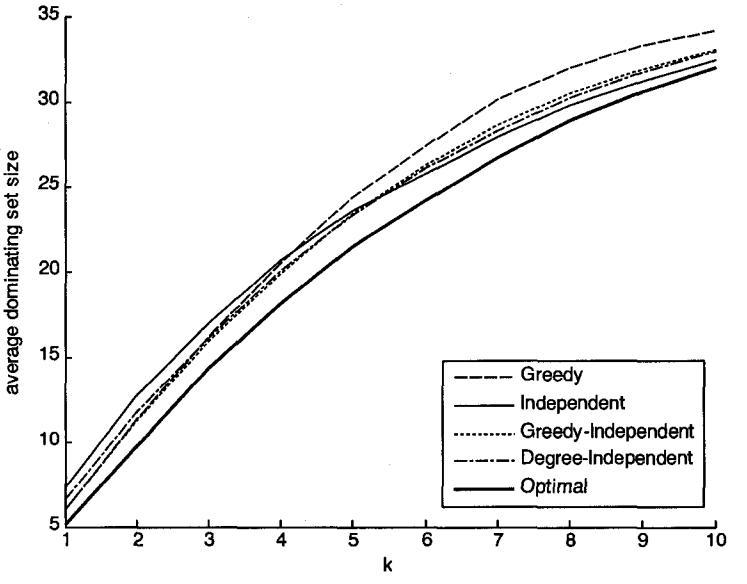
alg / k	1	2	3	4	5	6	7	8	9	10
greedy	26.03	47.17	67.3	86.69	105.01	122.39	138.29	152.32	164.32	174.21
ind	31.72	54.49	73.36	90.1	104.46	117.87	130.27	141.59	151.92	161.33
gr-ind	26.07	47.17	66.5	85.03	101.54	117.09	131.34	143.91	154.76	164.16
deg-ind	29.26	50.82	69.49	86.46	101.76	116.43	129.82	141.73	152.85	162.62

Figure 2.7: Average dominating set size for 200 nodes.

for each algorithm. For small values of k, the two global greedy algorithms are the best, followed by the distributed algorithm granting priority to high degree nodes. The algorithm simply based on identifiers performs the worst. However, as k grows, the results change completely. The algorithm simply based on identifiers becomes the best, and the basic greedy algorithm becomes the worst. The algorithm constructing independent sets by granting priority to high degree nodes performs slightly better than the greedy construction of an independent set.

2.7 Conclusion

In this chapter, we introduced a new algorithm to address the k-dominating set problem. Our algorithm has a deterministic performance ratio of six. The previously best algorithm had an expected performance ratio of $O(k)$ for an unspecified constant [55]. We showed that the size of the k-dominating set our algorithm produces may be five times bigger than the optimal one. However, it is an open issue whether or not the gap



alg / k	1	2	3	4	5	6	7	8	9	10
greedy	5.98	11.35	16.19	20.46	24.3	27.43	30.13	31.99	33.3	34.17
ind	7.35	12.78	16.93	20.67	23.57	25.78	27.95	29.76	31.19	32.45
gr-ind	5.99	11.31	15.95	19.89	23.4	26.25	28.68	30.46	31.87	33.03
deg-ind	6.66	11.82	16.09	20.04	23.26	26.09	28.25	30.26	31.74	32.9
optimal	5.1	9.81	14.31	18.09	21.48	24.18	26.68	28.93	30.61	31.96

Figure 2.8: Average dominating set size for 35 nodes.

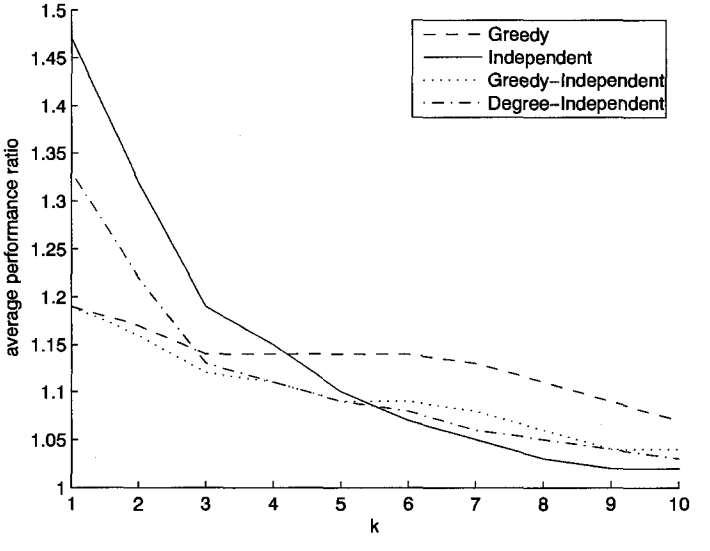


Figure 2.9: Average performance ratio for 35 nodes.

between five and six can be closed. The expected performance ratio is also unknown.

Simulation results showed that in some cases, the k-generalized version of the greedy dominating set algorithm performs better than ours. However, besides their worst-case performance ratio, another important difference between the greedy dominating set algorithm and ours is that the former is global while ours is distributed. As discussed in the introduction, distributed algorithms are often desirable in the context of ad hoc networking.

One potential drawback of our algorithm is that in the worst case, it may need a linear number of communication rounds to terminate. The algorithms proposed by Dai and Wu [26] and Kuhn *et al.* [55] have the advantage of being local, which means that the number of communication rounds is bounded by a constant. However, our worst case behavior is an artifact of location-obliviousness. As pointed out by Urrutia [82], a general framework exists to convert distributed algorithms on unit disk graphs (such as ours) into local algorithms by imposing a hexagonal tiling of the plane and solving the problem independently in each hexagonal tile. Since the number of hops between any pair of nodes within a tile is bounded by a constant, this process leads to a local algorithm that runs in a constant number of rounds in the worst case. Note that the resulting algorithm is location-aware. Following the idea of Urrutia [82], since a hexagonal tiling is three-colorable, the performance ratio of our algorithm within this local framework is at most $3 \times 6 = 18$ in the worst case.

We believe that differences between the performance of global, distributed and local algorithms is an interesting avenue of research. Important work in that field has been done by Kuhn [52] and Kuhn *et al.* [53, 54].

Chapter 3

Location-Oblivious Distributed Coloring

3.1 Introduction

In 1998, Breu and Kirkpatrick [16] 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. nodes 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 given an abstract unit disk graph (i.e. a valid representation exists but is not given), then this situation is referred to as *location-oblivious*. Location-oblivious algorithms are desirable in the wireless setting because they can be implemented without the use of a GPS (Global Positioning System). Also in 1998, Hunt *et al.* [44] provided location-aware polynomial time approximation schemes to solve the maximum independent set problem, the minimum vertex cover problem, and the minimum dominating set problem. In 2003, Raghavan and Spinrad [72] showed that the location-aware algorithm proposed by Clark *et al.* [21] to solve the maximum clique problem can be adapted to the location-oblivious setup. In 2004, Nieberg and Hurink [66] proposed a location-oblivious polynomial time approximation scheme for the dominating set problem on unit disk graphs. In general, although it is clearly useful to know the location of the nodes, there is no clear result about how necessary it is.

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 [42] 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 [47], even for unit disk graphs [36].

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 [30] 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 chapter, we introduce the first 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. Tsai *et al.* [81] gave a constructive proof that for any $k > 2$, there exists a unit disk graph G such that $\chi(G) = k$ and for which the sequential coloring algorithm may use as much as $4k - 3$ colors. This means that for any $\epsilon > 0$, there exists a unit disk graph for which the worst-case performance ratio of the sequential coloring algorithm is at least $4 - \epsilon$. Therefore, algorithms having a performance ratio of three outperform the sequential coloring algorithm in the worst-case.

The rest of this chapter is organized as follows: In Section 3.2, we review related work on coloring unit disk graphs. In Section 3.3, we give our coloring algorithm. We prove its termination, correctness and performance properties in Section 3.4. In Section 3.5, we give new lower bounds on the worst-case performance ratio of sequential coloring of unit disk graphs. In Section 3.6, using simulation, we compare

the average performance ratio of our algorithm with other algorithms. In Section 3.7, we discuss some optimization techniques we used to speed-up the simulation. We conclude in Section 3.8.

3.2 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. The greedy algorithm assigns to each node 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 $\Delta(G)+1$, $\chi(G) \leq \Delta(G)+1$. 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.* [61] 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 [30].

What distinguishes sequential coloring algorithms from each other is the order in which they color the nodes. When an arbitrary order is used, we 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, with ties broken according to the y-coordinate). For the case of unit disk graphs, Peeters [69] 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 [63] 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:

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

Table 3.1: Summary of Unit Disk Graph Coloring Algorithms Properties.

1. G_v is the subgraph of G induced by all nodes 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 [63] showed that the smallest-last ordering has minimum span. As pointed out by Gräf *et al.* [36], 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. For a more extensive survey on unit disk graph coloring, see Erlebach and Fiala [30].

Table 5.1 summarizes properties of unit disk graph coloring algorithms. 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. In this chapter, we show that in fact, no such trade-off exists by providing a location-oblivious distributed coloring algorithm that has a performance ratio of three.

3.3 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 3.1. Since each sector has diameter one, the nodes located within each sector, including

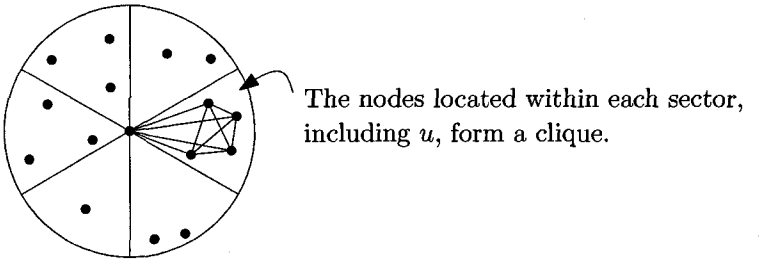


Figure 3.1: The neighborhood of a node does not contain more than $6\omega(G)-6$ nodes.

u, form a clique. Therefore, u has at most $6\omega(G) - 6$ neighbors. As we mentioned in Section 3.2, 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 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 [72], each node can locally determine whether or not 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 (i.e., those with the small neighborhood property) from the graph, recursively color the remaining subgraph, put the removed nodes back in, and then sequentially color them. Recursion is

Algorithm 3.1: 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 (all initial values are zero)

```
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  |  end
7  |  else
8  |  |  receive RANK(u, r);
9  |  |  ranks[u] ← r;
10 |  end
11 end
```

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 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 $\omega(u)$.

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

Algorithm 3.2: 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 nodes colors (initial values are all 0)

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

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 3.1 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 3.2 gives the details of the coloring phase.

3.4 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 3.1 *After Algorithm 3.1 terminates, all nodes have selected a rank.*

Proof: Suppose after Algorithm 3.1 terminates, there is a set of nodes S of G that have not yet chosen their rank. This means that every node $u \in S$ has more than $3\omega(u) - 3$ neighbors that have not yet chosen their rank (i.e. that are in S). In particular, this is true for a node v that is on the convex hull of S. Also, since v is on the convex hull of S, all of its neighbors that 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 3.2 *Algorithm 3.2 produces a valid coloring.*

Proof: First of all, Algorithm 3.2 terminates. The reason for this is that, among the nodes that have not yet chosen their color, there is always a node that is a global maximum according to the ordered pair $\langle rank, id \rangle$. In particular, this node is a local maximum that picks 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 that have not picked their color, at most one of them can satisfy the condition on line 5. Finally, no two neighbors can pick the same color. This is because the second one only picks a color that is still available (line 7). □

Lemma 3.3 *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 3.1, a node u chooses its rank only when fewer than $3\omega(u) - 3$ of its neighbors have undetermined rank (line 3). 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 3.4 *Using the order computed by Algorithm 3.1, the color chosen by a node u in Algorithm 3.2 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 3.3, 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 3.5 *Using the order computed by Algorithm 3.1, the number of colors used by Algorithm 3.2 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 on each link, each one having size $O(\log n)$.*

Proof: By Proposition 3.4, 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 3.1, each node broadcasts exactly one RANK message. In Algorithm 3.2 each node broadcasts exactly one COLOR message. This is why each node sends exactly two messages on each link. 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 $O(\log n)$. □

3.5 Lower Bounds

We now give new lower bounds on the worst-case performance ratio of the sequential coloring algorithm for unit disk graphs. 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 nodes of high degree. The difficulty lies in increasing the degree of nodes without increasing the chromatic number of the graph.

The currently greatest lower bound is $4 - \epsilon$, given by Tsai *et al.* [81]. They gave a constructive proof that for any $k > 2$, there exists a unit disk graph G such that $\chi(G) = k$ and for which the sequential coloring algorithm may use as much as $4k - 3$ colors.

A case where a performance ratio of three can be reached is shown in Figure 3.2. 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 assigns 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 assigns 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 then uses six colors whereas only two colors are necessary, which means that in that case, it achieves a performance ratio of three. Table 3.4 (page 46) gives the exact positions of the points generating the unit disk graph of Figure 3.2. 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 3.2 is triangle-free, we have the following:

Proposition 3.6 *For triangle-free unit disk graphs, the worst-case performance ratio of the sequential coloring algorithm is at least three.*

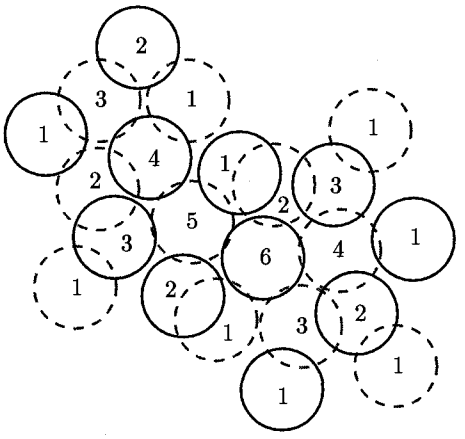


Figure 3.2: Lower Bound of 3.

The above result was also shown by Tsai *et al.* [81]. They had a different construction using 29 disks, whereas the construction presented here uses 22 disks. 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. The following result was also proved by Tsai *et al.* [81]. Here, we present an alternate proof.

Proposition 3.7 *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 [61], at least two of these six neighbors, say v and w, are neighbors of each other. Therefore, v, w and u form a triangle, and G is not triangle-free. □

For graphs that are not necessarily triangle-free, we show a worst-case lower bound of 10/3. For unit disk graphs having chromatic number at most four, this lower bound is greater than the one obtained by Tsai *et al.* [81] since their construction for $\chi(G) = 4$ gives a lower bound of 13/4. Our construction is depicted in Figure 3.3. As one can see, this graph can be colored using only three colors (solid, dashed and dotted

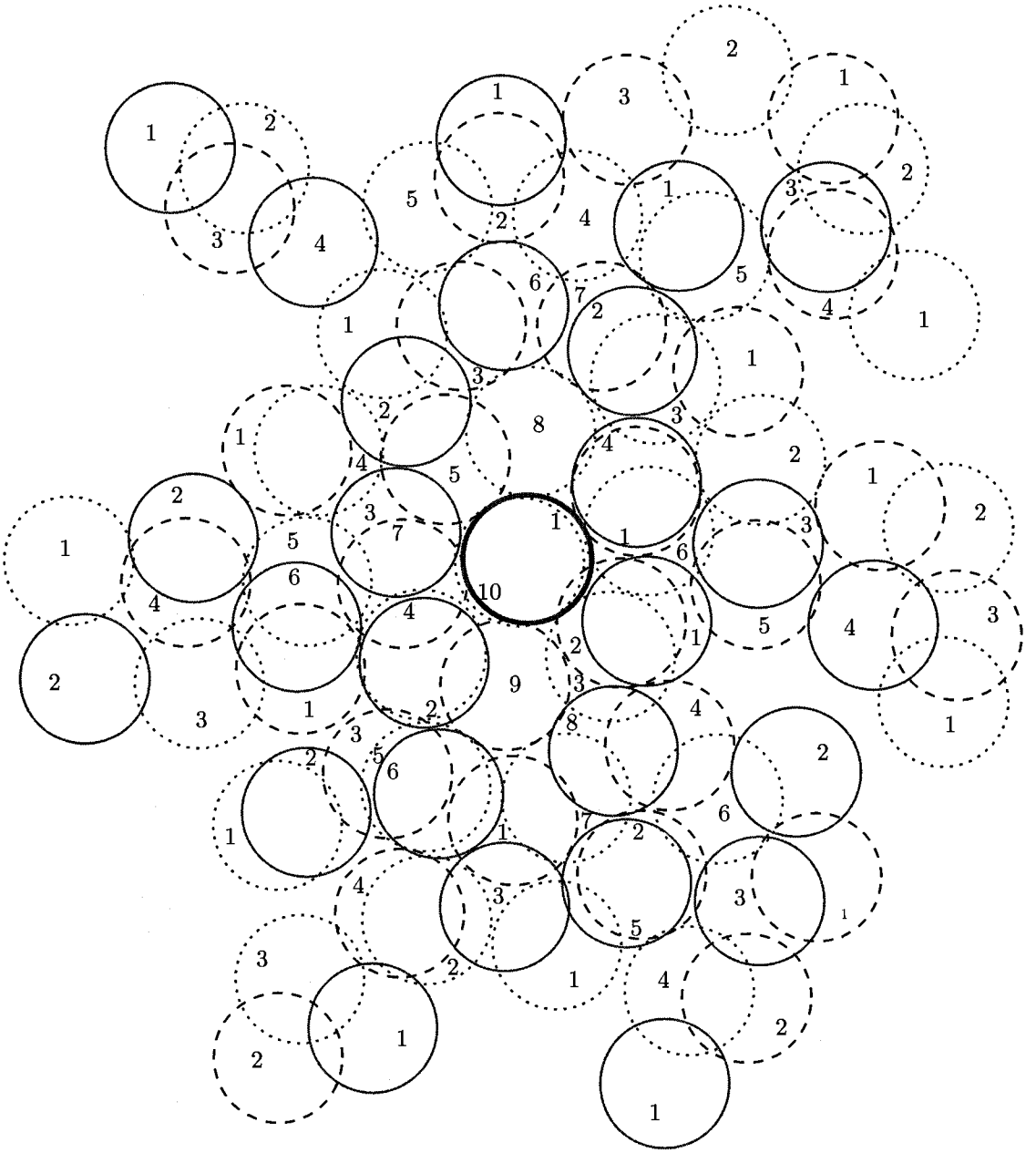


Figure 3.3: Lower Bound of 10/3.

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 that 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 3.5 (page 47), 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:

Proposition 3.8 *For unit disk graphs having chromatic number at most four, the worst-case performance ratio of the sequential coloring algorithm is at least 10/3.*

The existence of lower bounds greater than three confirms that there exist cases where it is worth making the effort to compute an ordering that 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.

3.6 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 chapter 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 algorithm introduced in this chapter).

The five coloring algorithms we used are the following: sequential (nodes are colored in the order induced by their identifier), three-cliques-last, 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 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 3.4 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 3.3 (page 46) gives the exact values of our simulation results. The 95% confidence interval is ± 0.12.

The first observation that can be made by looking at the simulation results is that the algorithm we proposed in this chapter (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 3.3, we can see that the ratio of the average number of colors used over maximum

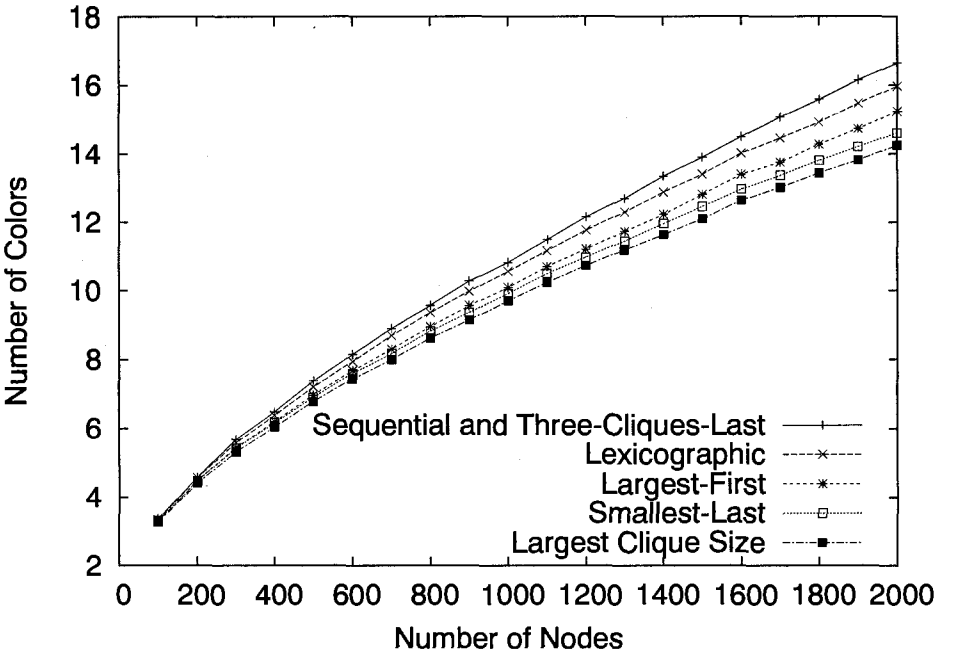


Figure 3.4: Simulation Results.

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 that performed the best is the smallest-last coloring. As discussed in Section 3.2, smallest-last ordering attains minimum span. Since the span of an ordering provides an upper bound on the number of colors that are 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 that 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.

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

3.7 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 that 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 is assigned rank one in Algorithm 3.1 anyway.

Table 3.2 shows the proportion of nodes u that were such that $|N(u)| \leq 3C(u)-3$. The 95% confidence interval is ± 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 it was worth using the heuristic. 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 3.1. Therefore, Table 3.2 also gives a lower bound on the proportion of nodes that are assigned rank one. Since this proportion is always higher than 99%, the order used by Algorithm 3.2 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.

3.8 Conclusion

We presented the first distributed location-oblivious coloring algorithm that achieves a performance ratio of three on unit disk graphs. However, simulation results showed that this algorithm does not provide a significant improvement over the algorithm that 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, in the worst case, use less than five or more than four times the minimum number of colors that are necessary.

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 3.3: Coloring 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 3.4: Sequential Coloring 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 3.5: Sequential Coloring Lower bound of 3.3 (radius = 1812).

Chapter 4

Towards Half-Space Proximal

4.1 Introduction

A graph G whose vertices are points in $\mathbb{R}^d$ and edges are segments weighted by their length is a *geometric graph*. A geometric graph G is a t-spanner (for $t \geq 1$) when the weight of the shortest path in G between any pair of points p, q does not exceed $t \cdot |pq|$ where $|pq|$ is the Euclidean distance between p and q. Any path from p to q in G whose length does not exceed $t \cdot |pq|$ is a t-spanning path. The smallest constant t having this property is the *spanning ratio* of the graph. A t-spanning path from p to q is *strong* if the length of every edge in the path is at most $|pq|$.

Geometric spanners have applications in wireless ad hoc networks [35]. They are used to answer the following question: how many links can a node drop without introducing too much routing overhead? The graph G is a *strong* t-spanner if there is a strong t-spanning path between every pair of vertices. For wireless networks where nodes have a uniform communication range, it is consequently desirable to use strong spanners because they guarantee the existence of paths that are not too long compared to the shortest path in the complete communication graph. When using a spanner that is not strong, it is possible that a spanning path contains an edge that is longer than the edge being approximated. If such an edge is also longer than the node's maximum communication range, then the spanning path cannot be used to route a message.

Another property of geometric spanners that is sometimes desirable is the existence of a local routing strategy. Locally finding a t-spanning path means that all the information a point p who wishes to send a message to a point q has is its own position, the position of its neighbors and the position of q [15].

The spanning properties of various geometric graphs have been studied extensively in the literature (see the book by Narasimhan and Smid [65] for a comprehensive

survey on the topic). We are particularly interested in spanners that are defined by some proximity measure or emptiness criterion (see for example Bose *et al.* [12]). The work presented in this chapter was initiated by Chavez *et al.* [20] who introduced a new geometric graph called Half-Space Proximal (HSP). Given a set P of n points in the plane, the HSP graph is constructed as follows: the $\binom{n}{2}$ edges of the complete graph on P are considered for insertion in increasing order of length. There is an (oriented) edge from p to q if there is no point r such that: (1) $|pr| < |pq|$, (2) (p, r) is an edge of HSP and (3) q is closer to r than to p.

The authors show that this graph has maximum out-degree[1] at most 6. The authors also claim that HSP has a spanning ratio[2] of $2\pi + 1$ and that this bound is tight[3]. Unfortunately, we found statements made in their proofs for the latter two claims to be erroneous or incomplete as we outline in Section 4.4. However, in reviewing their experimental results as well as running some of our own, although their proofs are incomplete, we felt that the claimed results might be correct. Our attempts at finding a correct proof to their claims was the starting point of this work. Although we have been unable to find a correct proof of their claims, we introduce a family of directed geometric graphs that approach HSP asymptotically and possess several other interesting characteristics outlined below.

In this chapter, we define a family of graphs denoted as G_λ^θ graph. This is a family of directed geometric graphs dependent on the parameters λ and θ, where $\frac{1}{2} < \lambda < 1$ and $0 \leq \theta < \frac{\pi}{2}$. We show that this family of graphs has bounded out-degree, and is a strong t-spanner, where both the out-degree and t depend on λ and θ. Furthermore, graphs in this family admit local routing algorithms that find strong t-spanning paths. Finally, we show that all strong t-spanners are also spanners of the unit-disk graph, which are often used to model ad hoc wireless networks.

The remainder of this chapter is organized as follows: in Section 4.2, we review related work on geometric spanners. In Section 4.3, we introduce the G_λ^θ graph and prove its main theoretical properties. In Section 4.4, we compare the G_λ^θ graph to the HSP. In Section 4.5, we show that the G_λ^θ graph is a strong t-spanner and,

[1]Theorem 1 in [20]
[2]Theorem 2 in [20]
[3]Construction in Figure 2 in [20]

consequently, a spanner of the unit disk graph. In Section 4.6, we present simulation results about the G_λ^θ graph. We conclude in Section 4.7.

4.2 Related Work

Well known examples of geometric t-spanners include the Yao graph [91], the θ-graph [75], graphs based on the Well-Separated Pair Decomposition (WSPD) [18], and the Delaunay graph [48]. The WSPD is reviewed in Chapter 6 and the Delaunay graph is reviewed in Chapter 7.

Let θ be an angle such that $2\pi/\theta = k$, where k is an integer. The Yao graph with angle θ is defined as follows. For every point p, partition the plane into k cones $C_{p,1}, \ldots, C_{p,k}$ of angle θ and apex p. Then, there is an oriented edge from p to q if and only if q is the nearest point to p in some cone $C_{p,i}$. The Yao graph is sometimes confused with the θ-graph, although they are different graphs. The first phase of the construction of the θ-graph using k cones with angle θ and apex p is identical to the construction of the θ-graph. This may be the root of the confusion. However, there is an oriented edge from p to q in the θ-graph if and only if q has the shortest projection on the bisector of the cone containing q.

On Yao graphs [91], the spanning ratio is at most $1/(\cos\theta - \sin\theta)$ provided that $\theta < \pi/4$. This means that the maximum out-degree of a node in the Yao graph is at least nine. To find a t-spanning path in the Yao graph, one simply has to choose as a next step the edge that lies in the same cone as the destination. Bose *et al.* [14] showed that the Yao graph is a strong t-spanner.

On θ-graphs, the spanning ratio is at most $1/(1 - 2\sin\frac{\theta}{2})$ [75]. For a fixed angle θ, this is better than for Yao graphs. Also, the bound holds for $\theta < \pi/3$, thus for this range of θ, the maximum out-degree is seven. However, the θ-graph is not a strong t-spanner.

Bose *et al.* [14] showed that the Delaunay graph is also a strong t-spanner, where $t = 2\pi/(3\cos(\pi/6))$. However, the problem of locally finding a good t-spanning path in the Delaunay graph is difficult. Bose and Morin [15] showed how this can be achieved, but there is a price to pay: the proven constant blows up to $9(\frac{1+\sqrt{5}}{2}\pi)+\frac{\pi}{2} \approx 47.32$.

More recently, Chavez *et al.* [20] introduced a new graph called Half-Space Proximal (HSP). Given a set P of n points in the plane, the HSP graph is constructed as follows: the $\binom{n}{2}$ edges of the complete graph on P are considered for insertion in increasing order of length. There is an (oriented) edge from p to q if there is no point r such that:

1. $|pr| < |pq|$,
2. (p, r) is an edge of HSP and
3. q is closer to r than to p.

The HSP graph can also be constructed from a unit disk graph. In that case, only the edges of the unit disk graph are considered for insertion. The HSP graph has maximum out-degree at most six. In [20], the authors present proofs that HSP has spanning ratio at most $2\pi + 1$ and that this bound is tight. In both cases, as we discuss in Section 4.4, there was insufficient detail in the proofs for us to reconstruct the arguments. Also, the authors did not provide a way to locally find a t-spanning path in HSP. In this chapter, we introduce a new family of strong parametric spanners that converge to HSP. These graphs also admit a routing strategy that allows to locally find a t-spanning path. Depending on the parameter values that are chosen, the out-degree of a node can be at most six.

In the HSP, the *construction path* from p to q is defined as follows: if there is an edge (p, q), then the path is the edge (p, q). Else, then let r be the nearest node to p such that there is an edge (p, r) and q is closer to r than to p. In that case, the construction path from p to q is the edge (p, r) followed by the construction path from r to q. In [20], it is unclear whether the construction path is also a spanning path. Because it is a simple routing strategy, it is desirable for construction paths to be also spanning paths. In the next section, we introduce a variation of the HSP having the property that the construction path is also a spanning path.

4.3 The family G_λ^θ of graphs

In this section, we define the G_λ^θ graph and prove that it is a strong t-spanner of bounded out-degree. We first introduce some notation. Let P be a set of points in

the plane, $0 \leq \theta < \frac{\pi}{2}$ and $\frac{1}{2} < \lambda < 1$.

Definition 4.1 *The* θ-cone(p, r) *is the cone of angle* 2θ *with apex* p *and having the line through* p *and* r *as bisector.*

Definition 4.2 *The* λ-half-plane(p, r) *is the half-plane containing* r *and having as boundary the hyperplane perpendicular to* $\overline{pr}$ *and intersecting* $\overline{pr}$ *at distance* $\frac{1}{2\lambda}|pr|$ *from* p.

Definition 4.3 *The* destruction region *of* r *with respect to* p, *denoted* $K(p, r)$, *is the intersection of the* θ*-cone*(p, r) *and the* λ*-half-plane*(p, r) *(see Figure 4.1).*

Definition 4.4 *The directed graph* $G_\lambda^\theta(P)$ *is computed as follows. For every point* $p \in P$, *do the following:*

1. *Let* $N(p)$ *be the set* $P \setminus \{p\}$.
2. *Let* r *be the closest point to* p *in* $N(p)$ *(ties on the distances are broken arbitrarily).*
3. *Add the edge* (p, r) *to* $G_\lambda^\theta(P)$.
4. *For all* q *in* $K(p, r)$, *remove* q *from* $N(p)$
   (i.e., $N(p) \leftarrow N(p) \setminus (K(p, r) \cap P)$*).*
   Note that for each q *deleted in this step,* r *is the* destroyer *of the edge* (p, q).
5. *If* $N(p)$ *is not empty go to 2.*

Remark 4.5 *The directed graph* $G_\lambda^\theta(P)$ *is the graph having* P *as nodes and there is an edge* $(p, q) \in G_\lambda^\theta(P)$ *iff there is no point* $r \in P$, *such that*

1. $|pr| \leq |pq|$,
2. (p, r) *is an edge of* $G_\lambda^\theta(P)$ *and*
3. $q \in K(p, r)$.

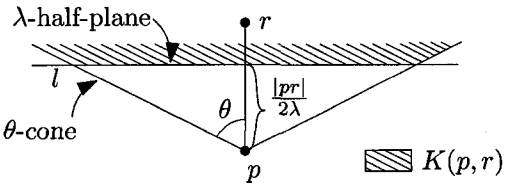


Figure 4.1: The destruction region of r with respect to p.

4.3.1 Location of Destroyers

What prevents the directed edge (p, q) from existing in G_λ^θ? It is the existence of one point acting as a destroyer. Given two points p, q, where can a point lie such that it acts as the destroyer of the edge (p, q)? In this subsection, we describe the region containing the points r such that $q \in K(p, r)$. This region is denoted $\overline{K}(p, q)$. In other words, $\overline{K}(p, q)$ is the description of all the locations of possible destroyers of an edge (p, q).

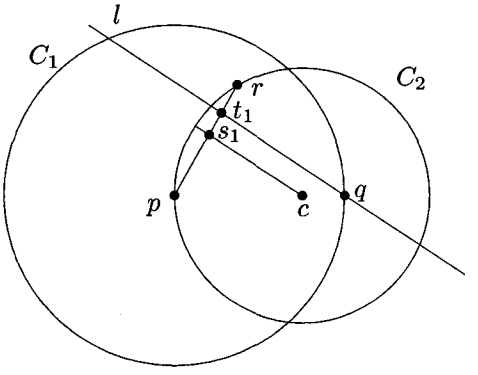


Figure 4.2: The location of a point r destroying the edge (p, q).

Proposition 4.6 *Let $R(p, q, \lambda)$ be the intersection of the disks C_1 centered at p with radius $|pq|$ and C_2 centered at $c = p + \lambda(q - p)$ with radius $\lambda|pq|$. If $q \in K(p, r)$ and $|pr| \leq |pq|$, then $r \in R(p, q, \lambda)$.*

Proof: If r destroyed (p, q), then $|pr| \leq |pq|$. Therefore, r is in C_1. To complete the proof, we need to show that r is in C_2. We begin by considering the case when q lies on the line l which is the boundary of λ-half-plane(p, r) (see Figure 4.2). Let s_1 be

the midpoint of $\overline{pr}$, t_1 the intersection of l with $\overline{pr}$ and c' the intersection of $\overline{pq}$ with the bisector of $\overline{pr}$. Since the triangles $\triangle pt_1q$ and $\triangle ps_1c'$ are similar, this implies that

$$|pc'| = |pq|\frac{|ps_1|}{|pt_1|} = |pq|\frac{|pr|}{2|pt_1|} = |pq|\frac{2\lambda|pr|}{2|pr|} = \lambda|pq| = |pc|.$$

Therefore, $c' = c$, which implies that $|cr| = |cp|$ thereby proving that r is on the boundary of C_2.

In the case when q is not on l, then we have $|pc'| < |pc|$ and r lies on a circle centered at c' going through p. Therefore, r is contained in C_2, which completes the proof. □

The following proposition follows directly from the definition of $K(p, r)$.

Proposition 4.7 *If $q \in K(p, r)$, then $\angle qpr \leq \theta$.*

Combining Proposition 4.6 and Proposition 4.7, we get:

Proposition 4.8 *Let $\overline{K}(p, q)$ be the intersection of $R(p, q, \lambda)$ with the θ-cone(p, q). If $q \in K(p, r)$ and $|pr| \leq |pq|$, then $r \in \overline{K}(p, q)$.*

4.3.2 Spanning Ratio

Theorem 4.9 *For $0 \leq \theta < \frac{\pi}{2}$ and $\frac{1}{2} < \lambda < 1$, the G^θ_λ graph is a strong t-spanner, with $t = \frac{1}{(1-\lambda)\cos\theta}$.*

Proof: Let P be a set of points in the plane, $p, q \in P$ and $d_G(p, q)$ be the length of the shortest path from p to q in $G^\theta_\lambda(P)$. We show by induction on the rank of the distance $|pq|$ that $d_G(p, q) \leq t|pq|$.

Base case: If p and q form a closest pair, then the edge (p, q) is in $G^\theta_\lambda(P)$ by definition. Therefore, $d_G(p, q) = |pq| \leq t|pq|$.

Inductive case: If the edge (p, q) is in $G^\theta_\lambda(P)$, then $d_G(p, q) = |pq| \leq t|pq|$ as required. We now address the case when (p, q) is *not* in $G^\theta_\lambda(P)$. By Proposition 4.8,

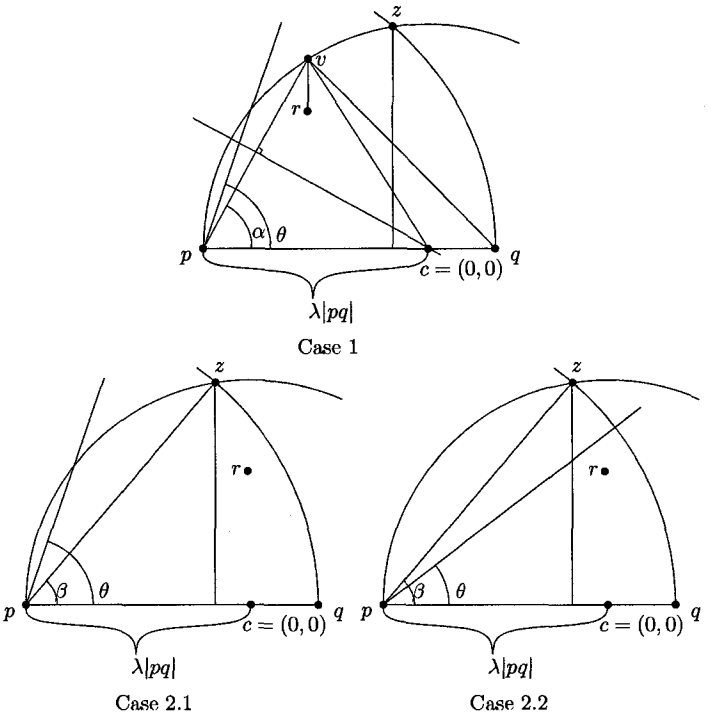


Figure 4.3: Cases for the proof of Theorem 4.9.

there must be a point $r \in \overline{K}(p,q) \cap P$ that is destroying (p,q) and that the edge (p,r) is in $G_\lambda^\theta(P)$. Since $r \in \overline{K}(p,q)$ and $|pr| < |pq|$, we have that $|rq| < |pq|$. By the inductive hypothesis, we have $d_G(r,q) \leq t|rq|$.

Let z be the intersection of the boundaries of the disks C_1 and C_2 defined in Proposition 4.6. We assume, without loss of generality, that c is the origin and that points p, q are on the x-axis with p to the left of q as depicted in Figure 4.3. The remainder of the proof addresses two cases, depending on whether or not $r_x \leq z_x$ (the notation p_x denotes the x-coordinate of a point p).

Case 1: $r_x \leq z_x$. Let $v \in \overline{K}(p,q)$ be the point with the same x-coordinate as r and having the greatest y-coordinate. In other words, v is the highest point in $\overline{K}(p,q)$ that is strictly above r. We have:

$$\begin{aligned} d_G(p,q) &\leq |pr| + d_G(r,q) \\ &\leq |pr| + t|rq| \qquad \text{(ind. hyp.)} \\ &\leq |pv| + t|vq|. \end{aligned}$$

Now, let $\alpha = \angle vpq \leq \theta$ We express $|pv|$ and $|vq|$ as a function of $\cos\alpha$. Consider the triangle $\triangle(pvc)$ and note that $|vc| = |pc|$ by construction.

$$\begin{aligned} \cos\alpha &= \frac{|pv|}{2\lambda|pq|} \\ |pv| &= 2\lambda|pq|\cos\alpha \end{aligned}$$

and from the law of cosines, we have:

$$\begin{aligned} |vq|^2 &= |pv|^2 + |pq|^2 - 2|pv||pq|\cos\alpha \\ &= 4\lambda^2|pq|^2\cos^2\alpha + |pq|^2 - 4\lambda|pq|^2\cos^2\alpha \\ &= |pq|^2(4\lambda^2\cos^2\alpha - 4\lambda\cos^2\alpha + 1) \end{aligned}$$

which implies that:

$$\begin{aligned} d_G(p,q) &\leq 2\lambda|pq|\cos\alpha + t|pq|\sqrt{4\lambda^2\cos^2\alpha - 4\lambda\cos^2\alpha + 1} \\ &= |pq|(2\lambda\cos\alpha + t\sqrt{4\lambda^2\cos^2\alpha - 4\lambda\cos^2\alpha + 1}). \end{aligned}$$

Therefore, we have to show that:

$t \geq 2\lambda\cos\alpha + t\sqrt{4\lambda^2\cos^2\alpha - 4\lambda\cos^2\alpha + 1}$, which can be rewritten as

$$t \geq \frac{2\lambda\cos\alpha}{1 - \sqrt{4\lambda^2\cos^2\alpha - 4\lambda\cos^2\alpha + 1}}$$

and since $\alpha \leq \theta < \pi/2$ implies $\cos\theta \leq \cos\alpha$, by straightforward algebraic manipulation we have that:

$$\frac{1}{(1-\lambda)\cos\alpha} \geq \frac{2\lambda\cos\alpha}{1 - \sqrt{4\lambda^2\cos^2\alpha - 4\lambda\cos^2\alpha + 1}}$$

Case 2: $r_x > z_x$. Let $\beta = \angle zpq$. We first compute the value of $\cos\beta$. From the definition of C_1 and C_2, we have

$$z_x^2 + z_y^2 = \lambda^2|pq|^2$$

and

$$(z_x - p_x)^2 + z_y^2 = |pq|^2.$$

Therefore, since $p_x = -\lambda|pq|$, we have $z_x = \frac{|pq|(1-2\lambda^2)}{2\lambda}$ which implies

$$\cos\beta = \frac{\lambda|pq| + z_x}{|pq|} = \lambda + \frac{1-2\lambda^2}{2\lambda} = \frac{1}{2\lambda}.$$

We need to consider two subcases, depending on whether or not $\beta \leq \theta$.

Case 2.1: $\beta \leq \theta$. In this case, we have:

$$\begin{aligned} d_G(p,q) &\leq |pr| + d_G(r,q) \\ &\leq |pr| + t|rq| \qquad \text{(ind. hyp.)} \\ &\leq |pz| + t|zq| \\ &= |pq| + t|zq| \end{aligned}$$

By the law of cosines, we have:

$$|zq|^2 = |pq|^2(2 - \frac{1}{\lambda})$$

which implies:

$$d_G(p,q) \leq |pq| + t|pq|\sqrt{2 - \frac{1}{\lambda}}.$$

Therefore, we have to show that:

$$t \geq \frac{1}{1 - \sqrt{2 - \frac{1}{\lambda}}}.$$

Since $\beta \leq \theta$, we have $\cos\beta \geq \cos\theta$, and then:

$$\begin{aligned} t &= \frac{1}{(1-\lambda)\cos\theta} \\ &\geq \frac{1}{(1-\lambda)\cos\beta} \\ &= \frac{1}{(1-\lambda)(1/2\lambda)} \\ &= \frac{2\lambda}{1-\lambda} \\ &\geq \frac{1}{1 - \sqrt{2 - \frac{1}{\lambda}}} \text{ iff } (\lambda - 1)^2 \geq 0 \end{aligned}$$

Case 2.2: $\beta > \theta$. By the law of cosines we have:

$$d_G(p,q) \leq |pq| + t|pq|\sqrt{2-2\cos\theta}$$

which means that

$$t \geq \frac{1}{1-\sqrt{2-2\cos\theta}}$$

But $\frac{1}{1-\sqrt{2-2\cos\theta}} \geq \frac{1}{(1-\lambda)\cos\theta}$ since $\beta > \theta$ implies $\cos\theta > \frac{1}{2\lambda}$. This completes the last case of the induction step. Note that the resulting t-spanning paths found in this inductive proof are strong since both $|pr|$ and $|rq|$ are shorter than $|pq|$. □

Notice that the definition of the G^θ_λ graph extends to $\mathbb{R}^d$ for any $d > 2$. In that case, the cone in Definition 4.1 is a d-dimensional cone and the half-plane in Definition 4.2 is a d-dimensional half-space. Moreover, the statement and the proof of Theorem 4.9 are exactly the same: let P be a set of points in $\mathbb{R}^d$, p and q be two points in P. If the edge (p,q) is in G^θ_λ, then we are done. Otherwise, there is a point $r \in P$ that is destroying the edge (p,q). Let Π be a two-dimensional plane that contains p, q and r. The proof of the inductive step of Theorem 4.9 applies in Π. Therefore, we have the following corollary:

Corollary 4.10 *For $0 \leq \theta < \frac{\pi}{2}$, $\frac{1}{2} < \lambda < 1$ and $d \geq 2$, the G^θ_λ graph of a set of points in $\mathbb{R}^d$ is a strong t-spanner, with $t = \frac{1}{(1-\lambda)\cos\theta}$.*

The above proof provides a simple local routing algorithm. A routing algorithm is considered local provided that the only information used to make a decision is the 1-neighborhood of the current node as well as the location of the destination (see [15] for a detailed description of the model). The routing algorithm proceeds as follows. To find a path from p to q, if the edge (p,q) is in $G^\theta_\lambda(P)$, then take the edge. If the edge (p,q) is not in $G^\theta_\lambda(P)$, then take an edge (p,r) where r is a destroyer of the edge (p,q). Recall that r is a destroyer of the edge (p,q) if $r \in \overline{K}(p,q)$. This can be computed solely with the positions of p, q and r. Therefore, determining which of the neighbors of p in $G^\theta_\lambda(P)$ destroyed the edge (p,q) is a local computation.

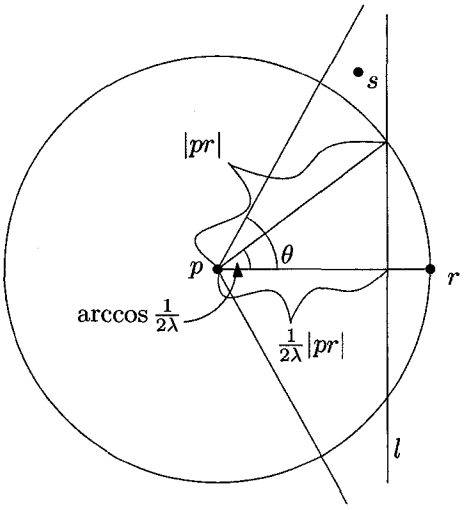


Figure 4.4: The G_λ^θ graph has bounded out-degree.

4.3.3 Bounded Out-Degree

Theorem 4.11 *The out-degree of a node in the G_λ^θ graph defined on points in the plane is at most $\lfloor 2\pi/\min(\theta, \arccos\frac{1}{2\lambda})\rfloor$.*

Proof: Let (p, r) and (p, s) be two edges of the G_λ^θ graph. Without loss of generality, $|pr| \leq |ps|$. Let l be the line perpendicular to $\overline{pr}$ through $p + \frac{1}{2\lambda}(r - p)$. Then either $\angle spr \geq \theta$ or s lies on the same side of l as p. In the latter case, the angle $\angle spr$ is at least $\arccos\frac{1}{2\lambda}$ (see Figure 4.4). The angle $\angle spr$ is then at least $\min(\theta, \arccos\frac{1}{2\lambda})$, which means that p has at most $\lfloor 2\pi/\min(\theta, \arccos\frac{1}{2\lambda})\rfloor$ outgoing edges. □

Corollary 4.12 *If $\theta \geq \pi/3$ and $\lambda > \frac{1}{2\cos(2\pi/7)}$, then the out-degree of a node in the G_λ^θ graph defined on points in the plane is at most six.*

In higher dimensions, the statement of Theorem 4.11 also holds, except that the expression for the bounded degree depends not only on λ and θ, but also on the dimension. The key observation that has to be made is that the angle between two edges that are outgoing from a same point still form an angle of at least $\min(\theta, \arccos\frac{1}{2\lambda})$. Therefore, Theorem 4.13 provides an upper bound on the out-degree of a node in the G_λ^θ graph defined on points in $\mathbb{R}^d$.

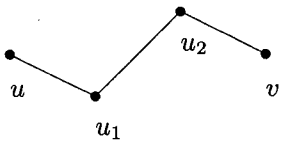


Figure 4.5: Counter-example to the proof of Theorem 2 of [20].

Theorem 4.13 [4] *Let $d \geq 2$ be an integer constant, let ϕ be a real number such that $0 < \phi < \pi$, and let P be a set of points in $\mathbb{R}^d \setminus \{0\}$, such that* $\text{angle}(q, r) > \phi$ *for any two distinct points q and r in P. The size of P is $O(1/\phi^{d-1})$.*

Corollary 4.14 *The out-degree of a node in the G^{θ}_{λ} graph defined on points in $\mathbb{R}^d$ is at most $O(1/\phi^{d-1})$, where $\phi = \min(\theta, \arccos \frac{1}{2\lambda})$.*

4.4 Half-Space Proximal

In this section, we review inconsistencies within statements of the proof of the upper and lower bounds of HSP given in Chavez *et al.* [20].

In the proof of the upper bound (Theorem 2 of Chavez *et al.* [20]), claim 4 states that in HSP, for every pair of nodes u and v, if (u, v) is not an edge of HSP and no neighbor of u is adjacent to v, then there is a path $u = u_0, u_1, u_2, \ldots, u_{k+1} = v$ such that for every i, $0 \leq i \leq k-1$, the nodes are either in clockwise or anticlockwise order around v. However, as stated, this is not true. A counter-example to this claim is shown in Figure 4.5. There is a unique path from u to v, namely uu_1u_2v, but this path is neither clockwise nor counter-clockwise around v. We believe that this situation may exist *in the worst case.* However, a characterization of the worst case situation must be given and it must be proven that the worst case situation has the claimed property.

Another property to note is that the shortest path between two points in HSP is not necessarily a strong path. Figure 4.6 shows an example where the shortest path from p_1 to p_2 (depicted in bold) is not strong since it contains the edge (p_6p_2), which is longer than $|p_1p_2|$. Also, this path is not monotone in the direction $\overrightarrow{p_1p_2}$. Furthermore, the unique path from p_1 to p_2 that is getting closer to p_2 in each step

[4] Theorem 5.3.1 of [65], where 0 is the origin and $\text{angle}(q, r)$ is the angle between q and r with the origin as apex.

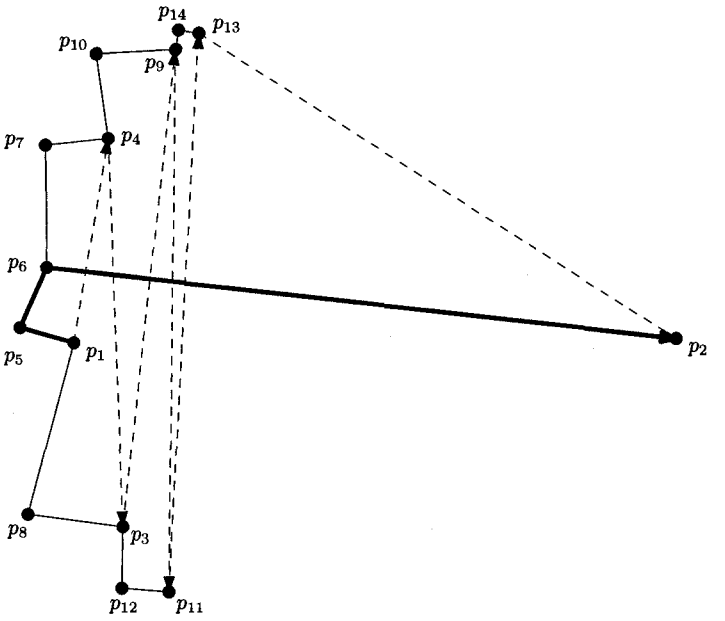


Figure 4.6: The shortest path from p_1 to p_2 is not strong (the HSP for these points consists of the solid and the dashed edges).

(depicted in dashed line segments) has spanning ratio of at least 4.5. The HSP for these points consists of the solid and the dashed edges. Even though Chavez *et al.* do not state how to locally find a strong spanning path, Figure 4.6 shows the difficulty of this problem when using HSP. The precise location of each point is given in Table 4.4.

For the lower bound, the authors also claim that the spanning ratio of HSP can be arbitrarily close to $2\pi + 1$. However, the proof they provide to support that claim is a construction depicted in Figure 4.7 (reproduced from [20]). The solid edges are

point	x-Coor.	y-Coor.		point	x-Coor.	y-Coor.
p_1	333	435		p_8	297	569
p_2	804	432		p_9	413	206
p_3	371	579		p_{10}	351	209
p_4	360	275		p_{11}	407	630
p_5	291	423		p_{12}	370	627
p_6	312	376		p_{13}	431	193
p_7	311	280		p_{14}	415	191

Table 4.1: Precise location of the points in Figure 4.6.

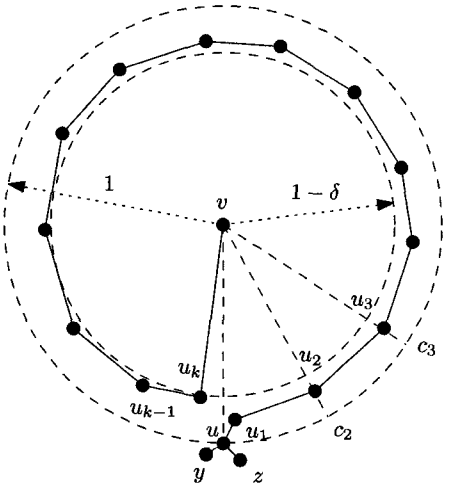


Figure 4.7: The illustration of the lower bound on the spanning ratio of [20].

in HSP, and the dashed edges are not in HSP. The claim is that the path from u to v can have length arbitrarily close to $(2\pi + 1)|uv|$. Although this may be true for the path that they highlight, this path is *not* the only path from u to v in HSP. The authors neglected the presence of the edge (u_1u_k) in their construction, which provides a shortcut that makes the distance between u and v much less than $2|uv|$. For a lower bound, every path from u to v must have length at least $(2\pi + 1)|uv|$.

One of the main reasons we believe the claims made in Chavez *et al.* [20] may be true is that in the simulations, all the graphs have small spanning ratio. In fact, the spanning ratio seems to be even smaller than $2\pi + 1$. However, at this point, no proof that HSP is a constant spanner is known. We provide a lower bound of $3 - \epsilon$ on the spanning ratio of HSP as depicted in the construction below.

Proposition 4.15 *For every $\epsilon > 0$, the HSP graph has spanning ratio at least $3 - \epsilon$.*

Proof: Consider the set of 6 points as in Figure 4.8, put $\delta = \epsilon/6$. The length of the path between p and q via a and b is equal to the length of the path between p and q via c and d. The length of both of these paths is $3 - 6\delta$. Since the shortest path between p and q in the HSP graph is one of the above paths, the spanning ratio is $3 - 6\delta = 3 - \epsilon$. □

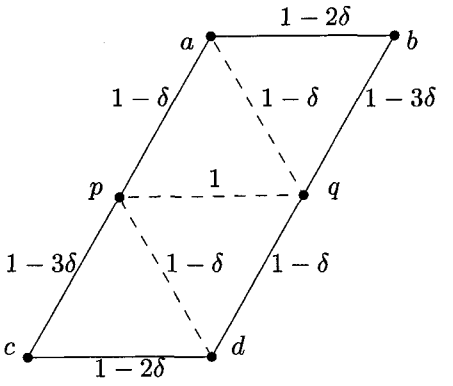


Figure 4.8: Example of a 6 nodes HSP with a spanning ratio of $3-\epsilon$. The solid edges are in HSP.

4.5 Unit Disk Graph Spanners

In Section 4.3, we showed that the G_λ^θ graph of a set of points in the plane is a strong t-spanner of the complete graph of these points, for a constant $t = \frac{1}{(1-\lambda)\cos\theta}$. We show in this section that strong t-spanners are also spanners of the unit disk graph. That is, the length of the shortest path between a pair of points in the graph resulting from the intersection of a strong t-spanner with the unit disk graph is not more than t times the length of the shortest path in the unit disk graph. Before proceeding, we need to introduce some notation.

For simplicity of exposition, we assume that given a set P of points in the plane, no two pairs of points are at equal distance from each other. The *complete geometric graph* defined on a set P of points, denoted $C(P)$, is the graph whose node set is P and whose edge set is $P \times P$, with each edge having its weight equal to the Euclidian distance between its nodes. Let $e_1, \ldots, e_{\binom{n}{2}}$ be the edges of $C(P)$ sorted according to their lengths $L_1, \ldots, L_{\binom{n}{2}}$. For $i = 1 \ldots \binom{n}{2}$, we denote by $C_i(P)$ the geometric graph consisting of all edges whose length is no more than L_i. In general, for any graph G whose node set is V, we define G_i as $G \cap C_i(V)$. Let UDG(P) be the unit disk graph of P, which is the graph whose node set is P and with edges between pairs of nodes whose distance is not more than one. Note that UDG(P) $= C_i(P)$ for some i. We now show the relationship between strong t-spanners and unit disk graphs.

Proposition 4.16 *If S is a strong t-spanner of $C(P)$, then for all $i = 1 \ldots \binom{n}{2}$ and*

all $j = 1 \ldots i$, S_i contains a t-spanning path linking the nodes of $e_j = (p_j q_j)$.

Proof: Consider a strong t-spanner path in S between p_j and q_j. Each edge on this path has length at most $|p_j q_j| = L_j \leq L_i$. Therefore, each edge is in S_i. □

Proposition 4.17 *If S is a strong t-spanner of $C(P)$, then for all $i = 1 \ldots \binom{n}{2}$, S_i is a t-spanner of $C_i(P)$.*

Proof: Let a and b be any two points such that $d_{C_i(P)}(a, b)$ is finite. We need to show that in S_i there exists a path whose length is at most $t \cdot d_{C_i(P)}(a, b)$. Let $a = p_1, p_2, \ldots, p_k = b$ be a shortest path in $C_i(P)$ between a and b. Hence:

$$d_{C_i(P)}(a, b) = \sum_{j=1}^{k-1} |p_j p_{j+1}|$$

Now, by Proposition 4.16, for each edge (p_j, p_{j+1}) there is a path in S_i between p_j and p_{j+1} whose length is at most $t \cdot |p_j p_{j+1}|$. Therefore:

$$d_{S_i(P)}(a, b) \leq \sum_{j=1}^{k-1} t \cdot |p_j p_{j+1}| = t \sum_{j=1}^{k-1} |p_j p_{j+1}| = t \cdot d_{C_i(P)}(a, b)$$

which means that in S_i, there exists a path whose length is at most $t \cdot d_{C_i(P)}(a, b)$. □

Corollary 4.18 *If S is a strong t-spanner of $C(P)$, then $S \cap \mathrm{UDG}(P)$ is a strong t-spanner of* $\mathrm{UDG}(P)$.

Proof: Just notice that $\mathrm{UDG} = C_i$ for some i and the result follows from Proposition 4.17. □

Thus, we showed sufficient conditions for a graph to be a spanner of the unit disk graph. We now show that these conditions are also necessary.

Proposition 4.19 *If S is a subgraph of $C(P)$ such that for all $i = 1 \ldots \binom{n}{2}$, S_i is a t-spanner of $C_i(P)$, then S is a strong t-spanner of $C(P)$.*

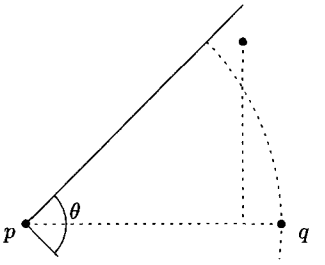


Figure 4.9: The θ-graph is not a strong t-spanner.

Proof: Let a, b be any pair of points chosen in P. We have to show that in S, there is a path between a and b such that:

1. its length is at most $t \cdot |ab|$ and

2. every edge on the path has length at most $|ab|$.

Let $e_i = (a, b)$. We know that S_i is a t-spanner of $C_i(P)$. Since $C_i(P)$ contains e_i, $d_{C_i(P)}(a, b) = |ab|$. Hence, there is a path in S_i (and therefore in S) whose length is at most $t \cdot d_{C_i(P)}(a, b) = t|ab|$. Also, since it is in S_i, all of its edges have length at most $L_i = |ab|$. □

The two last results, together, allow us to determine whether or not given families of geometric graphs are also spanners of the unit disk graph. First, since the G^{θ}_{λ} graph is a strong t-spanner, we already know that it is also a spanner of the unit disk graph. Second, Bose *et al.* [14] showed that the Yao graph [91] and the Delaunay triangulation are strong t-spanners. Therefore, these graphs are also spanners of the unit disk graph. Third, the θ-graph [48] is not always a spanner of the unit disk graph. The reason for that is that in a cone, the edge you chose may not be the shortest edge. Hence, the path from a point p to a point q may contain edges whose length is greater than $|pq|$ (see Figure 4.9). Using Proposition 4.19, we thus know that the intersection of the θ-graph with the unit disk graph may not be a spanner of the unit disk graph. Indeed, the intersection of the θ-graph with the unit disk graph may not even be connected.

4.6 Simulation Results

In Section 4.3, we provided worst-case analysis of the spanning ratio of the G_λ^θ graph. Using simulation, we now provide estimates of the average spanning ratio of the G_λ^θ graph. Using a uniform distribution, we generated 200 sets of 200 points each and computed the spanning ratio for λ ranging from 0.5 to 1 and θ ranging from 5° to 90° (for $\theta = 0°$, the spanning ratio is exactly 1). For each graph, we then computed the spanning ratio and the local routing ratio. The *spanning ratio* is defined as the maximum, over all pair of points (p, q), of the length of the shortest path from p to q in the G_λ^θ graph divided by $|pq|$. The *local routing ratio* is defined as the maximum, over all pair of points (p, q), of the length of the path produced by using a local routing strategy in the G_λ^θ graph divided by $|pq|$. The local routing strategy we used is the following: at each step, send the message to the neighbor that destroyed q. We also tried the strategy that consists in choosing the neighbor that is the nearest to q, and the results we obtained were the same.

Figure 4.10 and Table 4.2 show the results we obtained for the spanning ratio. Figure 4.11 and Table 4.3 show the results we obtained for the local routing ratio. For the spanning ratio, the 95% confidence interval for these values is ± 0.0319. For the local routing ratio, the 95% confidence interval is ± 0.0735.

One interesting conclusion we can draw from these results is that for the spanning ratio, θ has a more decisive influence than λ. Figure 4.12 shows the simulation results for the cases where $\lambda = 0.75$. We see that even though both ratios generally increase when θ increases, the spanning ratio varies between 1.07 and 2.21 (107% variation), while the local routing ratio only varies between 2.33 and 2.77 (19% variation). For the local routing ratio, it is the other way around. It is λ that has a more decisive influence. Figure 4.13 shows the influence of λ when $\theta = 45°$. In that case, the local routing ratio varies between 1.72 and 4.55 (165% variation), while the spanning ratio only varies between 1.52 and 1.81 (19% variation).

4.7 Conclusion

While trying to find a valid proof for the spanning ratio of the Half-Space Proximal graph, we found a new family of parametric strong spanners that admit a local routing strategy. To prove that the Half-Space Proximal graph has a constant spanning ratio is still an open problem.

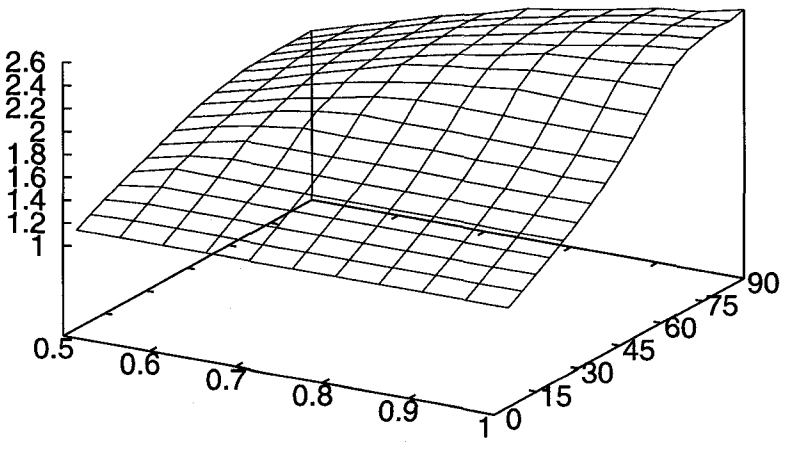


Figure 4.10: Spanning Ratio for $\lambda = 0.5$ to 1 and $\theta = 5°$ to 90°.

$\theta \setminus \lambda$	0.5	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1
5	1.07	1.07	1.07	1.07	1.07	1.07	1.07	1.07	1.07	1.07	1.07
10	1.14	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15
15	1.20	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23	1.23
20	1.26	1.31	1.31	1.31	1.31	1.31	1.31	1.31	1.31	1.31	1.31
25	1.31	1.40	1.40	1.39	1.40	1.40	1.39	1.40	1.40	1.39	1.39
30	1.36	1.45	1.49	1.48	1.48	1.48	1.48	1.48	1.48	1.48	1.48
35	1.41	1.50	1.58	1.58	1.58	1.58	1.59	1.58	1.58	1.58	1.59
40	1.46	1.57	1.64	1.69	1.69	1.70	1.71	1.70	1.70	1.68	1.69
45	1.52	1.60	1.71	1.78	1.81	1.81	1.81	1.81	1.80	1.81	1.81
50	1.56	1.65	1.75	1.82	1.90	1.94	1.95	1.95	1.95	1.94	1.95
55	1.59	1.69	1.80	1.89	1.96	2.02	2.06	2.11	2.10	2.09	2.09
60	1.61	1.72	1.83	1.94	2.05	2.10	2.15	2.21	2.22	2.25	2.25
65	1.65	1.75	1.86	1.95	2.05	2.14	2.20	2.28	2.31	2.34	2.39
70	1.66	1.77	1.88	1.99	2.09	2.16	2.24	2.29	2.36	2.42	2.46
75	1.66	1.77	1.88	2.00	2.09	2.18	2.26	2.34	2.39	2.45	2.50
80	1.67	1.79	1.88	1.99	2.10	2.20	2.27	2.36	2.42	2.47	2.52
85	1.66	1.78	1.89	2.00	2.10	2.19	2.27	2.35	2.42	2.47	2.51
90	1.67	1.78	1.89	2.00	2.09	2.21	2.26	2.33	2.41	2.46	2.54

Table 4.2: Spanning Ratio for $\lambda = 0.5$ to 1 and $\theta = 5°$ to 90°.

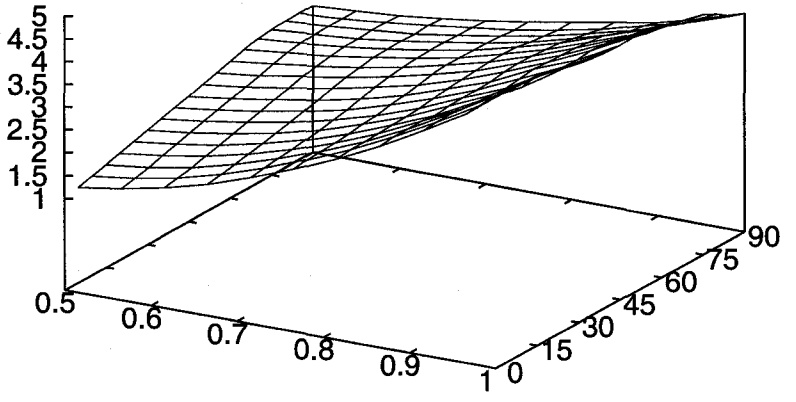


Figure 4.11: Local Routing Ratio for $\lambda = 0.5$ to 1 and $\theta = 5°$ to 90°.

$\theta \setminus \lambda$	0.5	0.55	0.60	0.65	0.70	0.75	0.80	0.85	0.90	0.95	1
5	1.08	1.22	1.41	1.65	1.97	2.33	2.72	3.17	3.68	4.22	4.92
10	1.15	1.27	1.45	1.70	2.01	2.36	2.76	3.21	3.70	4.23	4.90
15	1.23	1.35	1.51	1.75	2.04	2.38	2.76	3.21	3.72	4.27	4.90
20	1.31	1.43	1.59	1.81	2.07	2.40	2.77	3.21	3.67	4.22	4.84
25	1.39	1.51	1.66	1.87	2.12	2.43	2.79	3.17	3.65	4.13	4.73
30	1.48	1.59	1.74	1.92	2.17	2.48	2.80	3.21	3.63	4.19	4.66
35	1.56	1.68	1.82	2.00	2.23	2.50	2.83	3.23	3.60	4.14	4.60
40	1.64	1.76	1.90	2.08	2.29	2.56	2.87	3.23	3.65	4.07	4.58
45	1.72	1.84	1.99	2.15	2.37	2.61	2.92	3.24	3.69	4.06	4.55
50	1.81	1.92	2.07	2.25	2.45	2.66	2.95	3.30	3.67	4.10	4.58
55	1.90	2.01	2.17	2.33	2.51	2.73	2.98	3.31	3.66	4.08	4.50
60	1.99	2.10	2.23	2.37	2.55	2.77	3.00	3.31	3.65	4.03	4.45
65	2.08	2.18	2.29	2.44	2.58	2.77	3.01	3.28	3.61	3.96	4.40
70	2.15	2.23	2.33	2.45	2.60	2.77	3.00	3.26	3.59	3.92	4.31
75	2.18	2.26	2.34	2.46	2.60	2.75	2.94	3.19	3.46	3.80	4.24
80	2.20	2.28	2.36	2.44	2.58	2.74	2.91	3.13	3.37	3.67	4.08
85	2.22	2.27	2.36	2.47	2.58	2.70	2.87	3.05	3.32	3.55	3.87
90	2.21	2.27	2.34	2.45	2.57	2.72	2.88	3.05	3.22	3.47	3.76

Table 4.3: Local Routing Ratio for $\lambda = 0.5$ to 1 and $\theta = 5°$ to 90°.

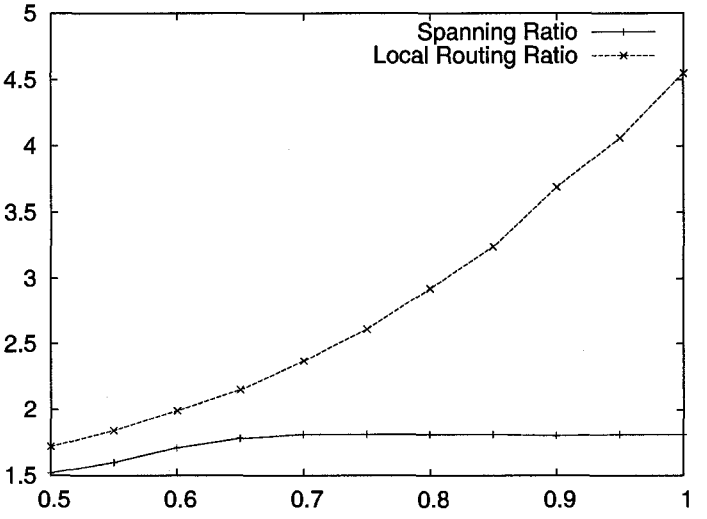


Figure 4.12: Ratios for $\theta = 45°$.

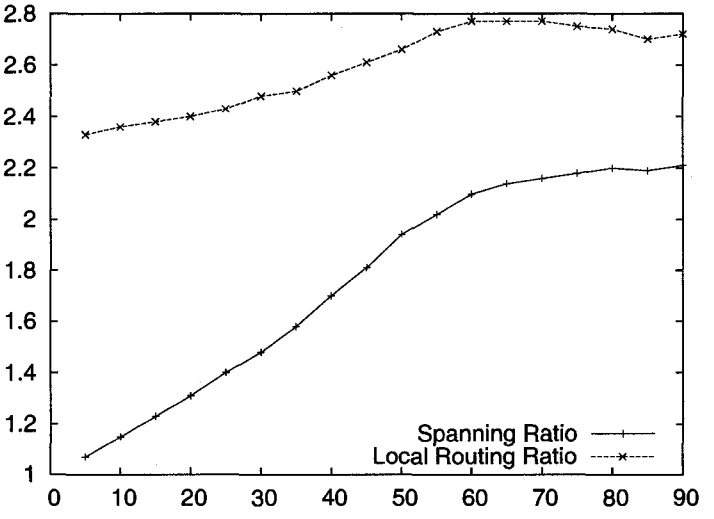


Figure 4.13: Ratios for $\lambda = 0.75$.

Chapter 5

Geometric Spanners With Small Chromatic Number

5.1 Introduction

In a recent paper, Raman and Chebrolu [74] proposed a new protocol, called 2P, allowing to address rural Internet connectivity in a low-cost manner using off-the-shelf 802.11 hardware. Since their infrastructure uses several directional antennae at one node rather than one single omnidirectional antenna, simultaneous communications are possible at one node. However, due to restrictions inherent in the 802.11 standard, backbone nodes have to communicate with each other using a single channel. While simultaneous transmissions and simultaneous receptions are possible, it is not physically possible for one node to both transmit and receive at the same time [74]. Therefore, backbone nodes have to alternate between the send and receive states (see Figure 5.1). This forces the backbone to be a bipartite graph, i.e., to have chromatic number two.

The backbone creation algorithm of Raman and Chebrolu [74] outputs a tree, which is obviously bipartite. However, the tree structure presents the following disadvantage: it is possible that the path that a message has to follow is much longer than the distance (either Euclidean or in terms of hops) between the originating node and its destination. For example, in Figure 5.1, a message routed from node 1 to node 3 has to go through nodes 2 and 4, whereas a direct link between 1 and 3 could be added while still satisfying the bipartition requirement.

Note that the physical constraint preventing nodes to simultaneously receive and transmit can be met even if the graph is not bipartite. In fact, any graph with chromatic number k would meet this requirement: all one has to do is to prevent two nodes that have different colors from transmitting simultaneously. A degenerate case is when each node has its own color, in which case at most one node can transmit at any given moment. This is undesirable, since the amount of time during which a

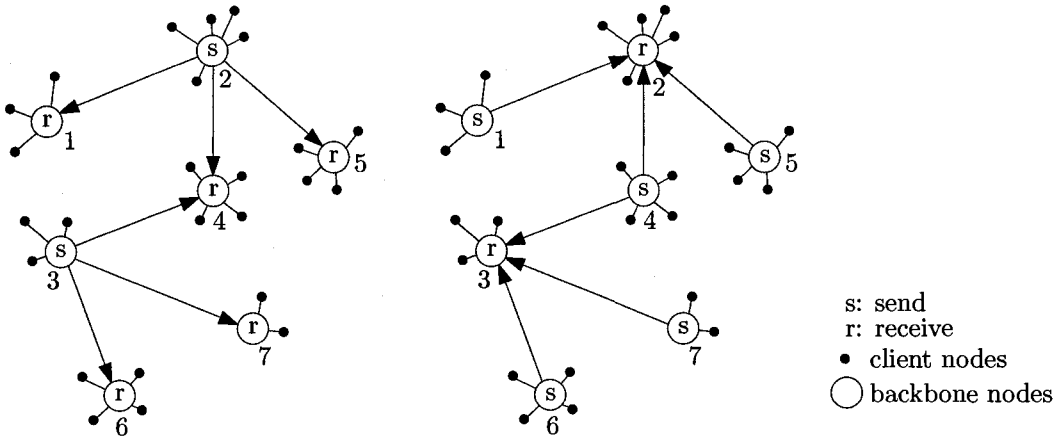


Figure 5.1: The two possible states of the backbone nodes.

node can transmit decreases as the size of the network increases.

For these reasons, it is desirable to have geometric graphs that have both small chromatic number and small spanning ratio. In this chapter, we consider the problem of computing t-spanners whose chromatic number is at most k, for some given value of k. The goal is to minimize the value of t over all finite sets P of points in the plane. We call a spanner whose chromatic number is at most k a *k-chromatic spanner.*

Problem 5.1 *Given an integer $k \geq 2$, let $t(k)$ be the infimum of all real numbers t with the property that for every finite set P of points in the plane, a k-chromatic t-spanner for P exists. Determine the value of $t(k)$.*

Observe that in the definition of $t(k)$, there is no requirement on the number of edges of the chromatic spanner. This is not a restriction, because, as shown by Gudmundsson *et al.* [39], any t-spanner for P contains a subgraph with $O(n)$ edges which is a $((1+\epsilon)t)$-spanner for P.

We show how to obtain a 2-chromatic 3-spanner for any point set P, thus showing that $t(2) \leq 3$. We also give an example of a point set P such that any 2-chromatic graph with vertex set P has spanning ratio at least three. Thus, we have $t(2) = 3$.

Next, we show how to compute a 3-chromatic 2-spanner of any point set P, thereby proving that $t(3) \leq 2$. We also show, by means of an example, that $t(3) \geq 2$. Thus, we obtain that $t(3) = 2$. For $k = 4$, we show how to compute a 4-chromatic $\sqrt{2}$-spanner of any point set P; thus $t(4) \leq \sqrt{2}$. Again by means of an example, we also

number of colors	$t(k)$ (off-line)		$t'(k)$ (on-line)	
k	lower bound	upper bound	lower bound	upper bound
2	3	3	3	3
3	2	2	$1+\sqrt{3}$	$1+\sqrt{3}$
4	$\sqrt{2}$	$\sqrt{2}$	$1+\sqrt{2}$	$1+\sqrt{2}$
k	$1/\cos\frac{\pi}{k+1}$	$1+2\sin\frac{\pi}{2(k-1)}$	$1/\cos\frac{\pi}{k}$	$1+2\sin\frac{\pi}{k}$

Table 5.1: Summary of our results.

show that $t(4) \geq \sqrt{2}$. Therefore, we have $t(4) = \sqrt{2}$.

For $k > 4$, we are not able to obtain the exact value of $t(k)$. Inspired by the *ordered Θ-graph* of Bose *et al.* [15], we show that $t(k) \leq 1+2\sin\frac{\pi}{2(k-1)}$. We also show that the vertex set of the regular $(k+1)$-gon gives $t(k) \geq 1/\cos\frac{\pi}{k+1}$.

In the second part of the chapter, we consider an on-line variant of the problem where the points of P are given one after another, and the color of a point must be assigned at the moment when the point is given; thus, later on, the color of a point cannot be changed. This makes the problem more difficult. Consequently, the bounds are higher, but still tight for $k = 2, 3, 4$. All our bounds are summarized in Table 5.1.

Problem 5.2 *Given an integer $k \geq 2$, let $t'(k)$ be the infimum of all real numbers t with the property that for every finite set P of points in the plane, which is given on-line, a k-chromatic t-spanner for P exists. Determine the value of $t'(k)$.*

A simple variant of the ordered Θ-graph shows that $t'(k) \leq 1+2\sin(\pi/k)$. Thus, we have $t'(2) \leq 3$, $t'(3) \leq 1+\sqrt{3}$ and $t'(4) \leq 1+\sqrt{2}$. Since $t'(2) \geq t(2) = 3$, it follows that $t'(2) = 3$. We also give examples showing that $t'(3) \geq 1+\sqrt{3}$ and $t'(4) \geq 1+\sqrt{2}$. We finally show that, for $k \geq 5$, $t'(k) \geq 1/\cos\frac{\pi}{k}$.

The rest of this chapter is organized as follows: in Section 5.2, we define the t-ellipse property and show its relationship to our problem. In Section 5.3, we give upper and lower bounds for the off-line problem (Problem 5.1). In Section 5.4, we give give upper and lower bounds for the on-line problem (Problem 5.2). In Section 5.5, we present simulation results. We conclude in Section 5.6. In Table 5.1, we summarize our results.

5.2 The t-Ellipse Property

In this section, we show that Problem 5.1, i.e., determining the smallest value of t such that a k-chromatic t-spanner exists for any point set P, is equivalent to minimizing the value of t such that any point set can be colored using k colors in a way that satisfies the so-called *t-ellipse property.*

Definition 5.3 (t-ellipse property) *Let $k \geq 2$ be an integer, let P be a finite set of points in the plane and let $c : P \to \{1, \ldots, k\}$ be a k-coloring of P. We say that the coloring c satisfies the* t-ellipse property *if, for each pair of distinct points p and q in P with $c(p) = c(q)$, there exists a point $r \in P$ such that $c(r) \neq c(p)$ and $|pr| + |rq| \leq t|pq|$.*

Thus, if p and q have the same color, then the ellipse $\{x \in \mathbb{R}^2 : |px| + |xq| \leq t|pq|\}$ contains a point r of P whose color is different from that of p and q.

Proposition 5.4 *Let $k \geq 2$, let P be a set of points in the plane, and let G be a k-chromatic t-spanner of P with k-coloring c. Then c satisfies the t-ellipse property.*

Proof: Let $p, q \in P$ be two points with $c(p) = c(q)$. Since G is a t-spanner, there exists a t-spanning path Π in G from p to q. Let r be the point on Π that is adjacent to p. Since the length of Π is at most $t|pq|$, we note that $|pr| + |rq|$ is at most $t|pq|$. Since the edge (p, r) is in G, it follows that $c(p) \neq c(r)$. Therefore, c satisfies the t-ellipse property. □

Proposition 5.5 *Let $k \geq 2$, let P be a set of points in the plane, and let $c : P \to \{1, \ldots, k\}$ be a k-coloring of P that satisfies the t-ellipse property. Then, there exists a k-chromatic t-spanner of P.*

Proof: Let $K_c(P)$ be the complete k-partite graph with vertex set P in which there is an edge between two points p and q if and only if $c(p) \neq c(q)$. By definition, $K_c(P)$ is k-colorable. We show that $K_c(P)$ is a t-spanner of P. Let p and q be two distinct points of P such that (p, q) is not an edge in $K_c(P)$. This means that $c(p) = c(q)$.

Since c has the t-ellipse property, there exists a point r in P such that $c(r) \neq c(p)$ and $|pr| + |rq| \leq t|pq|$. Since $c(r) \neq c(p)$ (and consequently, $c(r) \neq c(q)$), the edges (p, r) and (r, q) are both in $K_c(P)$. This means that (p, r, q) is a t-spanner path in $K_c(P)$ between p and q. □

From this point on, unless specified otherwise, we define the *spanning ratio* of a k-coloring of a point set to be the spanning ratio of the complete k-partite graph induced by this coloring. By Propositions 5.4 and 5.5, the problem of determining $t(k)$ is equivalent to determining the minimum spanning ratio of any k-coloring of any point set.

We conclude this section by showing why it is sufficient to focus on the coloring problem without worrying about the number of edges in the spanner. The following theorem is due to Gudmundsson *et al.* [39]; its proof is based on the well-separated pair decomposition of Callahan and Kosaraju [18].

Theorem 5.6 *[39] Let $\epsilon > 0$ and $t \geq 1$ be constants, let P be a set of n points in the plane, and let G be a t-spanner of P. There exists a subgraph G' of G, such that G' is a $((1+\epsilon)t)$-spanner of P and G' has $O(n)$ edges.*

Proposition 5.7 *Let $k \geq 2$, let P be a set of n points in the plane, and let $c : P \rightarrow \{1, \ldots, k\}$ be a k-coloring of P that satisfies the t-ellipse property. Then, for any constant $\epsilon > 0$, there exists a k-chromatic $((1+\epsilon)t)$-spanner of P that has $O(n)$ edges.*

Proof: By Proposition 5.5, there exists a k-chromatic t-spanner G of P. By Theorem 5.6, G contains a subgraph G' with $O(n)$ edges, such that G' is a $((1+\epsilon)t)$-spanner of P. Since G is k-chromatic, G' is k-chromatic as well. □

5.3 Upper and lower bounds on $t(k)$

The structure of this section is as follows: For $k = 2, 3$, and 4, we give coloring algorithms whose outputs have bounded spanning ratio. Then, we show that these

spanning ratios are tight by providing point sets for which no coloring algorithm can achieve a better spanning ratio. Then we present our upper and lower bounds for $t(k)$, when $k > 4$.

We now give the coloring algorithm for $k = 2$.

Algorithm 5.1: Offline 2 Colors

Input: P, a set of points in the plane

Output: c, a 2-coloring of P

1 Compute a Euclidean minimum spanning tree T of P;

2 $c \leftarrow$ a 2-coloring of T;

Proposition 5.8 *For any point set P, the 2-coloring computed by Algorithm 5.1 has spanning ratio at most 3. Thus, we have $t(2) \leq 3$.*

Proof: It is sufficient to show that the 2-coloring c computed by Algorithm 5.1 has the 3-ellipse property. Let p and q be two distinct points in P such that $c(p) = c(q)$. Observe that (p, q) is not an edge in the minimum spanning tree T. Let r be the nearest neighbor of p. Since the edge (p, r) is in T, we have $r \neq q$ and $c(r) \neq c(p)$. Since r is closer to p than q, we have

$$|pr| + |rq| \leq |pr| + |rp| + |pq| = 2|pr| + |pq| \leq 2|pq| + |pq| = 3|pq|.$$

□

Proposition 5.9 *For every $\epsilon > 0$, there exists a point set P such that every 2-coloring of P has spanning ratio at least $3 - \epsilon$. Thus, we have $t(2) \geq 3$.*

Proof: Let n be an odd integer, and let $P = \{p_1, \ldots, p_n\}$ be the set of vertices of a regular n-gon given in counter-clockwise order. Let c be an arbitrary 2-coloring of P. By the pigeonhole principle, there are two points in P which are adjacent on the n-gon and that have the same color. We may assume without loss of generality that these two points are p_1 and p_2. Also, we may assume that $|p_1p_2| = 1$ (see Figure 5.2). Let t be any real number such that c satisfies the t-ellipse property. Then

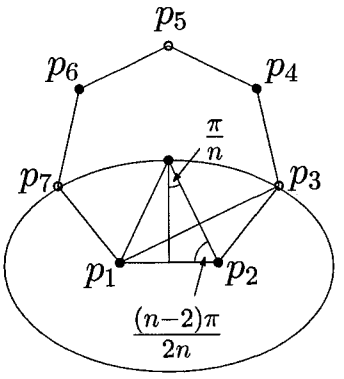


Figure 5.2: Lower bound of $3 - \epsilon$ for $k = 2$.

$|p_1p_3|+1 \leq t$. But $|p_1p_3| = 2\sin((n-2)\pi/2n)$, which tends to 2 as n goes to infinity. □

We now consider the case when $k = 3$. Our strategy is to construct a graph such that any coloring of that graph has the 2-ellipse property. We then show that this graph is 3-colorable.

Lemma 5.10 *The graph G computed by Algorithm 5.2 is triangle-free.*

Proof: Assume that G contains a triangle with vertices p, q, and r. We may assume without loss of generality that (p, r) was the last edge of this triangle that was considered by the algorithm. This means that (p, r) is the longest edge of the triangle. When $e_i = (p, r) = (p_i, q_i)$ in line 4, G already contains the edge (p, q). Since $|p_ip| + |pq_i| = |pp| + |pr| \leq 2|pr|$ and $|p_iq| + |qq_i| = |pq| + |qr| \leq 2|pr|$, the edge (p, r) is not added to G. This is a contradiction and, therefore, G is triangle-free. □

Proposition 5.11 *The graph G computed by Algorithm 5.2 is plane.*

Proof: Assume that G contains two crossing edges (p, q) and (s, t). We may assume without loss of generality that $s = (-1, 0)$, $t = (1, 0)$, and the pair (s, t) has a larger index than (p, q) after the pairs have been sorted in line 2. Thus, we have $|pq| \leq |st|$. When $e_i = (s, t) = (p_i, q_i)$ in line 4, G already contains the edge (p, q). Let E be the ellipse whose boundary is determined by the set of points e where $|se| + |et| = 2|st|$

Algorithm 5.2: Offline 3 Colors

Input: P, a set of n points in the plane

Output: c, a 3-coloring of P, and

G, a 3-chromatic graph whose vertex set is P

```
1  Let G be the graph with vertex set P and whose edge set is empty;
2  Let e_1, ..., e_{n choose 2} be the pairs of points of P in sorted order of their distances;
3  for i = 1 to (n choose 2) do
4  |  Let e_i = (p_i, q_i);
5  |  if G contains no edge (p, q) where |p_i p| + |p q_i| ≤ 2|p_i q_i| and
   |  |p_i q| + |q q_i| ≤ 2|p_i q_i| then
6  |  |  add the edge e_i to G;
7  |  end
8  end
9  //assertion: G is 3-colorable (see Lemma 5.12) ;
10 c ← a 3-coloring of G;
```

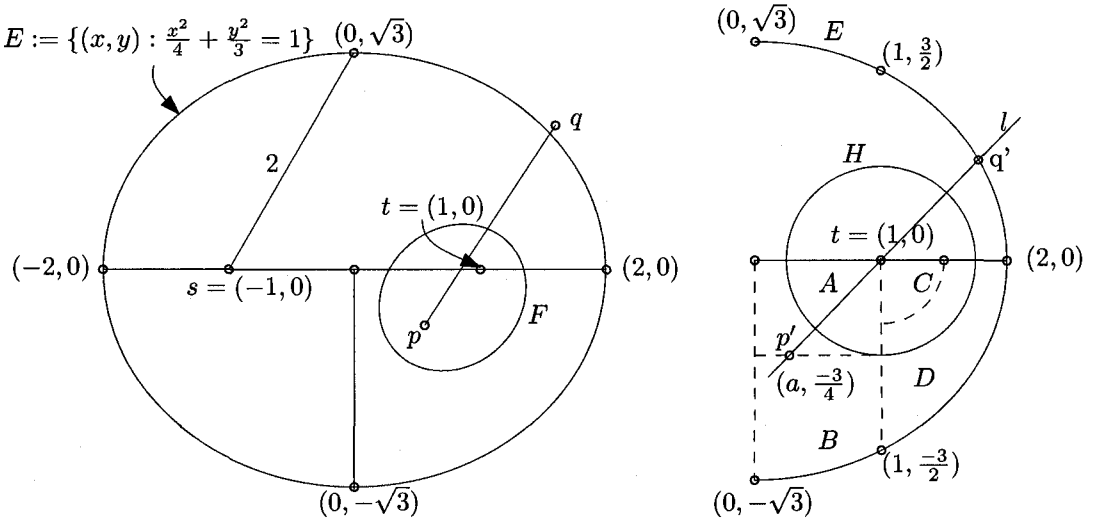


Figure 5.3: Proof of Lemma 5.11.

(see Figure 5.3, left). If both p and q are outside E, then $|pq| > |st|$ (since we assume that (p,q) and (s,t) are crossing). If both p and q are inside E, then it follows from step 5 of the algorithm that the edge (s,t) is not added to G. Therefore, exactly one point of $\{p,q\}$ is inside E.

Without loss of generality, p is inside E, q is outside E, the pair (p,t) has a smaller index than the pair (p,s) after the pairs have been sorted in line 2, and p is below the x-axis. We show below that the ellipse F whose boundary is the set of points f such that $|pf| + |ft| = 2|pt|$ is completely contained inside E. Thus, since E does not contain any edge (besides (s,t)), the same is true for F. It follows that the edge (p,t) is added to G, thus preventing the insertion of the edge (s,t) because the edge (p,t) is easily seen to have smaller index than (s,t).

It remains to prove that the ellipse F is contained in the ellipse E. The proof considers four cases, depending on the location of p (see Figure 5.3, right).

Case A: $[0 \leq p_x \leq 1$ **and** $-3/4 \leq p_y \leq 0]$ We show that for any point e on E, we have $|pe| + |et| > 2|pt|$. Note that we only need to check the case when e is below the x-axis and either $p_y = -3/4$ or $p_x = 0$. We consider these two cases separately.

If $p_y = -3/4$, let $a = p_x$. In this case, we have

$$|pe| + |et| - 2|pt| = \sqrt{(a-e_x)^2 + (3/4+e_y)^2} + \sqrt{(e_x-1)^2 + e_y^2} - 2\sqrt{(a-1)^2 + 9/16}.$$

Since $e_y = -\sqrt{(12-3e_x^2)}/2$, the above expression is completely determined by a and e_x. Elementary algebraic transformations (verified with Maple) show that it always evaluates to a positive value when e_x varies from -2 to 2 and a varies from 0 to 1.

If $p_x = 0$, let $b = p_y$. We have

$$|pe| + |et| - 2|pt| = \sqrt{e_x^2 + (b-e_y)^2} + \sqrt{(e_x-1)^2 + e_y^2} - 2\sqrt{1+b^2}.$$

As in the previous case, when e_x varies from -2 to 2 and b varies from 0 to -3/4, elementary algebraic manipulations (which we verified with Maple) show that the above expression evaluates to a positive number.

Case B: $[0 \leq p_x \leq 1$ **and** $p_y < -3/4]$ In this case, we show that $|pq| > |st|$. Let a be the x-coordinate of p, let p' be the point $(a,-3/4)$, and let q' be the intersection of the line l through t and p' with the ellipse E. Since $|pq| \geq |p'q'|$, it is sufficient to

show that $|p'q'| > |st| = 2$. The line l is given by the equation

$$y = \frac{3(x-1)}{4(1-a)}.$$

Since E is given by the equation $3x^2 + 4y^2 = 12$, the intersection between E and l is given by:

$$4(1-a)^2x^2 + 3(x-1)^2 - 16(1-a)^2 = 0.$$

For a fixed value of a, let $x(a)$ be the largest root of the above polynomial, and let $y(a)$ be the y-coordinate of l at $x = x(a)$. Then

$$|p'q'| = \sqrt{(x(a)-a)^2 + (y(a)+3/4)^2}.$$

When a varies from 0 to 1, this expression always evaluates to strictly more than 2.

Case C: $[p_x > 1$ **and** $|pt| \leq 1/2]$ In this case, the ellipse F is completely contained in the circle H centered at t whose radius is $3/2$. Since H is contained in E, F is also contained in E.

Case D: $[p_x > 1$ **and** $|pt| > 1/2]$ We separate this case into two subcases, depending on whether q has a positive or negative x-coordinate. If it is positive, then the part of $\overline{pq}$ that is above the x-axis has length at least $3/2$ and the part of $\overline{pq}$ that is below the x-axis has length more than $1/2$, which means that $|pq| > 2 = |st|$. If the x-coordinate of q is negative but greater than -1, then the same reasoning applies. If the x-coordinate of q is smaller than -1, then the projection of $\overline{pq}$ on the x-axis is larger than 2, which means that $|pq| > 2 = |st|$. □

Lemma 5.12 *The graph G computed by Algorithm 5.2 is 3-colorable.*

Proof: By Lemmas 5.10 and Lemma 5.11, G is plane and triangle-free. It is known that such a graph is 3-colorable; see [37],[80]. □

Proposition 5.13 *For any point set P, the 3-coloring of P computed by Algorithm 5.2 has spanning ratio at most 2. Thus, we have $t(3) \leq 2$.*

Proof: It is sufficient to show that the 3-coloring c produced by Algorithm 5.2 has the 2-ellipse property. Let p and q be points in P such that $c(p) = c(q)$. Let E be the ellipse whose boundary is the set of points e such that $|pe| + |eq| = 2|pq|$. Since (p, q) is not an edge in G, G must contain an edge (s, t) whose two endpoints are inside E. Since $c(s) \neq c(t)$, at least one of s and t has a different color than p and q. Without loss of generality, s is that point. Since s is inside E, we have that $|ps|+|sq| \leq 2|pq|$. □

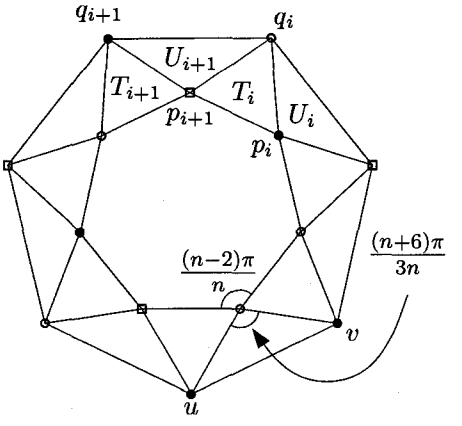


Figure 5.4: Lower bound of $2 - \epsilon$ for $k = 3$.

Proposition 5.14 *For every $\epsilon > 0$, there exists a point set P such that every 3-coloring of P has spanning ratio at least $2 - \epsilon$. Thus, we have $t(3) \geq 2$.*

Proof: Let n be an odd integer, and let $P = \{p_1, \ldots, p_n, q_1, \ldots, q_n\}$ where the p_i's are the vertices of a regular n-gon given in counter-clockwise order, and the q_i's are such that the triangles $T_i = (q_i, p_i, p_{i+1})$ are equilateral with interior lying outside the n-gon (indices are taken modulo n); see Figure 5.4. Now consider the set of triangles $\mathcal{T} = \{T_1, \ldots, T_n, U_1, \ldots, U_n\}$, where $U_i = (q_{i-1}, p_i, q_i)$. A simple parity argument shows that, for any 3-coloring of P, there is at least one triangle of $\mathcal{T}$ that has two vertices u and v that are assigned the same color. If this triangle is a T_i, then the spanning ratio between u and v is at least 2. If this triangle is a U_i, then the spanning ratio between u and v is at least $1/\sin((n + 6)\pi/6n)$, which tends to 2 as n goes to infinity. □

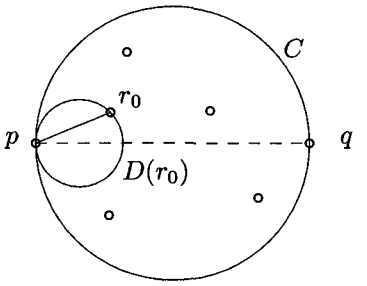


Figure 5.5: Upper bound of $\sqrt{2}$ for $k = 4$.

Next, we consider the case when $k = 4$. For this case, we simply use the Delaunay triangulation to find a 4-coloring. We then show that this coloring satisfies the $\sqrt{2}$-ellipse property.

Algorithm 5.3: Offline 4 Colors

Input: P, a set of points in the plane

Output: c, a 4-coloring of P

1 Compute the Delaunay triangulation D of P;

2 $c \leftarrow$ a 4-coloring of D;

Proposition 5.15 *For any point set P, the 4-coloring of P computed by Algorithm 5.3 has spanning ratio at most $\sqrt{2}$. Thus, we have $t(4) \leq \sqrt{2}$.*

Proof: It is sufficient to show that the coloring c computed by Algorithm 5.3 has the $\sqrt{2}$-ellipse property. Let p and q be points of P such that $c(p) = c(q)$. Since (p, q) is not an edge in the Delaunay triangulation, the circle C whose diameter is pq contains at least one point of P. For a point r inside C, let $D(r)$ be the circle through p and r whose center is on pq (see Figure 5.5). Let r_0 be the point inside C such that $D(r_0)$ has minimum diameter. Then, $D(r_0)$ is an empty circle with p and r_0 on its boundary, which means that (p, r_0) is a Delaunay edge. Therefore, $c(r_0) \neq c(p)$, and since r_0 is inside C, we have $|pr_0| + |r_0q| \leq \sqrt{2}|pq|$. □

Proposition 5.16 *For every $\epsilon > 0$, there exists a point set P such that every 4-coloring of P has spanning ratio at least $\sqrt{2} - \epsilon$. Thus, we have $t(4) \geq \sqrt{2}$.*

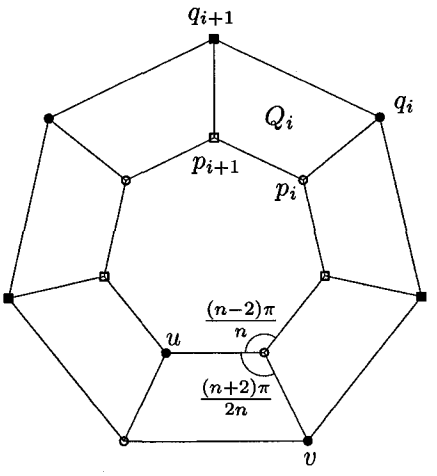


Figure 5.6: Lower bound of $\sqrt{2} - \epsilon$ for $k = 4$.

Proof: Let n be an odd integer, and let $P = \{p_1, \ldots, p_n, q_1, \ldots, q_n\}$, where the p_i's are the vertices of a regular n-gon, the q_i's are the vertices of a larger regular n-gon with the same center, and $|q_i p_i| = |p_i p_{i+1}|$ for all i; refer to Figure 5.6. Let Q_i be the quadrilateral $(p_i, p_{i+1}, q_{i+1}, q_i)$. A simple parity argument shows that for any 4-coloring of P, there is at least one Q_i that has two vertices u and v that are assigned the same color. The spanning ratio between u and v is then at least $1/\sin((n+2)\pi/4n)$, which tends to $\sqrt{2}$ when n goes to infinity. □

Our general algorithm for values $k > 4$ uses ideas from the ordered Θ-graph of Bose *et al.* [15]. We take advantage of the fact that we are in an off-line context, so that we can sort the points according to their y-coordinates. We process the points one by one from the lowest to the highest, splitting the half-plane below the current point p being processed into $k-1$ cones of angle $\pi/(k-1)$ and having their apex at p. For each such cone c_j, we take the point r_j in c_j that is closest to p. Then we assign p a color that has not been assigned to any of the r_j's. The fact that this algorithm uses at most k colors is straightforward, since there are at most $k-1$ such r_j.

Proposition 5.17 *For $k > 4$, we have $t(k) \leq 1 + 2\sin(\pi/(2k-2))$.*

Proof: Let p and q be points of P such that $c(p) = c(q)$. We may assume without loss of generality that $q_y \leq p_y$. Let c be the cone with apex at p that contains q in

Algorithm 5.4: Offline k Colors

Input: P, a set of points in the plane

Output: c, a k-coloring of P

1 Let $p_1, \ldots, p_n$ be the points of P sorted in non-decreasing order of y-coordinates;
2 **for** $i = 1$ *to* n **do**
3 | partition the half-plane below p_i into $k-1$ cones of angle $\theta = \pi/(k-1)$ and apex p_i;
4 | for each cone c_j, let r_j be the point in c_j that is closest to p_i;
5 | $c(p_i) \leftarrow \min\{l > 0 : \forall r_j, c(r_j) \neq l\}$;
6 **end**

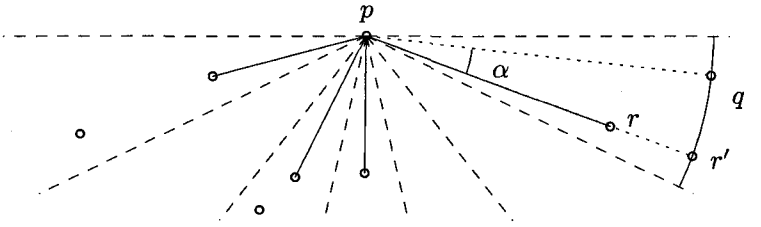


Figure 5.7: Upper bound of $1 + 2\sin(\pi/(2k-2))$ for $k > 4$.

line 4 of Algorithm 5.4, let r be the nearest neighbor of p in c, let r' be the intersection between the ray emanating from p through r and the circle centered at p with radius $|pq|$, and let $\alpha = \angle rpq$ (see Figure 5.7). Then,

$$|pr|+|rq| \leq |pr|+|rr'|+|r'q| = |pq|+|r'q| = |pq|+2\sin\frac{\alpha}{2}|pq| \leq (1+2\sin\frac{\pi}{2(k-1)})|pq|.$$

It follows that the coloring computed by Algorithm 5.4 has the $(1+2\sin(\pi/(2k-2))$-ellipse property. The result follows from the fact that $c(r) \neq c(p)$ and that Algorithm 5.4 uses at most k colors. □

Lemma 5.18 *Let p, q, r be three distinct vertices of a regular $(k+1)$-gon. Then the ratio $(|pr|+|rq|)/|pq|$ is at least $1/\cos(\frac{\pi}{k+1})$ and this value is achieved when p, r, and q are consecutive vertices.*

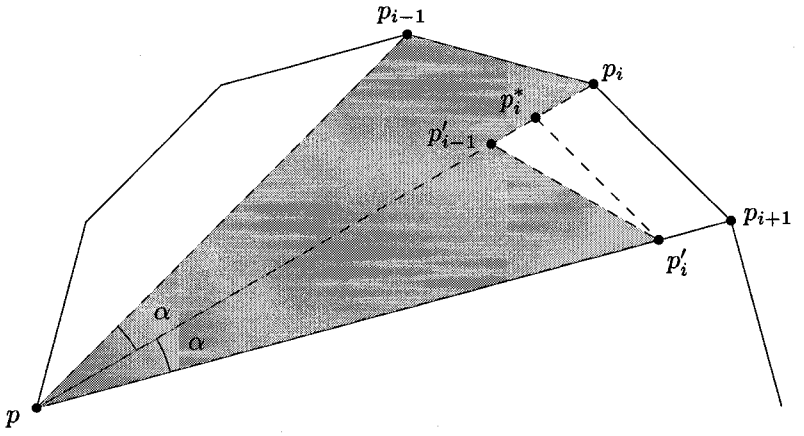


Figure 5.8: Illustration of the proof of Lemma 5.18.

Proof: For fixed p and q, the ratio $(|pr| + |rq|)/|pq|$ is minimized when r is adjacent to either p or q. In that case, the angle $\alpha = \angle qpr = \pi/(k+1)$. We show that for a fixed point p and three consecutive vertices p_{i-1}, p_i and p_{i+1} of the regular $(k+1)$-gon such that $|pp_{i-1}| < |pp_i| < |pp_{i+1}|$ (see Figure 5.8) the ratio $(|pp_{i-1}| + |p_{i-1}p_i|)/|pp_i|$ is smaller than $(|pp_i| + |p_ip_{i+1}|)/|pp_{i+1}|$ and the result follows.

Without loss of generality, p_{i-1}, p_i and p_{i+1} are in clockwise order. Let p'_{i-1} and p'_i be the rotation of p_{i-1} and p_i around p by a clockwise angle of α. Also, let p_i^* be the intersection of $\overline{pp_i}$ with the parallel line to $\overline{p_ip_{i+1}}$ through p'_i. Triangle $pp_i^*p'_i$ is similar to triangle pp_ip_{i+1}. Therefore,

$$\begin{aligned}
(|pp_i| + |p_ip_{i+1}|)/|pp_{i+1}| &= (|pp_i^*| + |p_i^*p'_i|)/|pp'_i| \\
&= (|pp'_{i-1}| + |p'_{i-1}p_i^*| + |p_i^*p'_i|)/|pp'_i| \\
&> (|pp'_{i-1}| + |p'_{i-1}p'_i|)/|pp'_i| \\
&= (|pp_{i-1}| + |p_{i-1}p_i|)/|pp_i|.
\end{aligned}$$

Therefore, the ratio $(|pp_{i-1}|+|p_{i-1}p_i|)/|pp_i|$ is minimized when p_{i-1} is adjacent to p. □

Proposition 5.19 *For $k > 4$, we have $t(k) \geq 1/\cos(\frac{\pi}{k+1})$.*

Proof: Let $P = \{p_1, \ldots, p_{k+1}\}$ be the vertex set of a regular $(k+1)$-gon. By Lemma 5.18, for any three distinct points p, q, and r in P, the ratio $(|pr| + |rq|)/|pq|$

is at least $1/\cos(\frac{\pi}{k+1})$ and this value is achieved when p, r, and q are consecutive vertices.

By the pigeonhole principle, any k-coloring of P has to assign the same color to at least two points, say p and q. By the argument above, the spanning ratio between p and q is at least $1/\cos(\frac{\pi}{k+1})$. □

The constructions we showed in this section use a quadratic number of edges since we consider the complete k-partite graph induced by the coloring of the points. To reduce this to a linear number of edges we apply Proposition 5.7, which slightly increases the spanning ratio, giving us the following:

Theorem 5.20 *The following are true:*

1. *For any point set P in the plane, the complete k-partite graph induced by the k-coloring of P computed by the above algorithms has a spanning ratio at most 3, 2, $\sqrt{2}$, and $1 + 2\sin\frac{\pi}{2(k+1)}$ for $k = 2$, $k = 3$, $k = 4$, $k > 4$, respectively.*

2. *For any $\epsilon > 0$, there exist point sets such that no coloring algorithm can compute a k-coloring that has the t-ellipse property for t smaller than $3-\epsilon$, $2-\epsilon$, $\sqrt{2}-\epsilon$, and $1/\cos\frac{\pi}{k+1}$ for $k = 2$, $k = 3$, $k = 4$, $k > 4$, respectively.*

3. *Thus, we have $t(2) = 3$, $t(3) = 2$, $t(4) = \sqrt{2}$, and $1 + 2\sin\frac{\pi}{2(k+1)} \geq t(k) \geq 1/\cos\frac{\pi}{k+1}$ for $k > 4$.*

4. *It is possible to obtain a $((1+\epsilon)t(k))$-spanner that has $O(|P|)$ edges, from the coloring computed by the above algorithms.*

5.4 Upper and lower bounds on $t'(k)$

Recall that in the on-line setting, the algorithm receives the points of P one at a time and assigns a color to a point as soon as it receives it. It cannot change the color of a point after this assignment. Naturally, this setting is more difficult which is reflected by higher bounds for $t'(3)$ and $t'(4)$. However, we are still able to give the exact value of $t'(k)$ for $k = 2, 3, 4$ and provide upper and lower bounds when $k > 4$. In the

online setting, we actually provide a general algorithm that is the same for all values of $k \geq 2$. Although it is similar to Algorithm 5.4, there are at least two important differences. First, since we are in an on-line setting, we cannot process the points in the order of their y-coordinates. Therefore, we have to use cones with an angle greater than $\pi/(k-1)$. If we choose the cones a priori as we do in Algorithm 5.4, we obtain cones whose angle is $2\pi/(k-1)$. However, by aligning the cones' bisectors on the points that are chosen to be neighbors, we can get a slightly better spanning ratio, since in this case, the angle is reduced to $2\pi/k$.

Algorithm 5.5: Online k Colors

Input: P, an arbitrarily ordered list of points in the plane

Output: c, a k-coloring of P

```
1  Let p_1, ..., p_n be the points of P in the given order;
2  for i = 1 to n do
3  |  P_i ← {p_1, ..., p_{i-1}};
4  |  j ← 0;
5  |  while P_i ≠ ∅ do
6  |  |  j ← j + 1;
7  |  |  r_j ← a nearest neighbor of p_i in P_i;
8  |  |  P_i ← P_i \ {r_j};
9  |  |  for each q ∈ P_i do
10 |  |  |  if ∠q p_i r_j ≤ 2π/k then
11 |  |  |  |  P_i ← P_i \ {q};
12 |  |  |  end
13 |  |  end
14 |  end
15 |  c(p_i) ← min{l > 0 | ∀r_j, c(r_j) ≠ l};
16 end
```

Proposition 5.21 *For $k \geq 2$, Algorithm 5.5 computes a k-coloring that has the t-ellipse property for $t = 1 + 2\sin(\pi/k)$. Thus, we have $t'(k) \leq 1 + 2\sin(\pi/k)$.*

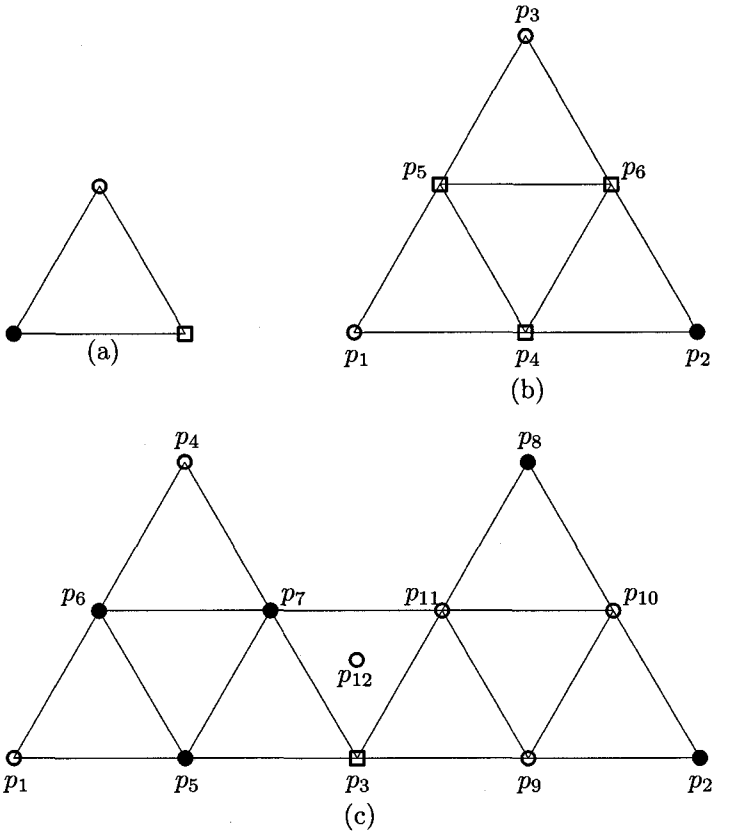


Figure 5.9: Online lower bound of $1+\sqrt{3}$ for $k=3$.

Proof: Algorithm 5.5 produces a k-coloring, because each p_i selects at most $k-1$ points r_j. If there were more than $k-1$ such points, then two of them would form an angle of $2\pi/k$ or less around p_i. However, this situation cannot occur because of lines 10 and 11. The proof on the spanning ratio is identical to the one given in Lemma 5.17. □

Corollary 5.22 *We have* $t'(2) \leq 3$, $t'(3) \leq 1+\sqrt{3}$ *and* $t'(4) \leq 1+\sqrt{2}$.

Since an off-line lower bound also provides an on-line lower bound, we have $t'(2) \geq t(2) = 3$. It follows that $t'(2) = 3$. We now prove that Algorithm 5.5 is also optimal for $k = 3$ and 4.

Proposition 5.23 *Let $\mathcal{A}$ be an arbitrary on-line coloring algorithm that guarantees a 3-coloring that has the t-ellipse property. Then its spanning ratio, t, is at least* $1+\sqrt{3}$.

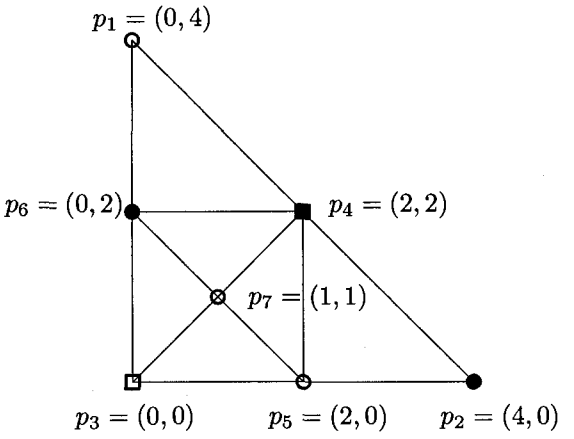


Figure 5.10: Online lower bound of $1+\sqrt{2}$ for $k = 4$.

Proof: The proof is by an adversarial argument, where the adversary forces a spanning ratio of at least $1+\sqrt{3}$. The main objective of the adversary is to force $\mathcal{A}$ to assign different colors to the vertices of an equilateral triangle. Then, the next point is placed in the center of this triangle (see Figure 5.9(a)). This results in a spanning ratio of $1+\sqrt{3}$.

Consider Figure 5.9(b), where the points are numbered by the order of insertion. Up to symmetry, there is only one way to assign colors to points p_1 to p_6 such that $t < 1+\sqrt{3}$ (e.g., $c(p_1) = red, c(p_2) = blue, c(p_3) = red, c(p_4) = green, c(p_5) = green, c(p_6) = green$). The key property is that the points p_4, p_5 and p_6 must be assigned the same color that is different from the colors assigned to the first three points. If any of these conditions is violated, then the spanning ratio is at least $1+\sqrt{3}$.

Next, consider Figure 5.9(c), where the point set of Figure 5.9(b) is reproduced twice. Consider triangles $\triangle(p_3, p_5, p_7)$, $\triangle(p_3, p_9, p_{11})$ and $\triangle(p_3, p_7, p_{11})$ after the insertion of p_{11}. At least one of these triangles has to be assigned three different colors, otherwise, the spanning ratio would already be $1+\sqrt{3}$. Assume w.l.o.g. that triangle $\triangle(p_3, p_7, p_{11})$ is assigned three different colors then by the insertion of point p_{12} in the center of the triangle, we force a spanning ratio of $1+\sqrt{3}$, as required. □

Proposition 5.24 *Let $\mathcal{A}$ be an arbitrary on-line coloring algorithm that guarantees a 4-coloring that has the t-ellipse property. Then the spanning ratio, t is at least* $1+\sqrt{2}$.

Proof: Consider the point set depicted in Figure 5.10. $\mathcal{A}$ must assign different colors to p_3, p_4, p_5 and p_6, otherwise the spanning ratio is already greater than $1+\sqrt{2}$. Upon introduction of p_7, $\mathcal{A}$ must assign it the same color as one of p_3, p_4, p_5 or p_6. The spanning ratio between p_7 and that point is $1+\sqrt{2}$. □

Proposition 5.25 *Let $\mathcal{A}$ be an arbitrary on-line coloring algorithm that guarantees a k-coloring that has the t-ellipse property. Then the spanning ratio, t, is at least $1/\cos(\frac{\pi}{k})$.*

Proof: Let $P = \{p_1, \ldots, p_k, q\}$, where the p_i' are the vertices of a regular k-gon K and q is the center of the circumcircle of K. If, after processing p_1 to p_k, $\mathcal{A}$ assigned the same color to two points, then as in Lemma 5.19, the spanning ratio is at least $1/\cos(\frac{\pi}{k})$. Otherwise, all p_i are assigned different colors. When q is introduced, the color $\mathcal{A}$ assigns to it has already been assigned to some other point p. In that case, the spanning ratio for the edge (q,p) is $1+4\sin(\pi/2k) > 1/\cos(\frac{\pi}{k})$. □

The constructions we showed in this section use a quadratic number of edges since we consider the complete k-partite graph induced by the coloring of the points. To reduce this to a linear number of edges we apply Proposition 5.7, which slightly increases the spanning ratio, giving us the following:

Theorem 5.26 *The following are true:*

1. *For any sequence P of points in the plane, the complete k-partite graph induced by the on-line k-coloring of P computed by the above algorithms has a spanning ratio at most 3, $1+\sqrt{3}$, $1+\sqrt{2}$, and $1+2\sin\frac{\pi}{k}$ for $k=2$, $k=3$, $k=4$, $k>4$, respectively.*

2. *For any $\epsilon > 0$, there exist point sets such that no on-line coloring algorithm can compute an on-line k-coloring that has the t-ellipse property for t smaller than $3-\epsilon$, $1+\sqrt{3}-\epsilon$, $1+\sqrt{2}-\epsilon$, and $1/\cos\frac{\pi}{k}$ for $k=2$, $k=3$, $k=4$, $k>4$, respectively.*

3. *Thus, we have* $t'(2) = 3$, $t'(3) = 1 + \sqrt{3}$, $t'(4) = 1 + \sqrt{2}$, *and* $1 + 2\sin\frac{\pi}{k} \geq t'(k) \geq 1/\cos\frac{\pi}{k}$ *for* $k > 4$.

4. *It is possible to obtain a* $((1+\epsilon)t'(k))$*-spanner that has* $O(|P|)$ *edges, from the coloring computed by the above algorithms.*

5.5 Simulation Results

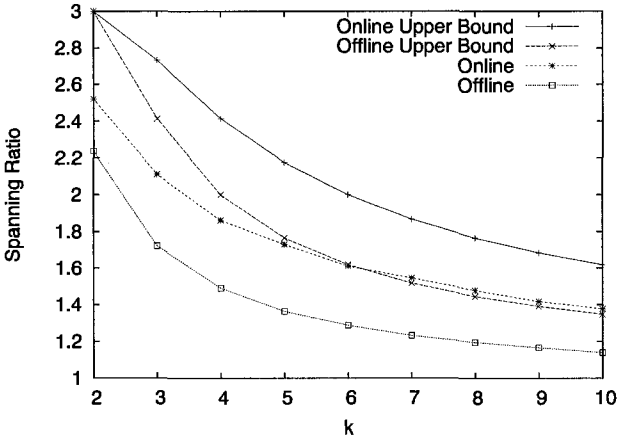
Using simulation, we now provide estimates of the average spanning ratio of the colorings produced by Algorithm 5.4 and Algorithm 5.5. Using a uniform distribution, we generated 200 sets of 50 points and 200 sets of 200 points. For each point set, we computed the spanning ratio for k ranging from 2 to 10. Figure 5.11 and Figure 5.12 show the results we obtained for the spanning ratio. The 95% confidence interval for these values is ± 0.0365.

The general behavior of the average case performance ratio of these algorithms is not much different than what can be expected from the worst case analysis. In particular, the off-line algorithm performs significantly better than the on-line algorithm. Also, in both cases, as k increases, the spanning ratio reduction becomes less and less important. Another interesting observation is that for k large enough ($k > 6$ for 50 points and $k > 3$ for 200 points), the average case spanning ratio of the on-line algorithm is worse than the worst case spanning ratio of the off-line algorithm.

For $k = 2, 3$ and 4, in the off-line case, we used the algorithm for general values of k. It is interesting to notice that for $k = 4$, the average spanning ratio obtained using Algorithm 5.4 is greater than the worst case spanning ratio obtained using Algorithm 5.3. However, Algorithm 5.3 is less practical, since we have to compute a 4-coloring of a planar graph.

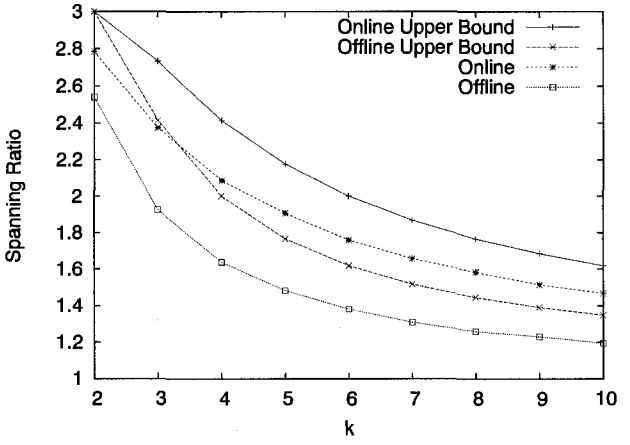
5.6 Conclusion

In this chapter, we investigated the problem of computing a spanner of a point set that has chromatic number k. To the best of our knowledge, this problem has not been considered before. For small values of k ($k \leq 4$), we provided tight upper and lower bounds on the smallest possible spanning ratio of such spanners. For larger values of



k	offline	online	k	offline	online
2	2.2383	2.5208	7	1.2329	1.5456
3	1.7219	2.1111	8	1.1947	1.4778
4	1.4907	1.8608	9	1.1658	1.4175
5	1.3631	1.7300	10	1.1384	1.3765
6	1.2877	1.6098			

Figure 5.11: Simulation results for 50 nodes using Algorithm 5.4 and Algorithm 5.5.



k	offline	online	k	offline	online
2	2.5390	2.7844	7	1.3079	1.6563
3	1.9245	2.3743	8	1.2579	1.5833
4	1.6377	2.0866	9	1.2283	1.5149
5	1.4831	1.9062	10	1.1945	1.4677
6	1.3809	1.7579			

Figure 5.12: Simulation results for 200 nodes using Algorithm 5.4 and Algorithm 5.5.

k, we provided general upper and lower bounds which, unfortunately, are not tight. Our construction algorithms show how to color a point set with k colors such that the complete k-partite graph induced by this coloring has the stated spanning ratio. The number of edges in these graphs can be reduced from quadratic to linear with a slight increase in the spanning ratio by applying the general technique of Gudmundsson *et al.* [39]. An interesting open problem in this setting of the problem is to find tight upper and lower bounds when $k > 4$.

We also considered an on-line variant of this problem where the points are presented sequentially and our algorithm assigns a color to each point upon reception such that the complete k-partite graph induced by the coloring is a constant spanner. Again, for small values of k ($k \leq 4$), we provided tight upper and lower bounds on the smallest possible spanning ratio of such spanners and for $k > 4$, we provided general upper and lower bounds that are not tight. A linear-sized spanner can be constructed after all the points have been colored by applying the technique of Gudmundsson *et al.* [39]. However, in this case, our algorithm for computing the linear-sized constant spanner is *not* on-line. Therefore, there are two open problems in the on-line setting. First, to close the gap between the upper and lower bound for $k > 4$. Second, provide an on-line algorithm that computes the linear-sized constant spanner.

Chapter 6

Spanners of Complete k-Partite Graphs

6.1 Introduction

In the previous chapter, we considered the problem of constructing spanners of point sets that have $O(n)$ edges and chromatic number at most k. We computed a spanner of the complete graph and our algorithms had the freedom to choose a "good" k-partition of the vertices. In this chapter, we consider a variant of this problem where the k-partition is given and we want to compute a spanner of the complete *k-partite graph*. Notice that in this chapter, the definition of spanner that we use is more general since we want to compute a spanner of a complete k-partite graph as opposed to a complete graph.

Let S be a set of n points in $\mathbb{R}^d$. A *geometric graph* with vertex set S is an undirected graph H whose edges are line segments $\overline{pq}$ that are weighted by the Euclidean distance $|pq|$ between p and q. Notice that H is not necessarily a complete graph. For any two points p and q in S, we denote by $d_H(p,q)$ the length of a shortest path in H between p and q. For a real number $t \geq 1$, a subgraph G of H is said to be a *t-spanner* of H, if $d_G(p,q) \leq t \cdot d_H(p,q)$ for all points p and q in S. The smallest t for which this property holds is called the *spanning ratio* of G. Thus, a subgraph G of H with spanning ratio t approximates the $\binom{n}{2}$ pairwise shortest-path lengths in H within a factor of t. If H is the complete geometric graph with vertex set S, then G is also called a t-spanner of the point set S.

In the literature, most of the work on constructing geometric spanners is for the case where H is the complete graph. It is well known that for any set S of n points in $\mathbb{R}^d$ and for any real constant $\epsilon > 0$, there exists a $(1+\epsilon)$-spanner of S containing $O(n)$ edges. Moreover, such spanners can be computed in $O(n \log n)$ time; see Salowe [76] and Vaidya [83]. For a detailed overview of results on spanners for point sets, see the book by Narasimhan and Smid [65].

For spanners of arbitrary geometric graphs, much less is known. Althöfer *et al.* [4] showed that for any $t > 1$, every weighted graph H with n vertices contains a subgraph with $O(n^{1+2/(t-1)})$ edges, that is a t-spanner of H. Observe that this result holds for any weighted graph; in particular, it is valid for any geometric graph. For geometric graphs, a lower bound was given by Gudmundsson and Smid [40]: They proved that for every real number t with $1 < t < \frac{1}{4}\log n$, there exists a geometric graph H with n vertices, such that every t-spanner of H contains $\Omega(n^{1+1/t})$ edges. Thus, if we are looking for spanners with $O(n)$ edges of arbitrary geometric graphs, then the best spanning ratio we can obtain is $\Theta(\log n)$.

In this chapter, we consider the case when the input graph is a complete k-partite geometric graph. Let S be a set of n points in $\mathbb{R}^d$, and let S be partitioned into subsets $C_1, C_2, \ldots, C_k$. Let $K_{C_1 \ldots C_k}$ denote the *complete k-partite graph on S*. This graph has S as its vertex set and two points p and q are connected by an edge (of length $|pq|$) if and only if p and q are in different subsets of the partition. The problem we address is defined as follows:

Problem 6.1 *Let $k \geq 2$ be an integer, let S be a set of n points in $\mathbb{R}^d$, and let S be partitioned into k subsets $C_1, C_2, \ldots, C_k$. Compute a t-spanner of the complete k-partite graph $K_{C_1 \ldots C_k}$ that has a "small" number of edges and whose spanning ratio t is "small".*

The main contribution of this chapter is to present an algorithm that computes such a t-spanner with $O(n)$ edges in $O(n \log n)$ time, where $t = 5 + \epsilon$ for any constant $\epsilon > 0$. We also show that if one is willing to use $O(n \log n)$ edges, then our algorithm adapts easily to reach a spanning ratio of $t = 3 + \epsilon$. Finally, we show that the latter result is optimal: For any k with $2 \leq k \leq n - \Theta(\sqrt{n \log n})$, spanners with $O(n \log n)$ edges and spanning ratio less than 3 do not exist for all complete k-partite geometric graphs.

The rest of this chapter is organized as follows. In Section 6.2, we review the properties of the Well-Separated Pair Decomposition (WSPD) that we use in our algorithm. In Section 6.3, we provide an algorithm that solves the problem of constructing a spanner of the complete k-partite graph. In Section 6.4, we show that the

spanner constructed by this algorithm has $O(n)$ edges and that its spanning ratio is bounded from above by a constant that depends only on the dimension d. In Section 6.5, we show how a simple modification to our algorithm improves the spanning ratio to $5+\epsilon$ while still having $O(n)$ edges. In Section 6.6, we show how to achieve a spanning ratio of $3+\epsilon$ using $O(n \log n)$ edges. We also prove that the latter result is optimal. We conclude in Section 6.7.

6.2 The Well-Separated Pair Decomposition

In this section, we review the crucial properties of the Well-Separated Pair Decomposition (WSPD) of Callahan and Kosaraju [18] that we use for our construction. The reader familiar with the WSPD may go directly to Section 6.3. Our presentation follows the one in Narasimhan and Smid [65]. Intuitively, a WSPD is a partition of the edges of a complete geometric graph such that all edges that are grouped together are *approximately* equal. To give a formal definition of the WSPD, we first need to define what it means for two sets to be well-separated.

Definition 6.2 *Let S be a set of points in $\mathbb{R}^d$. The* bounding box $\beta(S)$ *of S is the smallest axes-parallel hyperrectangle that contains S.*

Definition 6.3 *Let X and Y be two sets of points in $\mathbb{R}^d$ and let $s > 0$ be a real number. We say that X and Y are* well-separated *with respect to s if there exist two balls B_1 and B_2 such that*

1. *B_1 and B_2 have the same radius, say ρ,*
2. *B_1 contains the bounding box of X,*
3. *B_2 contains the bounding box of Y, and*
4. *the distance $\min\{|xy| : x \in B_1, y \in B_2\}$ between B_1 and B_2 is at least $s\rho$.*

Definition 6.4 *Let S be a set of points in $\mathbb{R}^d$ and let $s > 0$ be a real number. A* well-separated pair decomposition (WSPD) of S with separation constant s *is a set of unordered pairs of subsets of S that*

1. *are well-separated with respect to s, and*

2. *for any two distinct points $p, q \in S$ there is a unique pair $\{X, Y\}$ in the WSPD with $p \in X$ and $q \in Y$.*

Lemma 6.5 (Lemma 9.1.2 in [65]) *Let $s > 0$ be a real number and let X and Y be two point sets that are well-separated with respect to s.*

1. *If $p, p', p'' \in X$ and $q \in Y$, then $|p'p''| \leq (2/s)|pq|$.*

2. *If $p, p' \in X$ and $q, q' \in Y$, then $|p'q'| \leq (1 + 4/s)|pq|$.*

The first part of this lemma states that distances within one set are "small" compared to distances between pairs of points having one endpoint in each set. The second part states that all pairs of points having one endpoint in each set have approximately the "same" distance.

Callahan and Kosaraju [17] showed how to construct a t-spanner of S from a WSPD: All one has to do is pick from each pair $\{X, Y\}$ an arbitrary edge (p, q) with $p \in X$ and $q \in Y$. Using induction on the rank of the length of the edges in the complete graph K_S, it can be shown that, when $s > 4$, this process leads to a $((s+4)/(s-4))$-spanner. Thus, by choosing s to be a sufficiently large constant, the spanning ratio can be made arbitrarily close to 1.

In order to compute a spanner of S that has a linear number of edges, one needs a WSPD that has a linear number of pairs. Callahan and Kosaraju [18] showed that a WSPD with a linear number of pairs always exists and can be computed in time $O(n \log n)$. Their algorithm uses a split-tree, which we define below.

Definition 6.6 *Let S be a non-empty set of points in $\mathbb{R}^d$. The* split-tree *of S is defined as follows: if S contains only one point, then the split-tree is a single node that stores that point. Otherwise, the split-tree has a root that stores the bounding box $\beta(S)$ of S, as well as an arbitrary point of S called the* representative *of S denoted by rep(S). Split $\beta(S)$ into two hyperrectangles by cutting its longest interval into two equal parts, and let S_1 and S_2 be the subsets of S contained in the two hyperrectangles. The root of the split-tree of S has two sub-trees, which are recursively defined split-trees of S_1 and S_2.*

The precise way Callahan and Kosaraju used the split-tree to compute a WSPD with a linear number of pairs is of no importance to us. The only important aspect we need to retain is that each pair is uniquely determined by a pair of nodes in the tree. More precisely, for each well-separated pair $\{X, Y\}$ in the WSPD that is output by their algorithm, there are unique internal nodes u and v in the split-tree such that the sets S_u and S_v of points stored at the leaves of the subtrees rooted at u and v are precisely the sets X and Y. Since there is such a unique correspondence, we denote pairs in the WSPD by $\{S_u, S_v\}$, meaning that u and v are the nodes corresponding to the sets $X = S_u$ and $Y = S_v$. Also, although the WSPD of a point set is not unique, when we talk about *the* WSPD, we mean the WSPD that is computed by the algorithm of Callahan and Kosaraju.

Before we present our algorithm, we state the following lemmas that we use to analyze our algorithm in Section 6.4. If R is an axes-parallel hyperrectangle in $\mathbb{R}^d$, then we use $L_{\max}(R)$ to denote the length of a longest side of R.

Lemma 6.7 (Lemma 9.5.3 in [65]) *Let u be a node in the split-tree and let u' be a node in the subtree of u such that the path between them contains at least d edges. Then*

$$L_{\max}(\beta(S_{u'})) \leq \frac{1}{2} \cdot L_{\max}(\beta(S_u)).$$

Lemma 6.8 (Lemma 11.3.1 in [65]) *Let $\{S_u, S_v\}$ be a pair in the WSPD, let ℓ be the distance between the centers of $\beta(S_u)$ and $\beta(S_v)$, and let $\pi(u)$ be the parent of u in the split-tree. Then*

$$L_{\max}(\beta(S_{\pi(u)})) \geq \frac{2\ell}{\sqrt{d}(s+4)}.$$

6.3 A First Algorithm

We now show how the WSPD can be used to address the problem of computing a spanner of a complete k-partite graph. In this section, we introduce an algorithm that outputs a graph with constant spanning ratio and $O(n)$ edges. The analysis of this algorithm is presented in Section 6.4. In Section 6.5, we show how this algorithm can be improved to achieve a spanning ratio of $5 + \epsilon$.

The input set $S \subseteq \mathbb{R}^d$ is the disjoint union of k sets $C_1, C_2, \ldots, C_k$. We say that the elements of C_c have "color" c. The graph $K = K_{C_1 \ldots C_k}$ is the complete k-partite geometric graph.

Definition 6.9 *Let T be the split-tree of S that is used to compute the WSPD of S.*

1. *For any node u in T, we denote by S_u the set of all points in the subtree rooted at u.*

2. *We define MWSPD to be the subset of the WSPD obtained by removing all pairs $\{S_u, S_v\}$ for which all points of $S_u \cup S_v$ have the same color.*

3. *A node u in T is called* multichromatic *if there exist points p and q in S_u and a node v in T, such that p and q have different colors and $\{S_u, S_v\}$ is in the MWSPD.*

4. *A node u in T is called a c-*node *if all points of S_u have color c and there exists a node v in T such that $\{S_u, S_v\}$ is in the MWSPD.*

5. *A c-node u in T is called a c-*root *if it does not have a proper ancestor that is a c-node in T.*

6. *A c-node u in T is called a c-*leaf *if it has no descendent c-node.*

7. *A c-node u' in T is called a c-*child *of a c-node u in T if u' is in the subtree rooted at u and there is no c-node on the path between u and u'.*

8. *A c-node u in T is called a c-*parent *of a c-node u' in T if u' is a c-child of u.*

9. *For each color c and for each c-node u in T, $rep(S_u)$ denotes a fixed arbitrary point in S_u.*

10. *For each multichromatic node u in T, $rep(S_u)$ and $rep'(S_u)$ denote two fixed arbitrary points in S_u that have different colors.*

11. *The* distance *between two sets S_v and S_w, denoted by $dist(S_v, S_w)$, is defined to be the distance between the centers of their bounding boxes.*

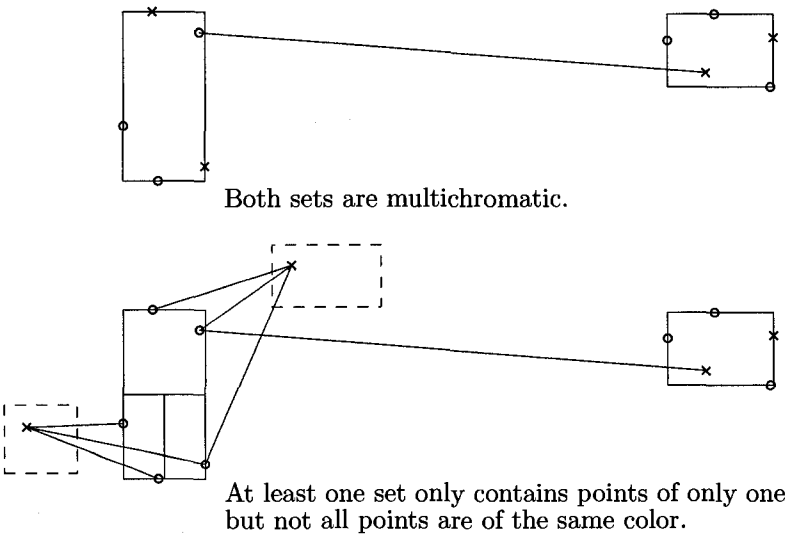


Figure 6.1: The two cases of Algorithm 6.1.

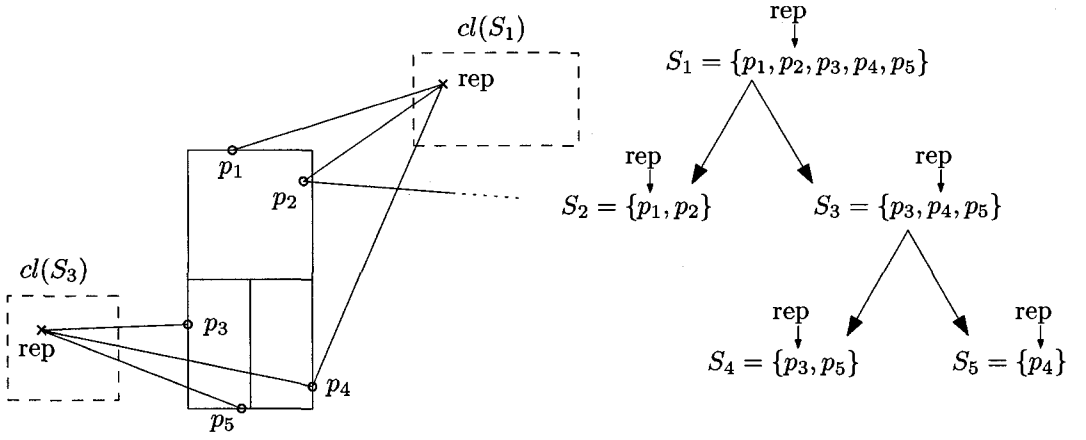


Figure 6.2: Handling a c-node.

12. *Let u be a c-node in T. Consider all pairs $\{S_v, S_w\}$ in the MWSPD, where v is a c-node on the path in T from u to the root (this path includes u). Let $\{S_v, S_w\}$ be such a pair for which $dist(S_v, S_w)$ is minimum. We define $cl(S_u)$ to be the set S_w.*

Algorithm 6.1 computes a spanner of a complete k-partite geometric graph $K = K_{C_1 \ldots C_k}$. The intuition behind this algorithm is the following. First, the algorithm computes the WSPD. Then, it considers each pair $\{S_u, S_v\}$ of the MWSPD, and decides whether or not it adds an edge between S_u and S_v. The outcome of this decision is based on the following two cases, illustrated in Figures 6.1 and 6.2.

Algorithm 6.1: Computing a sparse subgraph of $K_{C_1 \ldots C_k}$ whose spanning ratio is bounded by a constant.

Input: A set S of points in $\mathbb{R}^d$, which is partitioned into k subsets $C_1, \ldots, C_k$.
Output: A spanner $G = (S, E)$ of the complete k-partite graph $K_{C_1 \ldots C_k}$.

```
compute the split-tree T of S;
using T, compute the WSPD with respect to a separation constant s > 0;
using the WSPD, compute the MWSPD;
E ← ∅;
for each color c in {1, 2, ..., k} do
    for each c-root u in T do
        for each c-leaf u' in the subtree of u do
            for each p ∈ S_u' do
                if rep(cl(S_u')) does not have color c
                then add (p, rep(cl(S_u'))) to E;
                else add (p, rep'(cl(S_u'))) to E;
            end
        end
        for each c-node u' that is in the subtree of u (including u) do
            if rep(cl(S_u')) does not have color c
            then add (rep(S_u'), rep(cl(S_u'))) to E;
            else add (rep(S_u'), rep'(cl(S_u'))) to E;
            for each pair {S_u', S_v'} in the MWSPD do
                if rep(S_v') does not have color c
                then add (rep(S_u'), rep(S_v')) to E;
                else add (rep(S_u'), rep'(S_v')) to E;
            end
            for each c-child u'' of u' do
                if rep(cl(S_u')) does not have color c
                then add (rep(S_u''), rep(cl(S_u'))) to E;
                else add (rep(S_u''), rep'(cl(S_u'))) to E;
            end
        end
    end
end
for each {S_u, S_v} in the MWSPD for which both u and v are multichromatic do
    if rep(S_u) and rep(S_v) have distinct colors
    then add (rep(S_u), rep(S_v)) to E;
    else add (rep(S_u), rep'(S_v)) to E;
end
return the graph G = (S, E)
```

Case 1: Both S_u and S_v are multichromatic. In this case, the algorithm adds one edge between S_u and S_v to the spanner; see lines 33–34. Observe that the two vertices of this edge do not have the same color. This edge allows to approximate each edge (p, q) of K, where $p \in S_u$, $q \in S_v$, and p and q have different colors.

Case 2: All points in S_u are of the same color c (i.e., u is a c-node). In this case, an edge is added between $\text{rep}(S_u)$ and a representative of S_v whose color is not c; see lines 20–21. In order to approximate each edge of K having one vertex (of color c) in S_u and the other vertex (of a different color) in S_v, more edges have to be added. This is done in such a way that our final graph contains a "short" path between every point p of S_u and the representative $\text{rep}(S_u)$ of S_u. Observe that this path must contain points whose color is not equal to c; thus, these points are not in S_u. One way to achieve this is to add an edge between each point of S_u and a representative of $\text{cl}(S_u)$ whose color is not c; we call this construction a *star*. However, since the subtree rooted at u may contain other c-nodes, many edges may be added for each point in S_u, which could possibly lead to a quadratic number of edges in the final graph. To guarantee that the algorithm does not add too many edges, it introduces a star only if u is a c-leaf; see lines 8–12. If u is not a c-leaf (i.e., u is an internal c-node), the algorithm only adds one edge between $\text{rep}(S_u)$ and a representative of $\text{cl}(S_u)$ whose color is not c; see lines 16–17. Then, the algorithm links each c-node u'' that is not a c-root to its c-parent u'. This is done through an edge between $\text{rep}(S_{u''})$ and a representative of $\text{cl}(S_{u'})$ whose color is not c; see lines 25–26. This second case is illustrated in Figure 6.2.

6.4 Analysis of Algorithm 6.1

Lemma 6.10 *The graph G computed by Algorithm 6.1 has $O(|S|)$ edges.*

Proof: For each color c and for each c-leaf u', the algorithm adds $|S_{u'}|$ edges to G in lines 10–11. Since the sets $S_{u'}$, where u' ranges over all c-leaves and c ranges over all colors, are pairwise disjoint, the total number of edges that are added in lines 10–11 is $O(|S|)$.

The total number of edges that are added in lines 20–21 and 33–34 is at most the

number of pairs in the MWSPD. Since the WSPD contains $O(|S|)$ pairs (see [18]), the same upper bound holds for the number of edges added in lines 20–21 and 33–34.

The total number of edges that are added in lines 16–17 and 25–26 is at most twice the number of nodes in the split-tree, which is $O(|S|)$. □

Lemma 6.11 *Let G be the graph computed by Algorithm 6.1. Let p and q be two points of S with different colors, and let $\{S_u, S_v\}$ be the pair in the MWSPD for which $p \in S_u$ and $q \in S_v$. Assume that u is a c-node for some color c. Then there is a path in G between p and $rep(S_u)$ whose length is at most $t'|pq|$, where*

$$t' = 4\sqrt{d}(\mu d + 1)(1 + 4/s)^3,$$

$$\mu = \left\lceil \log\left(\sqrt{d}(1 + 4/s)\right)\right\rceil + 1,$$

and s is the separation constant of the WSPD.

Proof: Let w be the c-leaf such that $p \in S_w$, and let $w = w_0, \ldots, w_k = u$ be the sequence of c-nodes that are on the path in T from w to u.

Recall from Definition 6.9 that each set S_{w_i}, $0 \leq i \leq k$, has a representative $\text{rep}(S_{w_i})$ (of color c) associated with it. Also, recall the definition of the sets $\text{cl}(S_{w_i})$, $0 \leq i \leq k$; see Definition 6.9. If $\text{cl}(S_{w_i})$ is a c'-node for some color c', then this set has one representative $\text{rep}(\text{cl}(S_{w_i}))$ associated with it. Otherwise, $\text{cl}(S_{w_i})$ is multichromatic and this set has two representatives $\text{rep}(\text{cl}(S_{w_i}))$ and $\text{rep}'(\text{cl}(S_{w_i}))$ of different colors associated with it. We may assume without loss of generality that, for all $0 \leq i \leq k$, the color of $\text{rep}(\text{cl}(S_{w_i}))$ is not equal to c.

Let Π be the path

$$\begin{array}{rcccl} p & \rightarrow & \text{rep}(\text{cl}(S_{w_0})) & \rightarrow & \text{rep}(S_{w_0}) \\ & \rightarrow & \text{rep}(\text{cl}(S_{w_1})) & \rightarrow & \text{rep}(S_{w_1}) \\ & \vdots & & \vdots & \\ & \rightarrow & \text{rep}(\text{cl}(S_{w_k})) & \rightarrow & \text{rep}(S_{w_k}) \quad = \quad \text{rep}(S_u). \end{array}$$

The first edge on this path, i.e., $(p, \text{rep}(\text{cl}(S_{w_0})))$, is added to the graph G in lines 10–11 of the algorithm. The edges $(\text{rep}(\text{cl}(S_{w_i})), \text{rep}(S_{w_i}))$, $0 \leq i \leq k$, are added to G

in lines 16–17. Finally, the edges $(\text{rep}(S_{w_{i-1}}), \text{rep}(\text{cl}(S_{w_i})))$, $1 \leq i \leq k$, are added to G in lines 25–26. It follows that Π is a path in G between p and $\text{rep}(S_u)$. We show that the length of Π is at most $t'|pq|$.

Let i be an integer with $0 \leq i \leq k$. Recall the definition of $\text{cl}(S_{w_i})$; see Definition 6.9: We consider all pairs $\{S_x, S_y\}$ in the MWSPD, where x is a c-node on the path in T from w_i to the root, and pick the pair for which $\text{dist}(S_x, S_y)$ is minimum. We denote the pair picked by (S_{x_i}, S_{y_i}). Thus, x_i is a c-node on the path in T from w_i to the root, $\{S_{x_i}, S_{y_i}\}$ is a pair in the MWSPD, and $\text{cl}(S_{w_i}) = S_{y_i}$. We define

$$\ell_i = \text{dist}(S_{x_i}, S_{y_i}).$$

Consider the first edge $(p, \text{rep}(\text{cl}(S_{w_0})))$ on the path Π. Since $p \in S_{w_0} \subseteq S_{x_0}$ and $\text{rep}(\text{cl}(S_{w_0})) \in S_{y_0}$, it follows from Lemma 6.5 that

$$|p, \text{rep}(\text{cl}(S_{w_0}))| \leq (1 + 4/s) \cdot \text{dist}(S_{x_0}, S_{y_0}) = (1 + 4/s)\ell_0.$$

Let $0 \leq i \leq k$ and consider the edge $(\text{rep}(\text{cl}(S_{w_i})), \text{rep}(S_{w_i}))$ on Π. Since $\text{rep}(S_{w_i}) \in S_{w_i} \subseteq S_{x_i}$ and $\text{rep}(\text{cl}(S_{w_i})) \in S_{y_i}$, it follows from Lemma 6.5 that

$$|\text{rep}(\text{cl}(S_{w_i})), \text{rep}(S_{w_i})| \leq (1 + 4/s) \cdot \text{dist}(S_{x_i}, S_{y_i}) = (1 + 4/s)\ell_i. \tag{6.1}$$

Let $1 \leq i \leq k$ and consider the edge $(\text{rep}(S_{w_{i-1}}), \text{rep}(\text{cl}(S_{w_i})))$ on Π. Since $\text{rep}(S_{w_{i-1}}) \in S_{w_{i-1}} \subseteq S_{x_i}$ and $\text{rep}(\text{cl}(S_{w_i})) \in S_{y_i}$, it follows from Lemma 6.5 that

$$|\text{rep}(S_{w_{i-1}}), \text{rep}(\text{cl}(S_{w_i}))| \leq (1 + 4/s) \cdot \text{dist}(S_{x_i}, S_{y_i}) = (1 + 4/s)\ell_i.$$

Thus, the length of the path Π is at most

$$\sum_{i=0}^{k} 2(1 + 4/s)\ell_i.$$

Therefore, it is sufficient to prove that

$$\sum_{i=0}^{k} \ell_i \leq 2\sqrt{d}(\mu d + 1)(1 + 4/s)^2|pq|.$$

It follows from the definition of $\text{cl}(S_u) = \text{cl}(S_{w_k})$ that

$$\ell_k = \text{dist}(S_{x_k}, S_{y_k}) \leq \text{dist}(S_u, S_v).$$

Since, by Lemma 6.5, $\mathrm{dist}(S_u, S_v) \leq (1 + 4/s)|pq|$, it follows that

$$\ell_k \leq (1 + 4/s)|pq|. \tag{6.2}$$

Thus, it is sufficient to prove that

$$\sum_{i=0}^{k} \ell_i \leq 2\sqrt{d}(\mu d + 1)(1 + 4/s)\ell_k. \tag{6.3}$$

If $k = 0$, then (6.3) obviously holds. Assume from now on that $k \geq 1$. For each i with $0 \leq i \leq k$, we define

$$a_i = L_{\max}(\beta(S_{w_i})),$$

i.e., a_i is the length of a longest side of the bounding box of S_{w_i}.

Let $0 \leq i \leq k$. It follows from Lemma 6.5 that

$$L_{\max}(\beta(S_{x_i})) \leq \frac{2}{s}\ell_i.$$

Since w_i is in the subtree of x_i, we have $L_{\max}(\beta(S_{w_i})) \leq L_{\max}(\beta(S_{x_i}))$. Thus, we have

$$a_i \leq \frac{2}{s}\ell_i \text{ for } 0 \leq i \leq k. \tag{6.4}$$

Lemma 6.7 states that

$$a_i \leq \frac{1}{2}a_{i+d} \text{ for } 0 \leq i \leq k - d. \tag{6.5}$$

Let $0 \leq i \leq k - 1$. Since w_i is a c-node, there is a node w_i' such that $\{S_{w_i}, S_{w_i'}\}$ is a pair in the MWSPD. Then it follows from the definition of $\mathrm{cl}(S_{w_i})$ that

$$\ell_i = \mathrm{dist}(S_{x_i}, S_{y_i}) \leq \mathrm{dist}(S_{w_i}, S_{w_i'}).$$

By applying Lemma 6.8, we obtain

$$\begin{aligned} \mathrm{dist}(S_{w_i}, S_{w_i'}) &\leq \frac{\sqrt{d}(s+4)}{2} L_{\max}(\beta(S_{\pi(w_i)})) \\ &\leq \frac{\sqrt{d}(s+4)}{2} L_{\max}(\beta(S_{w_{i+1}})) \\ &= \frac{\sqrt{d}(s+4)}{2} a_{i+1}. \end{aligned}$$

Thus, we have

$$\ell_i \leq \frac{\sqrt{d}(s+4)}{2} a_{i+1} \text{ for } 0 \leq i \leq k - 1. \tag{6.6}$$

First assume that $1 \leq k \leq \mu d$. Let $0 \leq i \leq k-1$. By using (6.6), the fact that the sequence $a_0, a_1, \ldots, a_k$ is non-decreasing, and (6.4), we obtain

$$\ell_i \leq \frac{\sqrt{d}(s+4)}{2} a_{i+1} \leq \frac{\sqrt{d}(s+4)}{2} a_k \leq \sqrt{d}(1+4/s)\ell_k.$$

Therefore,

$$\sum_{i=0}^{k} \ell_i \leq k\sqrt{d}(1+4/s)\ell_k + \ell_k \leq (k+1)\sqrt{d}(1+4/s)\ell_k \leq (\mu d+1)\sqrt{d}(1+4/s)\ell_k,$$

which is less than the right-hand side in (6.3).

It remains to consider the case when $k > \mu d$. Let $i \geq 0$ and $j \geq 0$ be integers such that $i+1+jd \leq k$. By applying (6.6) once, (6.5) j times, and (6.4) once, we obtain

$$\ell_i \leq \frac{\sqrt{d}(s+4)}{2} a_{i+1} \leq \frac{\sqrt{d}(s+4)}{2} \left(\frac{1}{2}\right)^j a_{i+1+jd} \leq \sqrt{d}(1+4/s) \left(\frac{1}{2}\right)^j \ell_{i+1+jd}.$$

For $j = \mu = \lceil \log(\sqrt{d}(1+4/s)) \rceil + 1$, this implies that, for $0 \leq i \leq k-1-\mu d$,

$$\ell_i \leq \frac{1}{2} \ell_{i+1+\mu d}. \tag{6.7}$$

By re-arranging the terms in the summation in (6.3), we obtain

$$\sum_{i=0}^{k} \ell_i = \sum_{h=0}^{\mu d} \sum_{j=0}^{\lfloor (k-h)/(\mu d+1) \rfloor} \ell_{k-h-j(\mu d+1)}.$$

Let j be such that $0 \leq j \leq \lfloor (k-h)/(\mu d+1) \rfloor$. By applying (6.7) j times, we obtain

$$\ell_{k-h-j(\mu d+1)} \leq \left(\frac{1}{2}\right)^j \ell_{k-h}.$$

It follows that

$$\sum_{j=0}^{\lfloor (k-h)/(\mu d+1) \rfloor} \ell_{k-h-j(\mu d+1)} \leq \sum_{j=0}^{\infty} \left(\frac{1}{2}\right)^j \ell_{k-h} = 2\ell_{k-h}.$$

Thus, we have

$$\sum_{i=0}^{k} \ell_i \leq 2 \sum_{h=0}^{\mu d} \ell_{k-h}.$$

By applying (6.6), the fact that the sequence $a_0, a_1, \ldots, a_k$ is non-decreasing, followed by (6.4), we obtain, for $0 \leq i \leq k-1$ and $1 \leq j \leq k-i$,

$$\ell_i \leq \frac{\sqrt{d}(s+4)}{2} a_{i+1} \leq \frac{\sqrt{d}(s+4)}{2} a_{i+j} \leq \sqrt{d}(1+4/s)\ell_{i+j}.$$

Obviously, the inequality $\ell_i \leq \sqrt{d}(1+4/s)\ell_{i+j}$ also holds for $j = 0$. Thus, for $i = k-h$ and $j = h$, we get

$$\ell_{k-h} \leq \sqrt{d}(1+4/s)\ell_k \text{ for } 0 \leq h \leq \mu d.$$

It follows that

$$\sum_{i=0}^{k} \ell_i \leq 2 \sum_{h=0}^{\mu d} \sqrt{d}(1+4/s)\ell_k = 2\sqrt{d}(\mu d+1)(1+4/s)\ell_k,$$

completing the proof that (6.3) holds. □

Lemma 6.12 *Assuming that the separation constant s of the WSPD is chosen sufficiently large, the graph G computed by Algorithm 6.1 is a t-spanner of the complete k-partite graph $K_{C_1 \ldots C_k}$, where $t = 2t' + 1 + 4/s$ and t' is as in Lemma 6.11.*

Proof: We denote the graph $K_{C_1 \ldots C_k}$ by K. It suffices to show that for each edge (p, q) of K, the graph G contains a path between p and q of length at most $t|pq|$. We prove this by induction on the lengths of the edges in K.

Let p and q be two points of S with different colors, and let $\{S_u, S_v\}$ be the pair in the MWSPD for which $p \in S_u$ and $q \in S_v$.

The base case is when (p, q) is a shortest edge in K. Since $s > 2$, it follows from Lemma 6.5 that u is a c-node and v is a c'-node, for some colors c and c' with $c \neq c'$. In line 20 of Algorithm 6.1, the edge $(\text{rep}(S_u), \text{rep}(S_v))$ is added to G. By Lemma 6.5, the length of this edge is at most $(1+4/s)|pq|$. The claim follows from two applications of Lemma 6.11 to get from p to $\text{rep}(S_u)$ and from $\text{rep}(S_v)$ to q.

In the induction step, we distinguish four cases.

Case 1: u is a c-node and v is a c'-node, for some colors c and c' with $c \neq c'$.

This case is identical to the base case.

Case 2: u is a c-node for some color c and v is a multichromatic node.

In lines 20–21, the edge $(\text{rep}(S_u), \text{rep}(S_v))$ or $(\text{rep}(S_u), \text{rep}(S'_v))$ is added to G. We may assume without loss of generality that $(\text{rep}(S_u), \text{rep}(S_v))$ is added. By Lemma 6.5, the length of this edge is at most $(1 + 4/s)|pq|$.

By Lemma 6.11, there is a path in G between p and $\text{rep}(S_u)$ whose length is at most $t'|pq|$.

First assume that q and $\text{rep}(S_v)$ have the same color. Let r be a point in S_v that has a color different from q's color. Since $s > 2$, it follows from Lemma 6.5 that $|qr| < |pq|$. Thus, by induction, there is a path in G between q and r whose length is at most $t|qr|$, which, by Lemma 6.5, is at most $(2t/s)|pq|$. By a similar argument, since $|r, \text{rep}(S_v)| < |pq|$, there is a path in G between r and $\text{rep}(S_v)$ whose length is at most $(2t/s)|pq|$. Thus, G contains a path between q and $\text{rep}(S_v)$ of length at most $(4t/s)|pq|$. If q and $\text{rep}(S_v)$ have different colors, then, by induction, there is a path in G between q and $\text{rep}(S_v)$ whose length is at most $(2t/s)|pq| < (4t/s)|pq|$.

Thus, the graph G contains a path between q and $\text{rep}(S_v)$ of length at most $(4t/s)|pq|$.

We showed that there is a path in G between p and q whose length is at most

$$(t' + (1 + 4/s) + 4t/s)\,|pq|. \tag{6.8}$$

By choosing s sufficiently large, this quantity is at most $t|pq|$.

Case 3: u is a multichromatic node and v is a c-node for some color c.

This case is symmetric to Case 2.

Case 4: Both u and v are multichromatic nodes.

In lines 33–34, the edge $(\text{rep}(S_u), \text{rep}(S_v))$ or $(\text{rep}(S_u), \text{rep}(S'_v))$ is added to G. We may assume without loss of generality that $(\text{rep}(S_u), \text{rep}(S_v))$ is added. By Lemma 6.5, the length of this edge is at most $(1 + 4/s)|pq|$.

As in Case 2, the graph G contains a path between p and $\text{rep}(S_u)$ of length at most $(4t/s)|pq|$, and a path between q and $\text{rep}(S_v)$ of length at most $(4t/s)|pq|$.

It follows that there is a path in G between p and q whose length is at most

$$((1 + 4/s) + 8t/s)\,|pq|. \tag{6.9}$$

By choosing s sufficiently large, this quantity is at most $t|pq|$. □

Lemma 6.13 *The running time of Algorithm 6.1 is $O(n \log n)$, where $n = |S|$.*

Proof: Using the results of Callahan and Kosaraju [18], the split-tree T and the WSPD can be computed in $O(n \log n)$ time. The representatives of all sets S_u and all sets $\mathrm{cl}(S_u)$ can be computed in $O(n)$ time by traversing the split-tree in post-order and pre-order, respectively. The time for the rest of the algorithm, i.e., lines 3–36, is proportional to the sum of the size of T, the number of pairs in the WSPD and the number of edges in the graph G. Thus, the rest of the algorithm takes $O(n)$ time. □

To summarize, we showed the following: For any complete k-partite geometric graph K whose vertex set has size n, Algorithm 6.1 computes a t-spanner of K having $O(n)$ edges, where t is given in Lemma 6.12. The running time of this algorithm is $O(n \log n)$. By choosing the separation constant s sufficiently large, the spanning ratio t converges to

$$8\sqrt{d}\left(d\left\lceil\frac{1}{2}\log d\right\rceil + d + 1\right) + 1.$$

In the next section, we show how to modify the algorithm so that the bound in Lemma 6.11 is reduced, thus improving the spanning ratio. The price to pay is in the number of edges in G, however, it is still $O(n)$.

6.5 An Improved Algorithm

As before, we are given a set S of n points in $\mathbb{R}^d$ which is partitioned into k subsets $C_1, C_2, \ldots, C_k$. Intuitively, the way to improve the bound of Lemma 6.11 is by adding shortcuts along the path from each c-leaf to the c-root above it. More precisely, from (6.7) in the proof of Lemma 6.11, we know that if we go $1 + \mu d$ levels up in the split-tree T, then the length of the edge along the path doubles. Thus, for each c-node in T, we add edges to all $2\delta(1+\mu d)$ c-nodes above it in T. Here, δ is an integer constant that is chosen such that the best result is obtained in the improved bound.

Definition 6.14 *Let $c \in \{1, 2, \ldots, k\}$, and let u and u' be c-nodes in the split-tree T such that u' is in the subtree rooted at u. For any integer $\zeta \geq 1$, we say that u is ζ* levels above *u', if there are exactly $\zeta - 1$ c-nodes on the path strictly between u and u'. We say that u' is a* ζ-level c-child *of u if u is at most ζ levels above u'.*

The improved algorithm is given as Algorithm 6.2. The following lemma generalizes Lemma 6.11.

Algorithm 6.2: Computing a sparse $(5+\epsilon)$-spanner of $K_{C_1 \ldots C_k}$.

Input: A set S of points in $\mathbb{R}^d$, which is partitioned into k subsets $C_1, \ldots, C_k$, and a real constant $0 < \epsilon < 1$.

Output: A $(5+\epsilon)$-spanner $G = (S, E)$ of the complete k-partite graph $K_{C_1 \ldots C_k}$.

Choose a separation constant s such that $s \geq 12/\epsilon$ and $(1 + 4/s)^2 \leq 1 + \epsilon/36$ and choose an integer constant δ such that $\frac{2^\delta}{2^\delta - 1} \leq 1 + \epsilon/36$.

The rest of the algorithm is the same as Algorithm 6.1, except for lines 23–27, which are replaced by the following:

$\zeta \leftarrow 2\delta(\mu d + 1)$;

for each ζ-level c-child u'' of u' **do**

  if $\text{rep}(\text{cl}(S_{u'}))$ does not have color c **then** add $(\text{rep}(S_{u''}), \text{rep}(\text{cl}(S_{u'})))$ to E;

  else add $(\text{rep}(S_{u''}), \text{rep}'(\text{cl}(S_{u'})))$ to E;

  if $\text{rep}(\text{cl}(S_{u''}))$ does not have color c **then** add $(\text{rep}(\text{cl}(S_{u''})), \text{rep}(S_{u'}))$ to E;

  else add $(\text{rep}'(\text{cl}(S_{u''})), \text{rep}(S_{u'}))$ to E;

end

Lemma 6.15 *Let G be the graph computed by Algorithm 6.2. Let p and q be two points of S with different colors, and let $\{S_u, S_v\}$ be the pair in the MWSPD for which $p \in S_u$ and $q \in S_v$. Assume that u is a c-node for some color c. Then there is a path in G between p and $\text{rep}(S_u)$ whose length is at most $(2 + \epsilon/3)|pq|$.*

Proof: Let w be the c-leaf such that $p \in S_w$, and let $w = w_0, w_1, \ldots, w_k = u$ be the sequence of c-nodes that are on the path in T from w to u. As in the proof of

Lemma 6.11, we assume without loss of generality that, for all $0 \leq i \leq k$, the color of $\text{rep}(\text{cl}(S_{w_i}))$ is not equal to c.

Throughout the proof, we use the variables x_i, y_i, ℓ_i, and a_i, for $0 \leq i \leq k$, that were introduced in the proof of Lemma 6.11.

We first assume that $0 \leq k \leq 2\delta(\mu d + 1)$. Let Π be the path

$$p \to \text{rep}(\text{cl}(S_w)) \to \text{rep}(S_u).$$

It follows from Algorithm 6.2 that Π is a path in G. Since $p \in S_w = S_{w_0} \subseteq S_{x_0}$ and $\text{rep}(\text{cl}(S_w)) = \text{rep}(\text{cl}(S_{w_0})) \in S_{y_0}$, it follows from Lemma 6.5 that

$$|p, \text{rep}(\text{cl}(S_w))| \leq (1 + 4/s) \cdot \text{dist}(S_{x_0}, S_{y_0}) = (1 + 4/s)\ell_0. \tag{6.10}$$

Since $\{S_u, S_v\}$ is one of the pairs that is considered in the definition of $\text{cl}(S_{w_0})$, we have $\text{dist}(S_{x_0}, S_{y_0}) \leq \text{dist}(S_u, S_v)$. Again by Lemma 6.5, we have $\text{dist}(S_u, S_v) \leq (1 + 4/s)|pq|$. Thus, we showed that

$$|p, \text{rep}(\text{cl}(S_w))| \leq (1 + 4/s)^2|pq|.$$

By the triangle inequality, we have

$$|\text{rep}(\text{cl}(S_w)), \text{rep}(S_u)| \leq |\text{rep}(\text{cl}(S_w)), p| + |p, \text{rep}(S_u)|.$$

Since p and $\text{rep}(S_u)$ are both contained in S_u, it follows from Lemma 6.5 that $|p, \text{rep}(S_u)| \leq (2/s)|pq|$. Thus, we have

$$|\text{rep}(\text{cl}(S_w)), \text{rep}(S_u)| \leq (1 + 4/s)^2|pq| + (2/s)|pq|.$$

We showed that the length of the path Π is at most

$$\left(2(1 + 4/s)^2 + 2/s\right)|pq|,$$

which is at most $(2 + \epsilon/3)|pq|$ by our choice of s in Algorithm 6.2.

In the rest of the proof, we assume that $k > 2\delta(\mu d + 1)$. We define

$$m = k \bmod (\delta(\mu d + 1))$$

and

$$m' = \frac{k - m}{\delta(\mu d + 1)}.$$

We consider the sequence of c-nodes

$$w = w_0, w_{\delta(\mu d+1)+m}, w_{2\delta(\mu d+1)+m}, w_{3\delta(\mu d+1)+m}, \ldots, w_k = u,$$

and define Π to be the path

$$\begin{array}{rcccl}
p & \rightarrow & \mathrm{rep}(\mathrm{cl}(S_{w_0})) & \rightarrow & \mathrm{rep}(S_{w_{\delta(\mu d+1)+m}}) \\
& \rightarrow & \mathrm{rep}(\mathrm{cl}(S_{w_{2\delta(\mu d+1)+m}})) & \rightarrow & \mathrm{rep}(S_{w_{2\delta(\mu d+1)+m}}) \\
& \rightarrow & \mathrm{rep}(\mathrm{cl}(S_{w_{3\delta(\mu d+1)+m}})) & \rightarrow & \mathrm{rep}(S_{w_{3\delta(\mu d+1)+m}}) \\
& \vdots & & \vdots & \\
& \rightarrow & \mathrm{rep}(\mathrm{cl}(S_{w_k})) & \rightarrow & \mathrm{rep}(S_{w_k}) \quad = \quad \mathrm{rep}(S_u).
\end{array}$$

It follows from Algorithm 6.2 that Π is a path in G. We show that the length of this path is at most $(2+\epsilon/3)|pq|$.

We already showed (see (6.10)) that the length of the first edge on Π satisfies

$$|p, \mathrm{rep}(\mathrm{cl}(S_{w_0}))| \leq (1+4/s)\ell_0.$$

The length of the second edge satisfies

$$\begin{array}{rcl}
|\mathrm{rep}(\mathrm{cl}(S_{w_0})), \mathrm{rep}(S_{w_{\delta(\mu d+1)+m}})| & \leq & |\mathrm{rep}(\mathrm{cl}(S_{w_0})), p| + |p, \mathrm{rep}(S_{w_{\delta(\mu d+1)+m}})| \\
& \leq & (1+4/s)\ell_0 + |p, \mathrm{rep}(S_{w_{\delta(\mu d+1)+m}})|.
\end{array}$$

Since p and $\mathrm{rep}(S_{w_{\delta(\mu d+1)+m}})$ are both contained in S_u, it follows from Lemma 6.5 that

$$|p, \mathrm{rep}(S_{w_{\delta(\mu d+1)+m}})| \leq (2/s)|pq|.$$

Thus, the length of the second edge on Π satisfies

$$|\mathrm{rep}(\mathrm{cl}(S_{w_0})), \mathrm{rep}(S_{w_{\delta(\mu d+1)+m}})| \leq (1+4/s)\ell_0 + (2/s)|pq|.$$

Let $2 \leq j \leq m'$. We saw in (6.1) in the proof of Lemma 6.11 that the length of the edge

$$(\mathrm{rep}(\mathrm{cl}(S_{w_{j\delta(\mu d+1)+m}})), \mathrm{rep}(S_{w_{j\delta(\mu d+1)+m}}))$$

satisfies

$$|\mathrm{rep}(\mathrm{cl}(S_{w_{j\delta(\mu d+1)+m}})), \mathrm{rep}(S_{w_{j\delta(\mu d+1)+m}})| \leq (1+4/s)\ell_{j\delta(\mu d+1)+m}.$$

Again, let $2 \leq j \leq m'$. Since

$$\text{rep}(S_{w_{(j-1)\delta(\mu d+1)+m}}) \in S_{w_{j\delta(\mu d+1)+m}} \subseteq S_{x_{j\delta(\mu d+1)+m}}$$

and

$$\text{rep}(\text{cl}(S_{w_{j\delta(\mu d+1)+m}})) \in S_{y_{j\delta(\mu d+1)+m}}$$

it follows from Lemma 6.5 that the length of the edge

$$(\text{rep}(S_{w_{(j-1)\delta(\mu d+1)+m}}), \text{rep}(\text{cl}(S_{w_{j\delta(\mu d+1)+m}})))$$

satisfies

$$|\text{rep}(S_{w_{(j-1)\delta(\mu d+1)+m}}), \text{rep}(\text{cl}(S_{w_{j\delta(\mu d+1)+m}}))| \leq (1+4/s)\ell_{j\delta(\mu d+1)+m}.$$

We showed that the length of Π is at most

$$(2/s)|pq| + 2(1+4/s)\left(\ell_0 + \sum_{j=2}^{m'} \ell_{j\delta(\mu d+1)+m}\right).$$

The definition of $\ell_0, \ell_1, \ldots, \ell_k$ implies that this sequence is non-decreasing. Thus, $\ell_0 \leq \ell_{\delta(\mu d+1)+m}$ and it follows that the length of Π is at most

$$(2/s)|pq| + 2(1+4/s)\sum_{j=1}^{m'} \ell_{j\delta(\mu d+1)+m}.$$

Recall inequality (6.7) in the proof of Lemma 6.11, which states that

$$\ell_i \leq \frac{1}{2}\ell_{i+\mu d+1}.$$

By applying this inequality δ times, we obtain

$$\ell_i \leq \left(\frac{1}{2}\right)^{\delta} \ell_{i+\delta(\mu d+1)}.$$

For $i = j\delta(\mu d+1)+m$, this becomes

$$\ell_{j\delta(\mu d+1)+m} \leq \left(\frac{1}{2}\right)^{\delta} \ell_{(j+1)\delta(\mu d+1)+m}.$$

By repeatedly applying this inequality, we obtain, for $h \geq j$,

$$\ell_{j\delta(\mu d+1)+m} \leq \left(\frac{1}{2}\right)^{(h-j)\delta} \ell_{h\delta(\mu d+1)+m}.$$

For $h = m'$, the latter inequality becomes

$$\ell_{j\delta(\mu d+1)+m} \leq \left(\frac{1}{2}\right)^{(m'-j)\delta} \ell_k.$$

It follows that

$$\begin{aligned}
\sum_{j=1}^{m'} \ell_{j\delta(\mu d+1)+m} &\leq \sum_{j=1}^{m'} \left(\frac{1}{2}\right)^{(m'-j)\delta} \ell_k \\
&= \sum_{i=0}^{m'-1} \left(\frac{1}{2}\right)^{i\delta} \ell_k \\
&\leq \sum_{i=0}^{\infty} \left(\frac{1}{2^\delta}\right)^{i} \ell_k \\
&= \frac{2^\delta}{2^\delta - 1} \ell_k.
\end{aligned}$$

According to (6.2) in the proof of Lemma 6.11, we have

$$\ell_k \leq (1 + 4/s)|pq|.$$

We showed that the length of the path Π is at most

$$\left(2/s + 2(1 + 4/s)^2 \frac{2^\delta}{2^\delta - 1}\right) |pq|.$$

Our choices of s and δ (see Algorithm 6.2) imply that $2/s \leq \epsilon/6$, $(1+4/s)^2 \leq 1+\epsilon/36$ and $\frac{2^\delta}{2^\delta-1} \leq 1 + \epsilon/36$. Therefore, the length of Π is at most

$$\left(\epsilon/6 + 2(1 + \epsilon/36)^2\right) |pq| \leq (2 + \epsilon/3)|pq|,$$

where the latter inequality follows from our assumption that $0 < \epsilon < 1$. This completes the proof. □

Lemma 6.16 *Let $n = |S|$. The graph G computed by Algorithm 6.2 is a $(5 + \epsilon)$-spanner of the complete k-partite graph $K_{C_1 \ldots C_k}$ and the number of edges of this graph is $O(n)$. The running time of Algorithm 6.2 is $O(n \log n)$.*

Proof: The proof for the upper bound on the spanning ratio is similar to the one of Lemma 6.12. The difference is that instead of the value t' that was used in the proof of Lemma 6.12, we now use the value $t' = 2 + \epsilon/3$ of Lemma 6.15. Thus, the spanning ratio for the base case of the induction and for Case 1 is at most

$$(1 + 4/s) + 2t' = 5 + 4/s + 2\epsilon/3,$$

which is at most $5 + \epsilon$, because of our choice for s in Algorithm 6.2. For Cases 2 and 3, the spanning ratio is at most (see (6.8) in the proof of Lemma 6.12, where $t = 5 + \epsilon$)

$$t' + (1 + 4/s) + 4t/s = 3 + \epsilon/3 + (4/s)(6 + \epsilon),$$

which is at most $5 + \epsilon$, again because of our choice for s. Finally, the spanning ratio for Case 4 is at most (see (6.9) in the proof of Lemma 6.12, where $t = 5 + \epsilon$)

$$(1 + 4/s) + 8t/s = 1 + (4/s)(11 + 2\epsilon),$$

which is at most $5 + \epsilon$, because of our choice for s.

The analysis for the number of edges is the same as in Lemma 6.10, except that the number of edges that are added to each c-node in the modified for-loop is $2\delta(\mu d + 1)$ instead of one as is in Algorithm 6.1. Finally, the analysis of the running time is the same as in Lemma 6.13. □

We proved the following result.

Theorem 6.17 *Let $k \geq 2$ be an integer, let S be a set of n points in $\mathbb{R}^d$ which is partitioned into k subsets $C_1, C_2, \ldots, C_k$, and let $0 < \epsilon < 1$ be a real constant. In $O(n \log n)$ time, we can compute a $(5 + \epsilon)$-spanner of the complete k-partite graph $K_{C_1 \ldots C_k}$ having $O(n)$ edges.*

6.6 Improving the Spanning Ratio

We showed how to compute a $(5 + \epsilon)$-spanner with $O(n)$ edges of any complete k-partite graph. In this section, we show that if we are willing to use $O(n \log n)$ edges, the spanning ratio can be reduced to $3 + \epsilon$. We start by showing that a spanning ratio less than 3, while using $O(n \log n)$ edges, is not possible.

Theorem 6.18 *Let $c > 0$ be a constant and let n and k be positive integers with $2 \leq k \leq n - 2c\sqrt{n \log n}$. For every real number $0 < \epsilon < 1$, there exists a complete k-partite geometric graph K with n vertices such that the following is true: If G is any subgraph of K with at most $c^2 n \log n$ edges, then the spanning ratio of G is at least $3 - \epsilon$.*

Proof: Let D_1, D_2, and D_3 be three disks of radius $\epsilon/12$ centered at the points $(0, 0)$, $(1 + \epsilon/6, 0)$, and $(2 + \epsilon/3, 0)$, respectively. We place $(n - k + 2)/2$ red points inside D_1 and $(n - k + 2)/2$ blue points inside D_2. The remaining $k - 2$ points are placed inside D_3 and each of these points has a distinct color (which is not red or blue). Let K be the complete k-partite geometric graph defined by these n points. We claim that K satisfies the claim in the theorem.

Let G be an arbitrary subgraph of K and assume that G contains at most $c^2 n \log n$ edges. We show that the spanning ratio of G is at least $3 - \epsilon$.

Assume that G contains all red-blue edges. Then the number of edges in G is at least $((n-k+2)/2)^2$. Since $k \leq n - 2c\sqrt{n \log n}$, this quantity is larger than $c^2 n \log n$. Thus, there is a red point r and a blue point b, such that (r, b) is not an edge in G. The length of a shortest path in G between r and b is at least 3. Since $|rb| \leq 1 + \epsilon/3$, it follows that the spanning ratio of G is at least $\frac{3}{1+\epsilon/3}$, which is at least $3 - \epsilon$. □

Theorem 6.19 *Let $k \geq 2$ be an integer, let S be a set of n points in $\mathbb{R}^d$ which is partitioned into k subsets $C_1, C_2, \ldots, C_k$, and let $0 < \epsilon < 1$ be a real constant. In $O(n \log n)$ time, we can compute a $(3 + \epsilon)$-spanner of the complete k-partite graph $K_{C_1 \ldots C_k}$ having $O(n \log n)$ edges.*

Proof: Consider the following variant of the WSPD. For every pair $\{X, Y\}$ in the standard WSPD, where $|X| \leq |Y|$, we replace this pair by the $|X|$ pairs $\{\{x\}, Y\}$, where x ranges over all points of X. Thus, in this new WSPD, each pair contains at least one singleton set. Callahan and Kosaraju [18] showed that this new WSPD consists of $O(n \log n)$ pairs.

We run Algorithm 6.2 on the set S, using this new WSPD. Let G be the graph that is computed by this algorithm. Observe that Lemma 6.15 still holds for G. In

the proof of Lemma 6.16 of the upper bound on the spanning ratio of G, we have to apply Lemma 6.15 only once. Therefore, the spanning ratio of G is at most $3+\epsilon$. □

6.7 Conclusion

We showed that for every complete k-partite geometric graph K in $\mathbb{R}^d$ with n vertices and for every constant $\epsilon > 0$,

1. a $(5+\epsilon)$-spanner of K having $O(n)$ edges can be computed in $O(n \log n)$ time,
2. a $(3+\epsilon)$-spanner of K having $O(n \log n)$ edges can be computed in $O(n \log n)$ time.

The latter result is optimal for $2 \leq k \leq n - \Theta(\sqrt{n \log n})$, because a spanner of K having spanning ratio less than 3 and having $O(n \log n)$ edges does not exist for all complete k-partite geometric graphs.

We leave open the problem of determining the best spanning ratio that can be obtained by using $O(n)$ edges.

Future work may include verifying other properties that are known for the general geometric spanner problem. For example, is there a spanner of a complete k-partite geometric graph that has bounded degree? Is there a spanner of a complete k-partite geometric graph that is planar? From a more general point of view, it seems that little is known about geometric spanners of graphs other than the complete graph. The unit disk graph received great attention, but there are a large family of other graphs that also deserve attention.

Chapter 7

Spanners of Additively Weighted Point Sets

7.1 Introduction

It has been claimed (see [5, 77, 78]) that geometric spanners can be used to address the link selection problem in wireless networks. In most cases, however, two assumptions are made:

1. nodes can be represented as points in the plane and
2. the cost of routing a message is a function of the length of the links that are successively used.

However, these assumptions do not always hold. For example, the first assumption does not hold in the case of wide area mesh networks, where nodes are vast areas such as villages [73]. The second assumption does not take into account the fact that some nodes may have higher energy resources or introduce more delay than others. In such cases, an additional cost must be taken into account for each node. The study of spanners of additively weighted point sets is a first step in addressing some of these issues.

In this chapter, we address the problem of computing geometric spanners with additive constraints on the points. More precisely, we define a weighted point set as a set of pairs (p, r) where p is a point in the plane and r is a real number. The distance between two points (p_i, r_i) and (p_j, r_j) is defined as $|p_i p_j| - r_i - r_j$. The problem we address is to compute a spanner of a complete graph on a weighted point set. To the best of our knowledge, the problem of constructing a geometric spanner in this context has not been previously addressed. We show how the Yao graph can be adapted to compute a $(1 + \epsilon)$-spanner in the case where all r_i are positive real numbers and $|p_i p_j| \geq r_i + r_j$ for all i, j (in which case the points can be seen as non-intersecting disks in the plane). In the same case, we also show how the Additively Weighted

Delaunay graph (the face-dual of the Additively Weighted Voronoi diagram) provides a plane spanner that has the same spanning ratio as the Delaunay graph of a set of points.

The rest of this chapter is divided as follows: In Section 7.2, we review related work. In Section 7.3, we give a formal definition of our problem and show that it is not solved by a straightforward extension of the Yao graph. However, in Section 7.4, we show that a minor adjustment to the Yao graph allows to compute a $(1+\epsilon)$-spanner. In Section 7.5, we develop some tools used in Section 7.6 to show that the Additively Weighted Delaunay graph has a constant spanning ratio. In Section 7.7, we show how to compute a plane embedding of the Additively Weighted Delaunay graph that is also a spanner. We conclude in Section 7.8.

7.2 Related Work

Well known examples of geometric t-spanners include the Yao graph [91], the Well-Separated Pair Decomposition (WSPD) [18], and the Delaunay graph [48]. The Yao graph has been reviewed in Chapter 4, and the WSPD has been reviewed in Chapter 6.

Given a set of points in the plane, there is an edge between p and q in the Delaunay graph if and only if there is an empty circle with p and q on its boundary [48]. If no four points are cocircular, then the Delaunay graph is a uniquely defined near-triangulation. Otherwise, four or more co-circular points may create crossings. In that case, removing edges that cause crossings leads to *a* Delaunay triangulation. Since our results hold for any Delaunay triangulation, when we refer to *the* Delaunay triangulation in the case of co-circular points, we mean *any* Delaunay triangulation. Dobkin *et al.* [29] showed that the Delaunay triangulation has a spanning ratio of at most $\frac{1+\sqrt{5}}{2}\pi \approx 5.08$. This result was improved by Keil and Gutwin [48], who showed that the spanning ratio of the Delaunay triangulation is at most $2\pi/(3\cos(\pi/6)) \approx 2.42$. Later, Bose *et al.* [14] showed that the Delaunay triangulation is also a strong t-spanner for the same constant $t = 2\pi/(3\cos(\pi/6))$. Although the exact spanning ratio of the Delaunay triangulation is unknown, it is conjectured that the spanning ratio is $\pi/2$. For the remainder of this chapter, we will refer to the spanning ratio of the Delaunay triangulation as the spanning ratio of the *standard* Delaunay triangulation

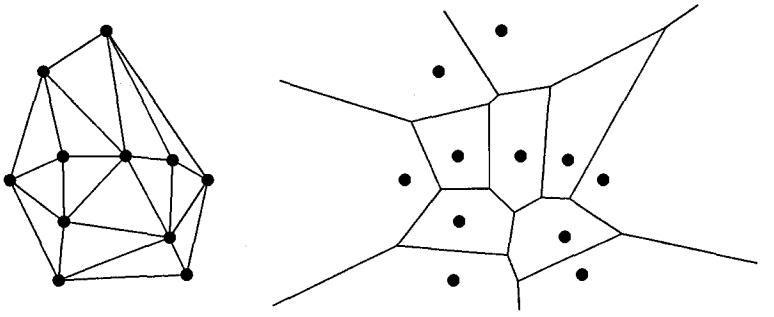


Figure 7.1: The Delaunay graph and its dual: the Voronoi diagram.

and denote it as SP-DT.

The *Voronoi diagram* [28] of a finite set of points P is a partition of the plane into $|P|$ regions such that each region contains exactly those points having the same nearest neighbor in P. The points in P are also called *sites*. It is well known that the Voronoi diagram of a set of points is the face dual of the Delaunay graph of that set of points [28], i.e. two points have adjacent Voronoi regions if and only if they share an edge in the Delaunay graph (see Figure 7.1).

Most of the work on computing geometric spanners is about spanners of the complete Euclidean graph on a set of points. Our work falls in the context of computing spanners for geometric graphs other than the complete Euclidean graph. Typically, variations of the spanner problem arise by either changing the distance function or removing edges from the complete graph. In Chapter 4, we showed how to compute spanners of the unit disk graph. Unit disk graphs can be seen as intersection graphs of disks of same radius in the plane. The general problem of computing spanners for geometric intersection graphs has been studied by Furer and Kasiviswanathan [33]. In Chapter 6, we showed how to compute spanners of complete k-partite graphs.

Another example of a graph that has been studied is the visibility graph. For a set P of points in the plane and a set C of non-intersecting line segments whose endpoints are in P, the *visibility graph* of P with respect to C is the geometric graph with vertex set P and there is an edge (p, q) if and only if the segment $\overline{pq}$ is in C or it does not cross any segment in C (in that case, p and q are said to be *visible*). A spanner of the visibility graph should then approximate Euclidean distances for every pair of points that are visible from each other. The constrained Delaunay triangulation (a variation

of the Delaunay triangulation) is a 2.42-spanner of the visibility graph [13].

In the literature, spanners that use a distance other than the Euclidean distance have also been proposed. For example, in a *power* spanner [7, 38, 58, 77], the distance used to measure the length of an edge is the square of the Euclidean distance between its two end points. This models the fact that in wireless networks, the amount of energy needed to send a packet is proportional to a power (not necessarily the square, however) of the Euclidean distance between the sender and receiver [68]. When reducing the latency is more important than reducing the amount of energy being used, a *hop* spanner [5], which gives an equal weight to every edge, can be used.

In this chapter, the Additively Weighted Voronoi diagram (AW-Voronoi diagram) is of particular interest. Its definition is similar to that of the (standard) Voronoi diagram, except that each site p_i is assigned a weight which is a real number r_i. Weights are used to define a weighted distance. More detail about how the weighted distance is used to define the AW-Voronoi diagram is given in Section 7.6. The Additively Weighted Delaunay graph (AW-Delaunay graph) is defined as the face-dual of the AW-Voronoi diagram. Properties of the AW-Voronoi diagram and its dual have been studied by Lee and Drysdale [57], who showed how to compute it in $O(n \log^2 n)$ time. Later on, Fortune [32] showed how to compute it in $O(n \log n)$ time. The AW-Voronoi diagram may have empty cells. For this reason, one would hope that it is possible to design an algorithm whose running time gets better as the number of empty cells increases. Karavelas and Yvinec [46] provided an $O(nT(h) + h \log h)$ time algorithm to compute the AW-Voronoi diagram where h is the number of non-empty cells and $T(h)$ is the time to locate the nearest neighbor of a query point within a set of h points. Experimental results suggested an $O(n \log h)$ behavior. In 3D, the complexity of the (Additively Weighted) Voronoi diagram is $\Theta(n^2)$ [51]. Aurenhammer [7] showed how to compute it in time $O(n^2)$ using Power Voronoi diagrams. Will [85] gave an $O(n^2 \log n)$ time algorithm with experimental results suggesting an $O(n \log^2 n)$ time behavior in the expected case. Kim *et al.* [49] showed how to obtain a running time of $O(nm)$, where m is the number of edges.

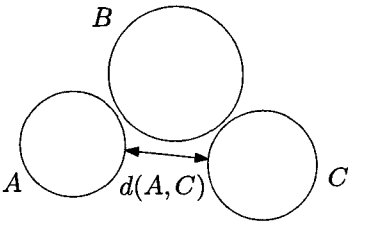


Figure 7.2: The additively weighted distance is not a metric.

7.3 Definitions and Notation

Definition 7.1 *A set $P = \{(p_1, r_1), \ldots, (p_n, r_n)\}$ of ordered pairs, where each p_i is a point in the plane and each r_i is a real number, is called a* weighted point set. *The notation $p_i \in P$ means that there exists an ordered pair (p_i, r_i) such that $(p_i, r_i) \in P$. The* additive distance *from a point $p \notin P$ in the plane to a point $p_i \in P$, noted $d(p, p_i)$, is defined as $|pp_i| - r_i$, where $|pp_i|$ is the Euclidean distance from p to p_i. The* additive distance *between two points $p_i, p_j \in P$, noted $d(p_i, p_j)$, is defined as $|p_ip_j| - r_i - r_j$, where $|p_ip_j|$ is the Euclidean distance from p_i to p_j.*

The problem we address in this chapter is the following:

Problem 7.2 *Let P be a weighted point set and let $K(P)$ be the complete weighted graph with vertex set P and edges weighted by the additive distance between their endpoints. Compute a t-spanner with $O(n)$ edges of $K(P)$ for a fixed constant $t > 1$.*

Notice that in the case where all r_i are positive numbers, the pairs (p_i, r_i) can be viewed as disks D_i in the plane. If, for all i, j we also have $d(p_i, p_j) \geq 0$, then the disks are disjoint. In that case, the distance $d(D_i, D_j) = d(p_i, p_j) = |p_ip_j| - r_i - r_j$ is also equal to $\min\{|q_iq_j| : q_i \in D_i \text{ and } q_j \in D_j\}$, where the notation $q_i \in D_i$ means $|p_iq_i| \leq r_i$. To compute a spanner of an additively weighted point set is then equivalent to computing a spanner of a set of disks in the plane. **From now to the end of this chapter, it is assumed that all r_i are positive numbers and $d(p_i, p_j) \geq 0$ for all i, j.** If $\mathcal{D}$ is a set of disks in the plane, then a *spanner* of $\mathcal{D}$ is a spanner of the complete graph whose vertex set is $\mathcal{D}$ and whose edges (D_i, D_j) are given weights equal to $d(D_i, D_j)$.

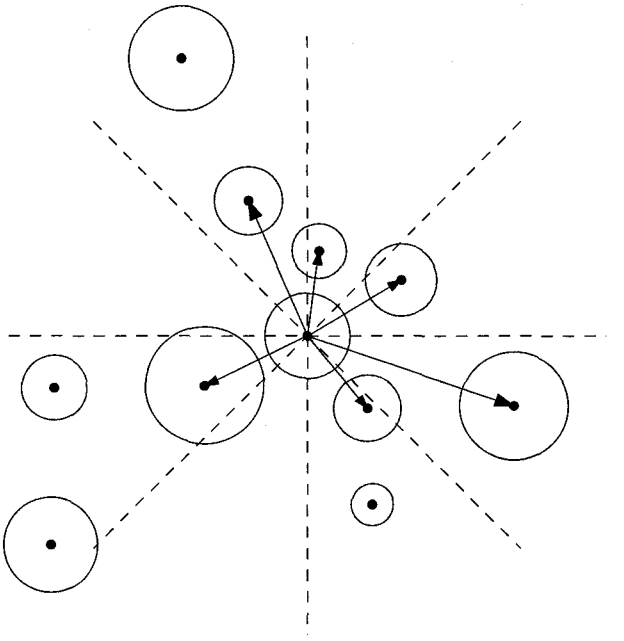


Figure 7.3: A straightforward generalization of the Yao graph.

Notice also that the additive distance may not be a metric since the triangle inequality does not necessarily hold (see Figure 7.2). Although this may seem counter-intuitive, this makes sense in some networks, since a direct communication is not always easier than routing through a common neighbor. For example, in wireless networks, the amount of energy that is needed to transmit a message is a power of the Euclidean distance between the sender and the receiver. Therefore, using several small hops can be more energy efficient than a direct communication over one long-distance link.

Figure 7.3 shows how the Yao graph can be generalized using the additive distance: for each cone, a disk keeps an outgoing edge with the closest disk whose center is contained in that cone. However, this graph is not a spanner. Figure 7.4 shows how to construct an example with six disks that has spanning ratio of $(1+\epsilon)/\epsilon$ for any $\epsilon > 0$. Nonetheless, in Section 7.4, we see that a minor adjustment to the Yao graph can be made in order to compute a $(1+\epsilon)$-spanner of a set of disjoint disks that has $O(n)$ edges.

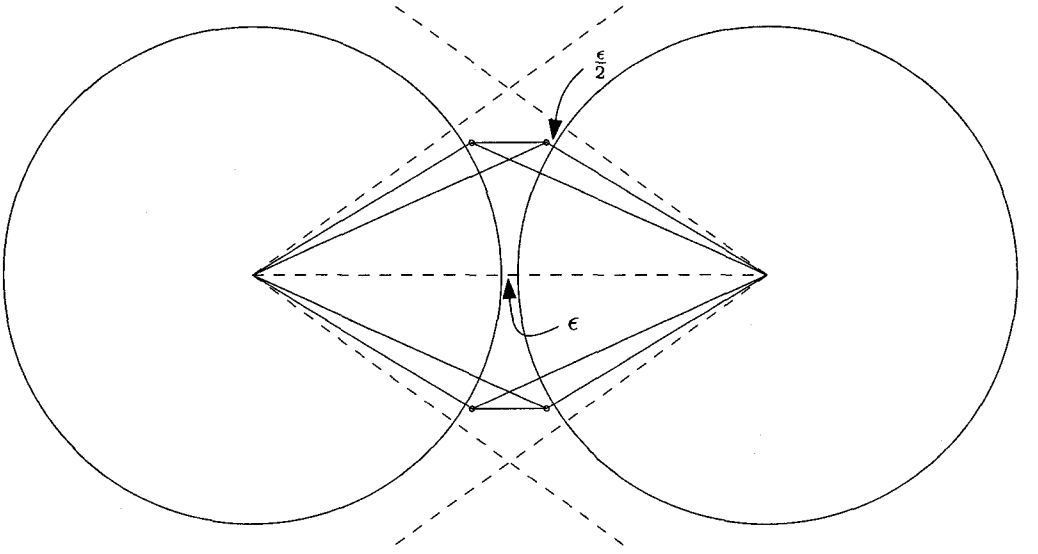


Figure 7.4: The straightforward generalization of the Yao graph does not have constant spanning ratio.

The Delaunay graph in the additively weighted setting is computable in time $O(n \log n)$ [32]. To the best of our knowledge, its spanning properties have not been previously studied. In Section 7.6, we show that it is a spanner and that its spanning ratio is SP-DT (i.e the same as that of the standard Delaunay graph).

7.4 The Additively Weighted Yao Graph

As we saw in the previous section, a straightforward generalization of the Yao graph fails to provide a graph with bounded spanning ratio. In this section, we show how a few subtle modifications to the construction, provide an approach to build a $(1+\epsilon)$-spanner. We define the modified Yao construction below.

Definition 7.3 *Let $\mathcal{D}$ be a finite set of disjoint disks and $\theta \leq 0.228$ be an angle such that $2\pi/\theta = k$, where k is an integer. The* $\text{Yao}(\theta, \mathcal{D})$ *graph is defined as follows. For every disk $D = (p, r)$, partition the plane into k cones $C_{p,1}, \ldots, C_{p,k}$ of angle θ and apex p. A disk* blocks *a cone $C_{p,i}$ provided that the disk intersects both rays of $C_{p,i}$. Let $F \in \mathcal{D}$ be a disk different from D with center in $C_{p,j}$. Add an edge from D to F in* $\text{Yao}(\theta, \mathcal{D})$ *if and only if one of the two following conditions is met:*

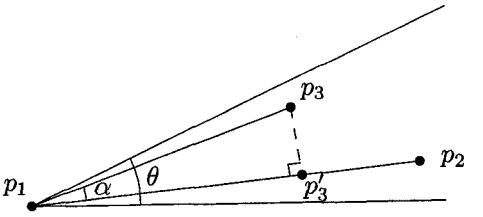


Figure 7.5: Illustration of the proof of Lemma 7.4.

1. *among all blocking disks that have their center in $C_{p,j}$, F is the one that is the closest to D;*

2. *among all disks that have their center in $C_{p,j}$ and are at a distance of at least r from D, F is the one that is the closest to D.*

Notice that there are two main changes. Within each cone, we now add potentially two edges as opposed to only one edge in the case of unweighted points. Next, in the second condition to add an edge, we do not add an edge to the closest disk within a cone but to the closest disk whose distance is at least r from the disk centered at the apex with radius r. We now prove that these two modifications imply that the resulting graph is a $(1+\epsilon)$-spanner.

Lemma 7.4 *Let p_1, p_2, p_3 such that the angle $\angle p_3 p_1 p_2 = \alpha \leq \theta < \pi/4$ and $|p_1p_3| \leq |p_1p_2|$. Then $|p_2p_3| \leq |p_1p_2| - (\cos\theta - \sin\theta)|p_1p_3|$.*

Proof: Let p'_3 be the projection of p_3 on the line through p_1 and p_2 (see Figure 7.5). Then

$$\begin{aligned} |p_2p_3| &\leq |p_2p'_3| + |p'_3p_3| \\ &= |p_1p_2| - |p_1p'_3| + |p'_3p_3| \\ &= |p_1p_2| - |p_1p_3|(\cos\alpha - \sin\alpha) \\ &\leq |p_1p_2| - |p_1p_3|(\cos\theta - \sin\theta) \end{aligned}$$

□

Theorem 7.5 *Let $\mathcal{D}$ be a finite set of disjoint disks and $\theta \leq 0.228$. Then $Y(\theta, \mathcal{D})$ is a t-spanner of $\mathcal{D}$, where $t = 1/(\cos 2\theta - \sin 2\theta - 2\sin(\theta/2))$.*

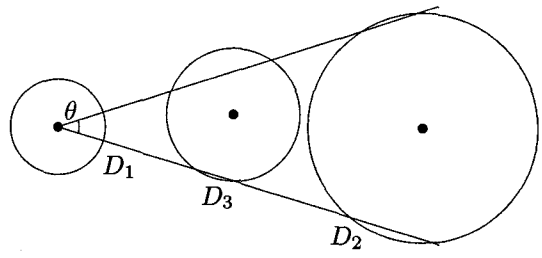


Figure 7.6: If D_2 blocks the cone but the edge (D_1, D_2) is not in Yao$(\theta, \mathcal{D})$, then there exists D_3 such that $d(D_1, D_3) + d(D_3, D_2) < d(D_1, D_2)$.

Proof: We proceed by induction on the rank of the distances between the pairs of disks D_1 and D_2.

Base case: The disks D_1 and D_2 form a closest pair. In that case, the edge (D_1, D_2) is in Yao$(\theta, \mathcal{D})$. To see this, let $r_1 \leq r_2$. If D_2 is blocking the cone centered at p_1 that contains it, then it is in Yao$(\theta, \mathcal{D})$ by Case 1 of Definition 7.3. Otherwise, then it is at distance at least r_1 from D_1 and therefore it is in Yao$(\theta, \mathcal{D})$ by Case 2 of Definition 7.3.

Induction case: Let $D_1 = (p_1, r_1)$ and $D_2 = (p_2, r_2)$. Without loss of generality, $r_1 \leq r_2$. If the edge (D_1, D_2) is in Yao$(\theta, \mathcal{D})$, then there is nothing to prove. Otherwise, there are two cases to consider depending on whether or not the shortest path from D_1 to D_2 in the complete graph on $\mathcal{D}$ is the edge (D_1, D_2). If the shortest path is not the edge (D_1, D_2), then all edges on the shortest path must have length less than $d(D_1, D_2)$. By applying the induction hypothesis on each of those edges, we conclude that the distance from D_1 to D_2 in Yao$(\theta, \mathcal{D})$ is at most t times the length of the shortest path D_1 to D_2 in the complete graph on $\mathcal{D}$, as required.

We now consider the case when the edge (D_1, D_2):

1. is not in Yao$(\theta, \mathcal{D})$ and

2. is the shortest path from D_1 to D_2 in the complete graph.

Observe that the conjunction of those two facts imply that the disk D_2 does not block the cone whose apex is p_1 and contains p_2: If D_2 was blocking the cone, then since (D_1, D_2) is not an edge in Yao$(\theta, \mathcal{D})$, there must be a disk D_3 that is also blocking the cone and is closer to D_1 than D_2. However, this implies that the shortest path from D_1 to D_2 in the complete graph is not the edge (D_1, D_2) (see Figure 7.6).

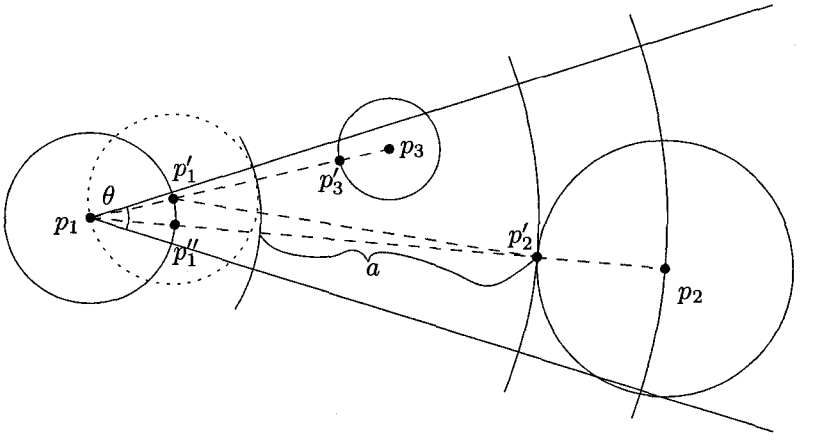


Figure 7.7: Illustration of the proof of Theorem 7.5.

The conjunction of the three following facts:

1. $r_1 \leq r_2$;

2. $\theta \leq 0.228 < \sin^{-1}(1/3)$ and

3. D_2 does not block the cone,

imply that $d(D_1, D_2) > r_1$. Since (D_1, D_2) is not an edge, there is another disk whose distance is at least r that is closer to D_1. Let $D_3 = (p_3, r_3)$ be the closest disk to D_1 such that p_3 is in the same θ-cone with apex at p_1 as p_2 and $d(D_1, D_3) \geq r_1$. By definition, the edge (D_1, D_3) is in $\text{Yao}(\theta, \mathcal{D})$. Observe that $d(D_2, D_3) < d(D_1, D_2)$. To see this, let $a := d(D_1, D_2) - r_1$. We have that

$$d(D_2, D_3) \leq a + 4r_1 \sin(\theta/2) \leq a + 4r_1 \sin(0.114) < a + r_1 = d(D_1, D_2).$$

Let p_1' be the point of D_1 that is the closest to D_3, p_1'' be the point of D_1 that is the closest to D_2, p_2' be the point of D_2 that is the closest to D_1, and p_3' be the point of D_3 that is the closest to D_1 (see Figure 7.7). Notice that $|p_1'p_3'| \leq |p_1'p_2'|$ and that since $d(D_1, D_2) \geq d(D_1, D_3) \geq r_1$, then the angle $\angle p_2'p_1'p_3'$ is at most $2\theta < \pi/4$. Therefore, we can apply Lemma 7.4 to conclude that

$$|p_2'p_3'| \leq |p_1'p_2'| - (\cos 2\theta - \sin 2\theta)|p_1'p_3'|,$$

which implies that

$$d(D_2, D_3) \leq d(D_1, D_2) + |p_1'p_1''| - (\cos 2\theta - \sin 2\theta)d(D_1, D_3).$$

Also, since $|p_1'p_1''| \leq 2\sin(\theta/2)r_1 \leq 2\sin(\theta/2)d(D_1, D_3)$, we have

$$d(D_2, D_3) \leq d(D_1, D_2) - (\cos 2\theta - \sin 2\theta - 2\sin(\theta/2))d(D_1, D_3).$$

Finally, since $d(D_2, D_3) < d(D_1, D_2)$, the induction hypothesis tells us that $\text{Yao}(\theta, \mathcal{D})$ contains a path from D_2 to D_3 whose length is at most $td(D_2, D_3)$. This means that the distance from D_1 to D_2 in $\text{Yao}(\theta, \mathcal{D})$ is at most

$$d(D_1, D_3) + td(D_2, D_3) \leq d(D_1, D_3) + t(d(D_1, D_2) - \frac{1}{t}d(D_1, D_3)) = td(D_1, D_2).$$

The value 0.228 is an upper bound on the values of θ such that $t > 0$. □

Corollary 7.6 *For any $\epsilon > 0$ and any set $\mathcal{D}$ of n disjoint disks, it is possible to compute a $(1+\epsilon)$-spanner of $\mathcal{D}$ that has $O(n)$ edges.*

Proof: The bound on the number of edges comes from the fact that each cone contains at most two edges, and the spanning ratio of $1+\epsilon$ comes from the fact that $\lim_{\theta \to 0} 1/(\cos 2\theta - \sin 2\theta - 2\sin(\theta/2)) = 1$. □

7.5 Quotient Graphs and Quotient Spanners

The main idea in the remainder of this chapter is the following: we show how to compute a set of points from each D_i such that the (standard) Delaunay graph of those points is *equivalent* to the Additively Weighted Delaunay graph. By choosing the appropriate equivalence relation as well as the appropriate point set, we can then show that the spanning ratio of the Additively Weighted Delaunay graph is bounded by the spanning ratio of the standard Delaunay graph. The reduction of one graph to another is done by means of a quotient:

Definition 7.7 *Let P_1 and P_2 be non-empty sets of points in the plane. The* distance *between P_1 and P_2, denoted by $|P_1P_2|$, is defined as the minimum $|p_1p_2|$ over all pairs of points such that $p_1 \in P_1$ and $p_2 \in P_2$.*

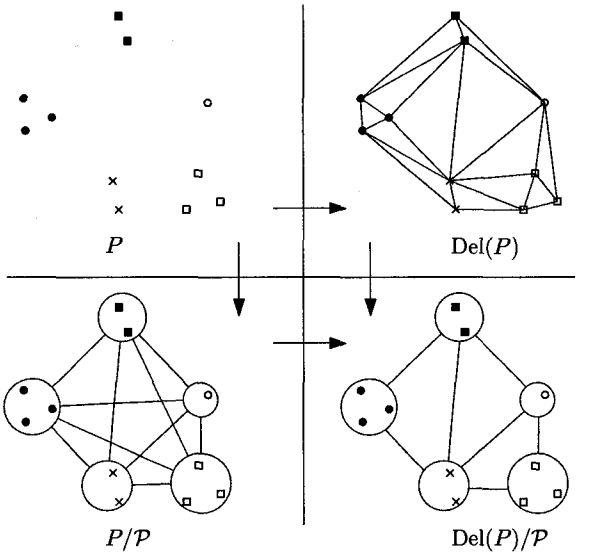


Figure 7.8: Illustration of Lemma 7.9.

Definition 7.8 *Let $G = (V, E)$ be a geometric graph and $\mathcal{V}$ be a partition of V. The* quotient graph *of G by $\mathcal{V}$, denoted $G/\mathcal{V}$, is the graph having $\mathcal{V}$ as vertices and there is an edge (U, W) (where U and W are in $\mathcal{V}$) if and only if there exists an edge $(u, w) \in E$ with $u \in U$ and $w \in W$. The weight of the edge (U, W) is equal to $|UW|$.*

If P is a (non-weighted) point set and $\mathcal{P}$ is a partition of P, then the notation $P/\mathcal{P}$ designates the quotient of the complete Euclidean graph on P by $\mathcal{P}$. If $\mathcal{S}$ is a set of pairwise disjoint sets of points in the plane such that $P \subseteq \bigcup \mathcal{S}$, then the notation $P/\mathcal{S}$ designates the quotient of the complete Euclidean graph on P by the partition of P induced by $\mathcal{S}$.

Lemma 7.9 *Let $G = (V, E)$ be a complete geometric graph, $\mathcal{V}$ be a partition of V and S be a t-spanner of G. Then $S/\mathcal{V}$ is a t-spanner of $G/\mathcal{V}$.*

Proof: Let (U, W) be an edge of $G/\mathcal{V}$ and (u, w) be an edge of G such that $|uw| = |UW|$. Since G is complete, the edge (u, w) is in G, and since S is a t-spanner of G, there is a path $\psi = u_1, \ldots, u_k$ in S such that $u_1 = u, u_k = w$ and the length of ψ is at most $t|uw|$. For each u_i of ψ, let $U_i \in \mathcal{V}$ be such that $u_i \in U_i$. Notice that it is possible that $U_i = U_{i+1}$ for some i. Let Ψ be the subsequence of $U = U_1, \ldots, U_k = W$ that consists in those U_i such that $i < k$ and $U_i \neq U_{i+1}$. By definition, the sequence

Ψ is a path in $S/\mathcal{V}$ and it consists of at most $k' \leq k$ nodes. The length of Ψ is at most

$$\sum_{i=1}^{k'-1} |U_i U_{i+1}| \leq \sum_{i=1}^{k-1} |u_i u_{i+1}| \leq t|uw| = t|UW|$$

which means that Ψ is a t-spanning path for (U, W) in $S/\mathcal{V}$. □

7.6 The Additively Weighted Delaunay Graph

Lee and Drysdale [57] studied a variant of the Voronoi diagram called the Additively Weighted Voronoi diagram, which is defined as follows: Let P be a weighted point set. The *Additively Weighted Voronoi diagram* of P is a partition of the plane into $|P|$ regions such that each region contains exactly the points in the plane having the same closest neighbor in P according to the additive distance. In other words, the Voronoi cell of a pair (p_i, r_i) contains the points p such that $d(p, p_i)$ is minimum over all other pairs in P. The *Additively Weighted Delaunay graph* (AW-Delaunay graph) is defined as the face-dual of the Additively Weighted Voronoi diagram.

Alternatively, if all r_i are positive and for all i, j, we have $|p_i p_j| \geq r_i + r_j$, then the pairs (p_i, r_i) can be seen as disks D_i of radius r_i centered at p_i and $d(p, D_i)$ is the minimum $|pq|$ over all $q \in D_i$. For a set $\mathcal{D}$ of disks in the plane, we denote the AW-Delaunay graph computed from $\mathcal{D}$ as $\mathrm{Del}(\mathcal{D})$. When no two disks intersect, the AW-Delaunay graph is a natural generalization of the Delaunay graph of a set of points. We say that two disks A and B *properly* intersect if $|A \cap B| > 1$.

Proposition 7.10 *Let $\mathcal{D}$ be a set of disjoint disks in the plane, and $A, B \in \mathcal{D}$. The edge (A, B) is in* $\mathrm{Del}(\mathcal{D})$ *if and only if there is a disk C that is tangent to both A and B and does not properly intersect any other disk in $\mathcal{D}$.*

Proof: Suppose (A, B) is in $\mathrm{Del}(\mathcal{D})$, and let c be a point on the boundary of the Voronoi cells of A and B and r be the distance from c to A. Since c is equidistant from A and B, it is also at distance r from B. This means that the disk C centered at c is tangent to both A and B. This disk cannot properly intersect any other disk of $\mathcal{D}$, since this would contradict the fact that c is in the Voronoi cells of A and B.

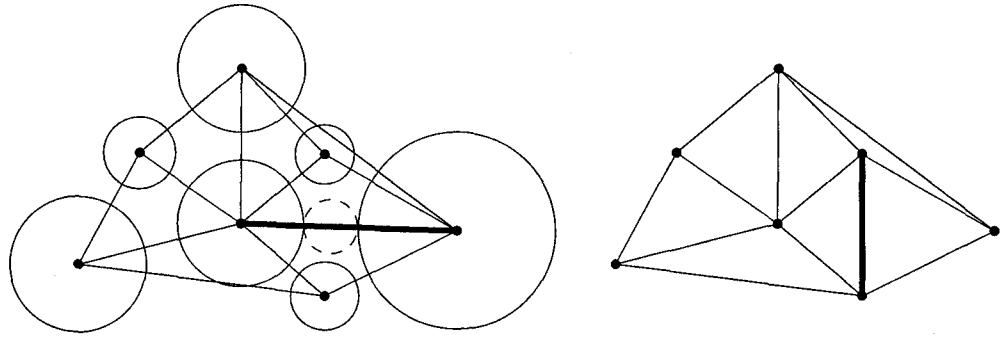


Figure 7.9: The Additively Weighted Delaunay graph compared with the Delaunay graph of the disks centers.

Similarly, if there is a disk that is tangent to both A and B but does not properly intersect any other disk of $\mathcal{D}$, then A and B are Voronoi neighbors. □

Note that the Additively Weighted Delaunay graph is not necessarily isomorphic to the Delaunay graph of the centers of the disks (see Figure 7.9). When all radii are equal, however, the two graphs coincide. We now show that if $\mathcal{D}$ is a set of disks in the plane, then $\text{Del}(\mathcal{D})$ is a spanner of $\mathcal{D}$. The intuition behind the proof is the following: we show the existence of a finite set of points P such that $K(P)/\mathcal{D}$ (where $K(P)$ is the complete graph with vertex set P) is isomorphic to the complete graph on $\mathcal{D}$ and $\text{Del}(P)/\mathcal{D}$ is a subgraph of $\text{Del}(\mathcal{D})$. Then, we use Lemma 7.9 to prove that $\text{Del}(P)/\mathcal{D}$ is a spanner of $\mathcal{D}$, which implies that $\text{Del}(\mathcal{D})$ is a spanner of $\mathcal{D}$.

Definition 7.11 *Let A, B be disjoint disks and S a set of points such that $A \cap S = \emptyset$ and $B \cap S = \emptyset$. A set of points R* represents *S with respect to A and B if for every disk F that is tangent to both A and B, we have $F \cap S \neq \emptyset \Rightarrow F \cap R \neq \emptyset$. If $\mathcal{D}$ is a set of disjoint disks, then a set of points $\mathcal{R}$* represents *$\mathcal{D}$ if for all $A, B, C \in \mathcal{D}$, there is a subset of $\mathcal{R}$ that represents C with respect to A and B.*

From here to the end of the proof of Lemma 7.15, unless stated otherwise, let

1. A, B be two disjoint disks in the plane having their center on the x-axis;
2. $D(y)$ be the disk that is tangent to both A and B and whose center has y-coordinate equal to y;

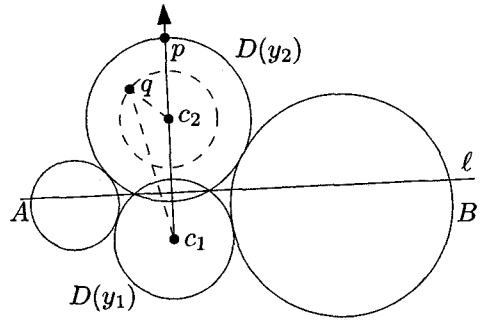


Figure 7.10: Illustration of the proof of Lemma 7.12.

3. $y(D)$ be the y-coordinate of the center of a disk D;

4. ℓ_1, ℓ_2 be the two lines that are outer-tangent to both A and B (respectively, from below and above);

5. y_1, y_2 be such that $y_1 < y_2$ and $D(y_1) \cap D(y_2) \neq \emptyset$;

6. ℓ be the line through the intersection points of the boundaries of $D(y_1)$ and $D(y_2)$ (if $D(y_1)$ and $D(y_2)$ are tangent, then ℓ is the unique line that is tangent to both $D(y_1)$ and $D(y_2)$);

7. $T(A, B)$ denote the region below ℓ_2, above ℓ_1 and between A and B; and

8. l^+ (l^-) be the closed half-plane above (below) a non-vertical line l.

Throughout this section, it is implicitly assumed that $D(\infty) = \ell_2^+$ and $D(-\infty) = \ell_1^-$.

Lemma 7.12 *Given $y_1 < y_2$ and $D(y_1) \cap D(y_2) \neq \emptyset$, we have $D(y_1) \cap \ell^+ \subset D(y_2) \cap \ell^+$ and $D(y_2) \cap \ell^- \subset D(y_1) \cap \ell^-$ (see Figure 7.10).*

Proof: Notice that either $D(y_1) \cap \ell^+ \subset D(y_2) \cap \ell^+$ or $D(y_2) \cap \ell^+ \subset D(y_1) \cap \ell^+$. Therefore, all we need to show is that $(D(y_2) \cap \ell^+) \setminus (D(y_1) \cap \ell^+)$ is not empty. Let c_1, c_2 be the respective centers of $D(y_1)$ and $D(y_2)$, and p be the intersection point of the infinite ray from c_1 through c_2 with the boundary of $D(y_1) \cup D(y_2)$.

We show by contradiction that p is not in $D(y_1)$. If that was the case, then $D(y_2)$ would be completely contained in $D(y_1)$. The reason for this is that there is

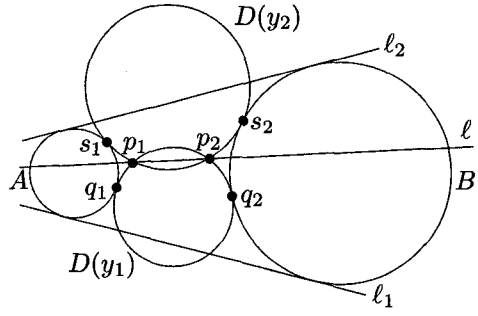


Figure 7.11: Illustration of the proof of Lemma 7.13.

no point of $D(y_2)$ that is farther from c_1 than p. Let q be a point of $D(y_2)$. Then $|qc_1| \leq |qc_2| + |c_2c_1| \leq |pc_2| + |c_2c_1| = |pc_1|$. But the fact that $D(y_2)$ is completely contained in $D(y_1)$ contradicts the fact that they are both tangent to A and B.

Therefore, since $p \in \ell^+$, we have $p \in (D(y_2) \cap \ell^+) \setminus (D(y_1) \cap \ell^+)$, which imply that $D(y_1) \cap \ell^+ \subset D(y_2) \cap \ell^+$. Similarly, $D(y_2) \cap \ell^- \subset D(y_1) \cap \ell^-$. □

Lemma 7.13 *Let p_1, p_2 be the intersection points of the boundaries of $D(y_1)$ and $D(y_2)$ (if $D(y_1)$ and $D(y_2)$ are tangent, then $p_1 = p_2$). Then p_1 and p_2 are in ℓ_2^- and in ℓ_1^+ (see Figure 7.11).*

Proof: Let q_1, q_2 be the tangency points of $D(y_1)$ with A and B and s_1, s_2 be the tangency points of $D(y_2)$ with A and B. By Lemma 7.12, q_1, q_2 are below ℓ and s_1, s_2 are above ℓ. Since ℓ is above q_1 and q_2, which are in turn above ℓ_1, it follows that p_1 and p_2 are above ℓ_1. By a symmetric argument, p_1 and p_2 are below ℓ_2. □

Lemma 7.14 *The following are true:*

1. *For all $p \in \ell_2^+$, there exists a line $y = y_0$ such that for all disk E that is tangent to both A and B, if the center of E is above y_0 then $p \in E$.*

2. *For all $p \in \ell_1^-$, there exists a line $y = y_1$ such that for all disk E that is tangent to both A and B, if the center of E is below y_1 then $p \in E$.*

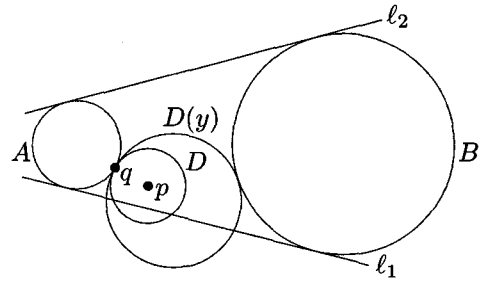


Figure 7.12: Illustration of the proof of Lemma 7.14 (3) (first part).

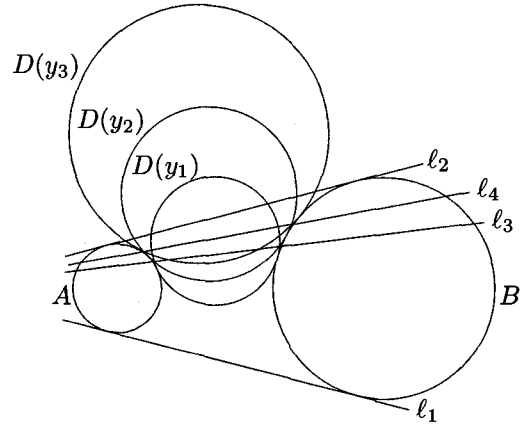


Figure 7.13: Illustration of the proof of Lemma 7.14 (3) (second part).

3. *For all p in $T(A, B)$, there exists two lines $y = y_0$ and $y = y_1$ such that for all disk E that is tangent to both A and B, $p \in E$ if and only if the center of E is between y_0 and y_1.*

Proof: For (1), the existence of y_0 is guaranteed by the fact that $\lim_{y\to\infty} D(y) = \ell_2^+$. Now, let y_0 be such that $p \in D(y_0)$ and $y' > y_0$. Let $L(y_0)$ and $L(y')$ be the lunes respectively defined by the intersection of $D(y_0)$ and $D(y')$ with the half-plane above ℓ_2. By Lemma 7.13, the two points where the boundaries of $D(y_0)$ and $D(y')$ intersect are below ℓ_2. Therefore, we have either $L(y_0) \subset L(y')$ or $L(y') \subset L(y_0)$. But since $y' > y_0$, by Lemma 7.12 we have $L(y_0) \subset L(y')$ and therefore $p \in L(y')$. The proof of (2) is symmetric.

For (3), the existence is easy to show. Without loss of generality, assume $d(p, A) \leq d(p, B)$. Let D be the disk centered at p that is tangent to A and let q be the tangency point of A and D see Figure 7.12. Since $q \in T(A, B)$, there exists y such that $D(y) \cap A = q$. Since $D \subseteq D(y)$, there exists a disk that is tangent to both A

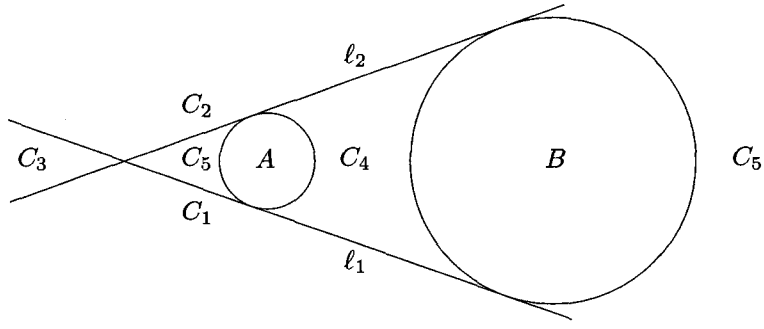


Figure 7.14: The five regions for Lemma 7.15.

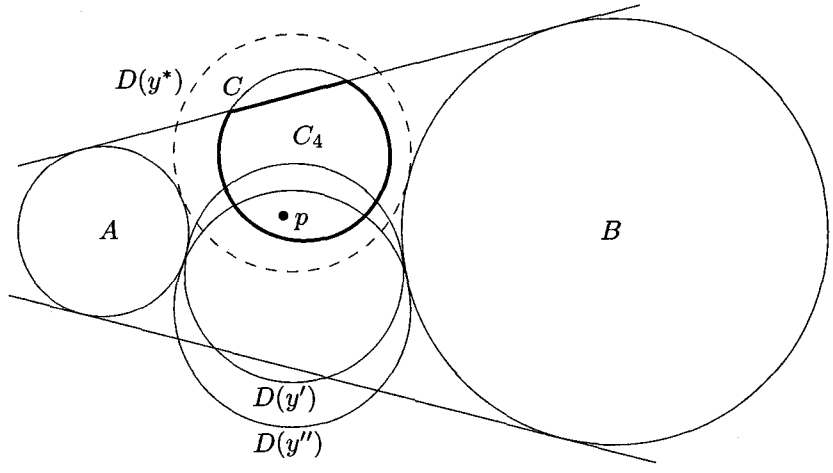


Figure 7.15: Case C_4 of the proof of Lemma 7.15.

and B and contains p.

We now show that $y_1 < y_2 < y_3$ implies $D(y_1) \cap D(y_3) \subseteq D(y_2)$ (see Figure 7.13). Let ℓ_3 be the line through the intersection points of the boundaries of $D(y_1)$ and $D(y_2)$ and let ℓ_4 be the line through the intersection points of the boundaries of $D(y_2)$ and $D(y_3)$. Let $p \in D(y_1) \cap D(y_3)$. Since ℓ_4 is above ℓ_3 in $D(y_1) \cap D(y_3)$, p is either above ℓ_3, below ℓ_4 or both. If $p \in \ell_3^+$, then since $y_1 < y_2$, by Lemma 7.12 we have that $D(y_1) \cap \ell_3^+ \subseteq D(y_2) \cap \ell_3^+$ and $p \in D(y_1) \cap D(y_2)$. Similarly, if $p \in \ell_4^-$, then since $y_2 < y_3$, by Lemma 7.12 we have that $D(y_3) \cap \ell_4^- \subseteq D(y_2) \cap \ell_4^-$ and $p \in D(y_3) \cap D(y_2)$. In either case, $p \in D(y_2)$, which completes the proof. □

Lemma 7.15 *Let C be a disk that is disjoint of both A and B. There exists a set of at most six points that represents C with respect to A and B.*

Proof: Let

$$\begin{aligned} C_1 &:= (C \cap \ell_1^-) \setminus \ell_2^+ \\ C_2 &:= (C \cap \ell_2^+) \setminus \ell_1^- \\ C_3 &:= C \cap \ell_1^- \cap \ell_2^+ \\ C_4 &:= C \cap T(A,B) \\ C_5 &:= (C \cap \ell_1^+ \cap \ell_2^-) \setminus T(A,B) \end{aligned}$$

These five regions partition the disk C (see Figure 7.14). We show that for each region, there is a finite set of points that represents it. The cardinality of the union of the sets is no more than six.

If $C_1 \neq \emptyset$, then let y_0 be the minimum y such that $D(y)$ intersects C_1. Let $p_1 \in C_1 \cap D(y_0)$. By definition of y_0, for any disk E that is tangent to both A and B and intersects C_1, we have $y(E) \geq y_0$, and by Lemma 7.14, we have $p_1 \in E$.

Similarly, if $C_2 \neq \emptyset$, then let y_1 be the maximum y such that $D(y)$ intersects C_2. Let $p_2 \in C_2 \cap D(y_1)$. By definition of y_1, for any disk E that is tangent to both A and B and intersects C_2, we have $y(E) \leq y_1$, and by Lemma 7.14, we have $p_2 \in E$.

If $C_3 \neq \emptyset$, then let y_0 be the minimum $y > 0$ such that $D(y)$ intersects C_3 and y_1 as the maximum $y < 0$ such that $D(y)$ intersects C_3. Let $p_3 \in C_3 \cap D(y_0)$ and $p_4 \in C_3 \cap D(y_1)$. By definition of y_0, for any disk E with $y(E) > 0$ that is tangent to both A and B and intersects C_3, we have $y(E) \geq y_0$, and by Lemma 7.14, we have $p_3 \in E$. The same reasoning applies to p_4 when $y(E) < 0$.

If $C_4 \neq \emptyset$, then let y_0 be the minimum y such that $D(y)$ intersects C_4 and y_1 as the maximum y such that $D(y)$ intersects C_4. Let $p_5 \in C_4 \cap D(y_0)$ and $p_6 \in C_4 \cap D(y_1)$. Let y^* be such that $C \subseteq D(y^*)$ (see Figure 7.15). Let E be a disk that is tangent to both A and B and intersects C_4. We show that $y(E) \leq y^* \implies p_5 \in E$ (and similarly, $y(E) \geq y^* \implies p_6 \in E$). It is sufficient to show that $y'' < y' < y^* \implies C \cap D(y'') \subset C \cap D(y')$. Let $p \in D(y'') \cap C$. By Lemma 7.14, $\exists y_0(p), y_1(p)$ such that $\forall$ disk E tangent to both A and B, we have $y_0(p) \leq y(E) \leq y_1(p) \Leftrightarrow p \in E$. Therefore, the following hold:

$$\begin{aligned} y_0(p) \leq y^* &\leq y_1(p) \\ y_0(p) \leq y'' &\leq y_1(p) \end{aligned}$$

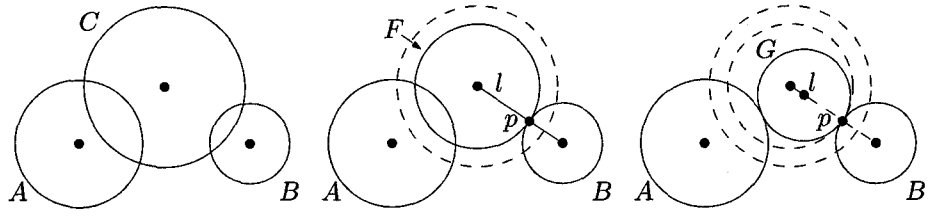


Figure 7.16: Proof of Lemma 7.17.

But since $y'' < y' < y^*$, we have $y'' < y' < y^*$, which imply that $p \in C \cap D(y')$.

Finally, since $C_5 \cap E = \emptyset$ for any disk E that is tangent to both A and B, there is no need to select representative points for C_5. □

Careful analysis of the proof of Lemma 7.15 allows us to observe that in fact, only two points are necessary to represent a disk C with respect to two other disks A and B. First, note that $C_4 \neq \emptyset \implies C_3 = \emptyset$ and $C_3 \neq \emptyset \implies C_4 = \emptyset$. This reduces to four the number of points that are necessary. Also, if $C_1 \neq \emptyset$ and $C_4 \neq \emptyset$, then p_6 is on ℓ_2 and is not required since any disk that contains it also intersects C_1 and therefore contains p_1. Similarly, if $C_2 \neq \emptyset$ and $C_4 \neq \emptyset$, then p_5 is not required since any disk that contains it also intersects C_2 and therefore contains p_2. Therefore, if $C_4 \neq \emptyset$, then the number of points that are necessary is at most two. A similar argument applies to the case where $C_3 \neq \emptyset$. Finally, if both C_3 and C_4 are empty, then only p_1 and p_2 may be required. Therefore, we have the following corollary:

Corollary 7.16 *Let $\mathcal{D}$ be a set of n disjoint disks. There exists a set of at most $2\binom{n}{3}$ points that represents $\mathcal{D}$.*

Lemma 7.17 *Let A and B be two disjoint disks and C be a disk intersecting both of them. Then there exists a disk G inside C that is tangent to both A and B.*

Proof: We show how to construct G. Let a, b, c and r_A, r_B, r_C respectively be the centers and radii of A, B and C. Without loss of generality, assume $|ac| - r_C \leq |bc| - r_B$. Let F be the disk centered at c and having radius $r_F = |bc| - r_B$ (see Figure 7.16). The disk F is tangent to B. If F is also tangent to A, then let $G = F$ and we are done. Otherwise, F is properly intersecting A. In that case, let p be the tangency

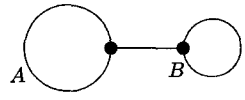


Figure 7.17: The distance points of A and B.

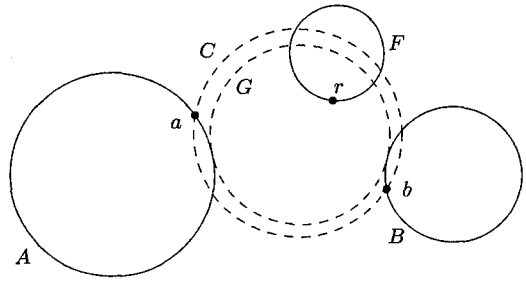


Figure 7.18: Illustration of the proof of Theorem 7.19.

point of F and B, l be the line through b and c, and G be the disk through p having its center on l and tangent to A. The result follows from the fact that G is tangent to B and inside C. □

Definition 7.18 *Let A and B be two disks in the plane. The* distance points *of A and B are the two ends of the shortest line segment between A and B (see Figure 7.17). If $\mathcal{D}$ is a set of disjoint disks, then the set of* distance points *of $\mathcal{D}$ is the set containing the distance points of every pair of disks in $\mathcal{D}$.*

Theorem 7.19 *Let $\mathcal{D}$ be a set of n disjoint disks. Then* Del($\mathcal{D}$) *is a* SP-DT*-spanner of $\mathcal{D}$, where* SP-DT*is the spanning ratio of the Delaunay triangulation of a set of points.*

Proof: By Corollary 7.16, let R be a set of size at most $2\binom{n}{3}$ that represents $\mathcal{D}$, let S be the set of distance points of $\mathcal{D}$, and let $P = R \cup S$. Since Del(P) is a SP-DT-spanner of P, by Lemma 7.9, we have Del(P)/$\mathcal{D}$ is a SP-DT-spanner of $K(P)/\mathcal{D}$, where $K(P)$ is the complete graph with vertex set P. Since P contains the distance points of $\mathcal{D}$, $K(P)/\mathcal{D}$ is isomorphic to the complete graph defined on $\mathcal{D}$. We show that each edge (A, B) of Del(P)/$\mathcal{D}$ is in Del($\mathcal{D}$). Let (A, B) be an edge of Del(P)/$\mathcal{D}$. This means that in P, there are two points a and b with $a \in A, b \in B$ such that

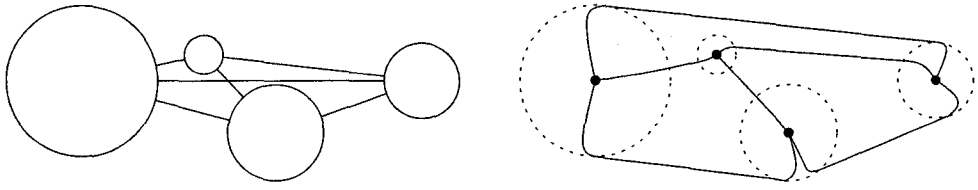


Figure 7.19: Even if the embedding of the AW-Delaunay graph that consists of straight line segments between the centers of the disks is not necessarily a plane graph, it is planar.

there is an empty circle C through a and b. By Lemma 7.17, C contains a disk G that is tangent to both A and B. The disk G is a witness of the presence of the edge (A, B) in $\mathrm{Del}(\mathcal{D})$. If that was not the case, this would mean that there exists a disk $F \in \mathcal{D}$ such that $G \cap F \neq \emptyset$. By definition of R, this implies that $G \cap R \neq \emptyset$ and thus $C \cap P \neq \emptyset$, which contradicts the fact that C is an empty circle. Therefore, the edge (A, B) is in $\mathrm{Del}(\mathcal{D})$. Since $\mathrm{Del}(P)/\mathcal{D}$ is a SP-DT-spanner of $\mathcal{D}$ and a subgraph of $\mathrm{Del}(\mathcal{D})$, we conclude that $\mathrm{Del}(\mathcal{D})$ is a SP-DT-spanner of $\mathcal{D}$. □

7.7 Computing a Plane Embedding

Note that the embedding of the AW-Delaunay graph that consists of straight line segments between the centers of the disks is not necessarily a plane graph (see Figure 7.19). However, the Voronoi diagram of a set of disks $\mathcal{D}$, denoted $\mathrm{Vor}(\mathcal{D})$, is planar [67]. Since $\mathrm{Del}(\mathcal{D})$ is the face-dual of $\mathrm{Vor}(\mathcal{D})$, it is also planar. An important characteristic of the Delaunay graph of a set of points regarded as a spanner is that it is a plane graph. Therefore, a natural question is whether $\mathrm{Del}(\mathcal{D})$ has a plane embedding that is also a spanner.

The proof of Theorem 7.19 suggests the existence of an algorithm allowing to compute such an embedding: compute the Delaunay triangulation of the set P that contains the distance points and the representative of $\mathcal{D}$. The graph $\mathrm{Del}(P)$ can be regarded as a multigraph whose vertex set is $\mathcal{D}$. Then, for each pair of disks that share one or more edges, just keep the shortest of those edges. This simple algorithm allows us to compute a plane embedding of $\mathrm{Del}(\mathcal{D})$ that is also a spanner of $\mathcal{D}$. However, its

running time is $O(n^3 \log n)$.

On the other hand, it is also possible to compute in time $O(n \log n)$ a plane spanner of $\mathcal{D}$ whose spanning ratio is SP-DT^2 (i.e. the square of the spanning ratio of the Delaunay graph of a set of points). Here is how to do this: First, compute Del($\mathcal{D}$). Then, let P be the set of distance points of all pairs of disks that share an edge in Del($\mathcal{D}$). Compute Del(P). Since P has size $O(n)$, this can be done in time $O(n \log n)$. Also, Del(P) is a plane graph. As in the above paragraph, the graph Del(P) can be regarded as a multigraph whose vertex set is $\mathcal{D}$. Again, for each pair of disks that share one or more edges, just keep the shortest of those edges. All that remains to explain is why the resulting graph is a (SP-DT^2)-spanner of $\mathcal{D}$. Let $D_1, D_2 \in \mathcal{D}$. The straight line embedding of Del($\mathcal{D}$) contains a SP-DT-spanning path between D_1 and D_2. The endpoints of the edges of that path are the distance points between the disks. In Del(P), each of those edges is approximated within a factor of SP-DT, leading to a spanning ratio of SP-DT^2. Therefore, we showed the following:

Theorem 7.20 *Let $\mathcal{D}$ be a set of n disjoint disks and* SP-DT*be the spanning ratio of the Delaunay triangulation of a set of points. In time $O(n^3 \log n)$, it is possible to compute a plane* SP-DT*-spanner of $\mathcal{D}$, and in time $O(n \log n)$, it is possible to compute a plane* SP-DT^2*-spanner of $\mathcal{D}$.*

Whether or not it is possible to compute a plane embedding of Del($\mathcal{D}$) that is also a SP-DT-spanner of $\mathcal{D}$ in time $O(n \log n)$ remains a open question.

7.8 Conclusion

In this chapter, we showed how, given a weighted point set where weights are positive and $|p_i p_j| \geq r_i + r_j$ for all $i \neq j$, it is possible to compute a $(1+\epsilon)$-spanner of that point set that has a linear number of edges. We also showed that the Additively Weighted Delaunay graph is a t-spanner of an additively weighted point set in the same case. The constant t is the same as for the Delaunay triangulation of a point set (the best current value is 2.42 [48]). We could not see how the Well-Separated Pair Decomposition (WSPD) can be adapted to solve that problem. The first difficulty resides in the fact that it is not even clear that, given a weighted point set, a WSPD

of that point set always exists. Other obvious open questions are whether our results still hold when some weights are negative or $|p_i p_j| < r_i + r_j$ for some $i \neq j$. Also, we did not verify whether our variant of the Yao graph can be computed in time $O(n \log n)$. Finally, another problem that could be explored is whether it is possible to compute t-spanners for multiplicatively weighted point sets.

Chapter 8

Conclusions

In this thesis, we explored algorithmic aspects of ad hoc wireless networks. The problems we addressed combined geometric and combinatorial difficulties. Even if the literature on the subject is incredibly vast, there are still many problems to be explored. We conclude with a summary of open problems that arose from our research.

1. Find a dominating set algorithm for unit disk graphs that is local, location-oblivious and has a constant deterministic performance ratio.

2. Close the gap between the lower bound of four and the upper bound of five for the performance ratio of the sequential coloring algorithm for unit disk graphs.

3. Determine whether or not the Half-Space Proximal has a constant spanning ratio.

4. Compute $t(k)$ and $t'(k)$ for values of k greater than 4.

5. Compute a spanner of complete k-partite graph that is planar.

6. For complete k-partite graphs, either give an algorithm that computes a $(3+\epsilon)$-spanner with $O(n)$ edges or give a lower bound of $(5-\epsilon)$ for the problem.

7. Compute a spanner of an additively weighted point set where the weights are negative.

8. Compute a spanner of a multiplicatively weighted point set.

9. Compute a plane embedding of the Additively Weighted Delaunay graph that has constant spanning ratio in time $O(n \log n)$.

Bibliography

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A survey on sensor networks. *Communications Magazine, IEEE*, **40(8)**:102–114, 2002.

[2] J. N. Al-Karaki and A. E. Kamal, Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, **11(6)**:6–28, 2004.

[3] J. Alber, M. R. Fellows, and R. Niedermeier, Polynomial-time data reduction for dominating set. *J. ACM*, **51(3)**:363–384, 2004.

[4] I. Althöfer, G. Das, D. P. Dobkin, D. Joseph, and J. Soares, On sparse spanners of weighted graphs. *Discrete & Computational Geometry*, **9**:81–100, 1993.

[5] K. M. Alzoubi, X.-Y. Li, Y. Wang, P.-J. Wan, and O. Frieder, Geometric spanners for wireless ad hoc networks. *IEEE Trans. Parallel Distrib. Syst.*, **14(4)**:408–421, 2003.

[6] K. M. Alzoubi, P.-J. Wan, and O. Frieder, Message-optimal connected dominating sets in mobile ad hoc networks. In *MobiHoc '02: Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pp. 157–164, ACM Press, New York, NY, USA, 2002.

[7] F. Aurenhammer, Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, **16(1)**:78–96, 1987.

[8] P. Bose, P. Carmi, and M. Couture, Spanners of additively weighted point sets. In *SWAT '08: Proceedings of the 11th Scandinavian Workshop on Algorithm Theory*, Gothenburg, Sweden, 2008.

[9] P. Bose, P. Carmi, M. Couture, A. Maheshwari, P. Morin, and M. Smid, Spanners of complete k-partite geometric graphs. In *LATIN '08: Proceedings of the 8th Latin American Theoretical Informatics Symposium*, Rio de Janeiro, Brazil, 2008.

[10] P. Bose, P. Carmi, M. Couture, A. Maheshwari, M. Smid, and N. Zeh, Geometric spanners with small chromatic number. In *WAOA '07: Proceedings of the 5th Workshop on Approximation and Online Algorithms*, Eilat, Israel, 2007.

[11] P. Bose, P. Carmi, M. Couture, M. Smid, and D. Xu, On a family of strong geometric spanners that admit local routing strategies. In *WADS '07: Proceedings of the Workshop on Algorithms and Data Structures*, Halifax, Canada, 2007.

[12] P. Bose, L. Devroye, W. Evans, and D. Kirkpatrick, On the spanning ratio of Gabriel graphs and β-skeletons. *SIAM Journal on Discrete Mathematics*, **20(2)**:412–427, 2006.

[13] P. Bose and J. M. Keil, On the stretch factor of the constrained delaunay triangulation. In *ISVD '06: Proceedings of the 3rd International Symposium on Voronoi Diagrams in Science and Engineering*, pp. 25–31, IEEE Computer Society, Washington, DC, USA, 2006.

[14] P. Bose, A. Maheshwari, G. Narasimhan, M. Smid, and N. Zeh, Approximating geometric bottleneck shortest paths. *Comput. Geom. Theory Appl.*, **29(3)**:233–249, 2004.

[15] P. Bose and P. Morin, Online routing in triangulations. *SIAM Journal on Computing*, **33(4)**:937–951, 2004.

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

[17] P. B. Callahan and S. R. Kosaraju, Faster algorithms for some geometric graph problems in higher dimensions. In *Proceedings of the 4th ACM-SIAM Symposium on Discrete Algorithms*, pp. 291–300, 1993.

[18] P. B. Callahan and S. R. Kosaraju, A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, **42**:67–90, 1995.

[19] M. Cardei, X. Cheng, X. Cheng, and D.-Z. Du, Connected domination in multihop ad hoc wireless networks. In *Proc. the 6th International Conference on Computer Science and Informatics (CS&I'2002)*, Durham, NC, USA, 2002.

[20] E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, H. Tejeda, and J. Urrutia, Half-space proximal: A new local test for extracting a bounded dilation spanner. In Springer-Verlag, ed., *OPODIS' 05: Proceddings of the 9th International Conference on Principles of Distributed Systems*, LNCS, 2005.

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

[22] M. Couture, M. Barbeau, P. Bose, P. Carmi, and E. Kranakis, Location oblivious distributed unit disk graph coloring. In *SIROCCO '07: Proceedings of the 14th International Colloquium on Structural Information and Communication Complexity*, Castiglioncello (LI), Italy, 2007.

[23] M. COUTURE, M. BARBEAU, P. BOSE, AND E. KRANAKIS, Incremental construction of k-dominating sets in wireless sensor networks. In A. A. SHVARTSMAN, ed., *OPODIS*, vol. 4305 of *Lecture Notes in Computer Science*, pp. 202–214, Springer, 2006.

[24] M. COUTURE, M. BARBEAU, P. BOSE, AND E. KRANAKIS, Incremental construction of k-dominating sets in wireless sensor networks. *International Journal of Ad Hoc & Sensor Wireless Networks*, **5(1)**:47–68, 2008.

[25] F. DAI AND J. WU, Distributed dominant pruning in ad hoc networks. In *Proceedings of the International Conference on Communications (ICC)*, Anchorage, AK, 2003.

[26] F. DAI AND J. WU, On constructing k-connected k-dominating set in wireless networks. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium*, p. 81.1, IEEE Computer Society, Washington, DC, USA, 2005.

[27] B. DAS AND V. BHARGHAVAN, Routing in ad-hoc networks using minimum connected dominating sets. In *Proceedings of the International Conference on Communications (ICC)*, pp. 376–380, 1997.

[28] M. DE BERG, M. VAN KREVELD, M. OVERMARS, AND O. SCHWARZKOPF, *Computational Geometry: Algorithms and Applications*. Springer, 1997.

[29] D. P. DOBKIN, S. J. FRIEDMAN, AND K. J. SUPOWIT, Delaunay graphs are almost as good as complete graphs. *Discrete Comput. Geom.*, **5(4)**:399–407, 1990.

[30] 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.

[31] U. FEIGE, M. M. HALLDORSSON, G. KORTSARZ, AND A. SRINIVASAN, Approximating the domatic number. *SIAM Journal on Computing*, **32(1)**:172–195, 2003.

[32] S. FORTUNE, A sweepline algorithm for voronoi diagrams. *Algorithmica*, **2**:153–174, 1987.

[33] M. FURER AND S. P. KASIVISWANATHAN, Spanners for geometric intersection graphs. In *CCCG'07: Proceddings of the 19th Canadian Conference on Computational Geometry*, 2007.

[34] J. GAO, L. GUIBAS, J. HERSHBERGER, L. ZHANG, AND A. ZHU, Discrete mobile centers. In *SCG '01: Proceedings of the seventeenth annual symposium on Computational geometry*, pp. 188–196, ACM Press, New York, NY, USA, 2001.

[35] J. GAO, L. J. GUIBAS, J. HERSHBERGER, L. ZHANG, AND A. ZHU, Geometric spanner for routing in mobile networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pp. 45–55, ACM Press, New York, NY, USA, 2001.

[36] A. GRÄF, M. STUMPF, AND G. WEISSENFELS, On coloring unit disk graphs. *Algorithmica*, **20(3)**:277–293, 1998.

[37] H. GRÖTZSCH, Ein Dreifarbensatz für dreiecksfreie Netze auf der Kugel. *Wiss. Z. Martin-Luther-Univ. Halle-Wittenberg Math.-Natur. Reihe*, **8**:109–120, 1959.

[38] M. GRUNEWALD, T. LUKOVSZKI, C. SCHINDELHAUER, AND K. VOLBERT, Distributed maintenance of resource efficient wireless network topologies (distinguished paper). In *Euro-Par '02: Proceedings of the 8th International Euro-Par Conference on Parallel Processing*, pp. 935–946, Springer-Verlag, London, UK, 2002.

[39] J. GUDMUNDSSON, C. LEVCOPOULOS, G. NARASIMHAN, AND M. SMID, Approximate distance oracles for geometric graphs. In *Proceedings of the 13th ACM-SIAM Symposium on Discrete Algorithms*, pp. 828–837, 2002.

[40] J. GUDMUNDSSON AND M. SMID, On spanners of geometric graphs. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory*, vol. 4059 of *Lecture Notes in Computer Science*, pp. 388–399, Springer-Verlag, Berlin, 2006.

[41] S. GUHA AND S. KHULLER, Approximation algorithms for connected dominating sets. In *ESA '96: Proceedings of the Fourth Annual European Symposium on Algorithms*, pp. 179–193, Springer-Verlag, London, UK, 1996.

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

[43] J. C. HANSEN AND E. SCHMUTZ, The expected size of the rule k dominating set. *CoRR*, **cs.DM/0408067**, 2004.

[44] H. B. HUNT, M. V. MARATHE, V. RADHAKRISHNAN, S. S. RAVI, D. J. ROSENKRANTZ, AND R. E. STEARNS, Nc-approximation schemes for np- and pspace-hard problems for geometric graphs. *J. Algorithms*, **26(2)**:238–274, 1998.

[45] D. S. JOHNSON, Approximation algorithms for combinatorial problems. In *STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing*, pp. 38–49, ACM Press, New York, NY, USA, 1973.

[46] M. I. KARAVELAS AND M. YVINEC, Dynamic additively weighted voronoi diagrams in 2d. In *ESA '02: Proceedings of the 10th Annual European Symposium on Algorithms*, pp. 586–598, Springer-Verlag, London, UK, 2002.

[47] 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.

[48] J. M. KEIL AND C. A. GUTWIN, Classes of graphs which approximate the complete Euclidean graph. *Discrete Comput. Geom.*, **7(1)**:13–28, 1992.

[49] D.-S. KIM, Y. CHO, AND D. KIM, Euclidean Voronoi diagram of 3d balls and its computation via tracing edges. *Computer-Aided Design*, **37(13)**:1412–1424, 2005.

[50] R. KLASING AND C. LAFOREST, Hardness results and approximation algorithms of k-tuple domination in graphs. *Inf. Process. Lett.*, **89(2)**:75–83, 2004.

[51] V. KLEE, On the complexity of d-dimensional voronoi diagrams. *Archiv der Mathematik*, **34(1)**:75–80, 1980.

[52] F. KUHN, The Price of Locality: Exploring the Complexity of Distributed Coordination Primitives. In *PhD Thesis, ETH Zurich, Diss. ETH No. 16213*, 2005.

[53] F. KUHN, T. MOSCIBRODA, AND R. WATTENHOFER, What cannot be computed locally! In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pp. 300–309, ACM Press, New York, NY, USA, 2004.

[54] F. KUHN, T. MOSCIBRODA, AND R. WATTENHOFER, On the locality of bounded growth. In *PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pp. 60–68, ACM Press, New York, NY, USA, 2005.

[55] F. KUHN, T. MOSCIBRODA, AND R. WATTENHOFER, Fault-tolerant clustering in ad hoc and sensor networks. In *ICDCS '06: Proceedings of the 26th International Conference on Distributed Computing Systems*, 2006.

[56] S. KUTTEN AND D. PELEG, Fast distributed construction of small k-dominating sets and applications. *J. Algorithms*, **28(1)**:40–66, 1998.

[57] D. T. LEE AND R. L. DRYSDALE, Generalization of voronoi diagrams in the plane. *SIAM Journal on Computing*, **10(1)**:73–87, 1981.

[58] X.-Y. LI, P.-J. WAN, AND Y. WANG, Power efficient and sparse spanner for wireless ad hoc networks. In *ICCCN '01: Proceedings of the IEEE International Conference on Computer Communications and Networks*, pp. 564–567, 2001.

[59] Y. Li, M. T. Thai, F. Wang, C.-W. Yi, P. Wan, and D.-Z. Du, On greedy construction of connected dominating sets in wireless networks. Tech. Rep. 04-048, University of Minnesota - Computer Science and Engineering, 2004.

[60] Y. Li, M. T. Thai, F. Wang, C.-W. Yi, P. Wan, and D.-Z. Du, On greedy construction of connected dominating sets in wireless networks. In *Wireless Communications and Mobile Computing (WCMC)*, vol. 5, pp. 927–932, 2005.

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

[62] S. Masuyama, T. Ibaraki, and T. Hasegawa, The computational complexity of the m-center problems on the plane. *IEICE TRANSACTIONS*, **E64-E(2)**:57–64, 1981.

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

[64] T. Moscibroda and R. Wattenhofer, Maximizing the lifetime of dominating sets. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 12*, p. 242.2, IEEE Computer Society, Washington, DC, USA, 2005.

[65] G. Narasimhan and M. Smid, *Geometric Spanner Networks.* Cambridge University Press, New York, NY, USA, 2007.

[66] T. Nieberg and J. L. Hurink, A PTAS for the minimum dominating set problem in unit disk graphs. Memorandum 1732, Enschede, 2004.

[67] A. Okabe, B. Boots, and K. Sugihara, *Spatial tessellations: concepts and applications of Voronoi diagrams.* John Wiley & Sons, Inc., New York, NY, USA, 2nd edn., 2000.

[68] K. Pahlavan and A. H. Levesque, *Wireless information networks.* Wiley-Interscience, New York, NY, USA, 1995.

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

[70] M. D. Penrose, The longest edge of the random minimal spanning tree. *The Annals of Applied Probability*, **7(2)**:340–361, 1997.

[71] M. D. Penrose, On k-connectivity for a geometric random graph. *Random Struct. Algorithms*, **15(2)**:145–164, 1999.

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

[73] B. RAMAN AND K. CHEBROLU, Revisiting mac design for an 802.11-based mesh network. In *SIGCOMM HotNetsIII Workshop*, 2004.

[74] B. RAMAN AND K. CHEBROLU, Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pp. 156–169, ACM Press, New York, NY, USA, 2005.

[75] J. RUPPERT AND R. SEIDEL, Approximating the d-dimensional complete Euclidean graph. In *CCCG'91: Proceddings of the 3rd Canadian Conference on Computational Geometry*, pp. 207–210, 1991.

[76] J. S. SALOWE, Constructing multidimensional spanner graphs. *International Journal of Computational Geometry & Applications*, **1**:99–107, 1991.

[77] C. SCHINDELHAUER, K. VOLBERT, AND M. ZIEGLER, Spanners, weak spanners, and power spanners for wireless networks. In R. FLEISCHER AND G. TRIPPEN, eds., *Proc. of 15th Annual International Symposium on Algorithms and Computation (ISAAC'04)*, vol. 3341 of *Springer Lecture Notes in Computer Science LNCS*, pp. 805–821, Springer Verlag, 2004.

[78] C. SCHINDELHAUER, K. VOLBERT, AND M. ZIEGLER, Geometric spanners with applications in wireless networks. *Comput. Geom. Theory Appl.*, **36(3)**:197–214, 2007.

[79] I. STOJMENOVIC, M. SEDDIGH, AND J. ZUNIC, Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks. *IEEE Trans. Parallel Distrib. Syst.*, **13(1)**:14–25, 2002.

[80] C. THOMASSEN, A short list color proof of grotzsch's theorem. *Journal of Combinatorial Theory B*, **88**:189–192, 2003.

[81] Y.-T. TSAI, Y.-L. LIN, AND F.-R. HSU, The on-line first-fit algorithm for radio frequency assignment problems. *Inf. Process. Lett.*, **84(4)**:195–199, 2002.

[82] J. URRUTIA, Local solutions for global problems in wireless networks. *J. of Discrete Algorithms*, **5(3)**:395–407, 2007.

[83] P. M. VAIDYA, A sparse graph almost as good as the complete graph on points in K dimensions. *Discrete & Computational Geometry*, **6**:369–381, 1991.

[84] P.-J. WAN, K. M. ALZOUBI, AND O. FRIEDER, Distributed construction of connected dominating set in wireless ad hoc networks. *Mob. Netw. Appl.*, **9(2)**:141–149, 2004.

[85] H.-M. WILL, Fast and efficient computation of additively weighted voronoi cells for applications in molecular biology. In *SWAT '98: Proceedings of the 6th Scandinavian Workshop on Algorithm Theory*, pp. 310–321, Springer-Verlag, London, UK, 1998.

[86] J. WU AND F. DAI, A generic distributed broadcast scheme in ad hoc wireless networks. *IEEE Trans. Comput.*, **53(10)**:1343–1354, 2004.

[87] J. WU AND H. LI, On calculating connected dominating set for efficient routing in ad hoc wireless networks. In *DIALM '99: Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, pp. 7–14, ACM Press, New York, NY, USA, 1999.

[88] J. WU AND H. LI, A dominating-set-based routing scheme in ad hoc wireless networks. *Telecommunication Systems*, **18(1–3)**:13–36, 2001.

[89] W. WU, H. DU, X. JIA, Y. LI, C.-H. HUANG, AND D.-Z. DU, Minimum connected dominating sets and maximal independent sets in unit disk graphs. Tech. Rep. 04-047, Department of Computer Science and Engineering, University of Minnesota, 2004.

[90] S. YANG, F. DAI, M. CARDEI, AND J. WU, On multiple point coverage in wireless sensor networks. In *MASS '05: Proceedings of the 2nd International Conference on Mobile Adhoc and Sensor Systems Conference*, 2005.

[91] A. C.-C. YAO, On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, **11(4)**:721–736, 1982.