Spanners of Complete k-Partite Geometric Graphs

Prosenjit Bose* Paz Carmi* Mathieu Couture* Anil Maheshwari* Pat Morin*
Michiel Smid*

September 25, 2007

Abstract

We address the following problem: Given a complete k-partite geometric graph K whose vertex set is a set of n points in $\mathbb{R}^d$, compute a spanner of K that has a "small" stretch factor and "few" edges. 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 there exist complete k-partite geometric graphs K such that every subgraph of K with a subquadratic number of edges has stretch factor at least 3.

1 Introduction

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. For any two points p and q in S, we denote by $\delta_H(p,q)$ 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 $\delta_G(p,q) \leq t \cdot \delta_H(p,q)$ for all points p and q in S. The smallest t for which this property holds is called the *stretch factor* of G. Thus, a subgraph G of H with stretch factor 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.

Most of the work on constructing spanners has been done for the case when 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 [8] and Vaidya [9]. For a detailed overview of results on spanners for point sets, see the book by Narasimhan and Smid [6].

For spanners of arbitrary geometric graphs, much less is known. Althöfer *et al.* [1] have shown that for any $t > 1$, every weighted graph H with n vertices contains a subgraph with $O(n^{1+2/(t-1)})$ edges, which 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 [5]: 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

*School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6. Research partially supported by HPCVL, NSERC, MRI, CFI, and MITACS.

looking for spanners with $O(n)$ edges of arbitrary geometric graphs, then the best stretch factor we can obtain is $\Theta(\log n)$.

In this paper, we consider the case when the input graph is a complete k-partite Euclidean graph. Let S be a set of n points in $\mathbb{R}^d$, and let S be partitioned into subsets $S_1, S_2, \ldots, S_k$. Let $K_{S_1 \ldots S_k}$ 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 formally defined as follows:

Problem 1.1 *Let $k > 1$ be an integer, let S be a set of n points in $\mathbb{R}^d$, and let S be partitioned into subsets $S_1, S_2, \ldots, S_k$. Compute a t-spanner of the k-partite complete graph $K_{S_1 \ldots S_k}$ that has a "small" number of edges and whose stretch factor t is "small".*

The main contribution of this paper 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 stretch factor of $t = 3 + \epsilon$. Finally, we show that the latter result is optimal: We give an example of a complete k-partite geometric graph K such that every subgraph of K with a subquadratic number of edges has stretch factor at least 3.

We remark that in a recent paper, Bose *et al.* [2] considered the problem of constructing spanners of point sets that have $O(n)$ edges and whose chromatic number is a most k. This problem is different from ours: Bose *et al.* compute a spanner of the complete graph and their algorithm can choose a "good" k-partition of the vertices. In our problem, the k-partition is given and we want to compute a spanner of the complete k-partite graph.

Possible applications of our algorithm are in wireless networks having the property that communicating nodes are partitioned into sets such that two nodes can communicate if and only if they do not belong to the same set. This would be the case, for example, when Time Division Multiplexing (TDMA) is used. Since the wireless medium prohibits simultaneous transmission and reception at one node, two nodes communicating during the same time slots cannot communicate with each other; see Raman and Chebrolu [7].

The rest of this paper is organized as follows. In Section 2, we recall properties of the Well-Separated Pair Decomposition (WSPD) that we use in our algorithm. In Section 3, we provide an algorithm that solves the problem of constructing a spanner of the complete k-partite graph. In Section 4, we show that the spanner constructed by this algorithm has $O(n)$ edges and that its stretch factor is bounded from above by a constant that depends only on the dimension d. In Section 5, we show how a simple modification to our algorithm improves the stretch factor to $5 + \epsilon$ while still having $O(n)$ edges. In Section 6, we show how to achieve a stretch factor of $3 + \epsilon$ using $O(n \log n)$ edges. We also provide a lower bound of 3 on the stretch factor for the general geometric k-partite spanner problem.

For ease of presentation, we will present all our results for the case when $k = 2$. The generalization to arbitrary values of k is a little bit more involved and will be given in the full version.

2 The Well-Separated Pair Decomposition

In this section, we recall crucial properties of the Well-Separated Pair Decomposition (WSPD) of Callahan and Kosaraju [4] that we use for our construction. Our presentation follows the one in Narasimhan and Smid [6].

Definition 2.1 *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 2.2 *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 exists two balls B_1 and B_2 such that (i) B_1 and B_2 have the same radius, say ρ, (ii) $\beta(X) \subseteq B_1$, (iii) $\beta(Y) \subseteq B_2$, and (iv) $\min\{|xy| : x \in B_1 \cap \mathbb{R}^d, y \in B_2 \cap \mathbb{R}^d\} \geq s\rho$.*

Definition 2.3 *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 are well-separated with respect to s, such that for any two distinct points $p, q \in S$ there is a unique pair $\{X, Y\}$ in the WSPD such that $p \in X$ and $q \in Y$.*

Lemma 2.4 (Lemma 9.1.2 in [6]) *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 $|p'q'| \leq (1 + 4/s)|pq|$.*

Callahan and Kosaraju [3] have shown 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 $q \in Y$. 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 [4] 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.

Definition 2.5 *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 and 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 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 X and Y. Since there is such a unique correspondence, we will 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 $Y = S_v$.

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 2.6 (Lemma 9.5.3 in [6]) *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 2.7 (Lemma 11.3.1 in [6]) *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*

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

3 A First Algorithm

We now show how the WSPD can be used to address the problem of computing a spanner of a complete bipartite graph. In this section, we introduce an algorithm that outputs a graph with constant stretch factor and $O(n)$ edges. The analysis of this algorithm is presented in Section 4. In Section 5, we show how this algorithm can be improved to achieve a stretch factor of $5 + \epsilon$.

The input set $S \subseteq \mathbb{R}^d$ is the disjoint union of two sets R and B containing red and blue points, respectively. The graph K_{RB} is the complete bipartite geometric graph. We first need a definition.

Definition 3.1 *Let T be the split-tree of S that is used to compute the WSPD.*

1. *For a node u in T, we denote by S_u the set of all points in the subtree rooted at u.*
2. *We define BWSPD to be the subset of the WSPD obtained by removing all pairs $\{S_u, S_v\}$ such that $S_u \cup S_v \subseteq R$ or $S_u \cup S_v \subseteq B$.*
3. *A node u in T is* bichromatic *if there exist points r and b in S_u and a node v in T such that $r \in R$, $b \in B$, and $\{S_u, S_v\}$ is in the BWSPD.*
4. *A node u in T is a* red-node *if $S_u \subseteq R$ and there exists a node v in T such that $\{S_u, S_v\}$ is in the BWSPD.*
5. *A red-node u in T is a* red-root *if it does not have a proper ancestor that is a red-node in T.*
6. *A red-node u in T is a* red-leaf *if it does not have another red-node in its subtree.*
7. *A red-node u' in T is a* red-child *of a red-node u in T if u' is in the subtree rooted at u and there is no red-node on the path strictly between u and u'.*
8. *The notions of* blue-node, blue-root, blue-leaf, *and* blue-child *are defined as above, by replacing red by blue.*
9. *For each set S_u that contains at least one red point, $rep_R(S_u)$ denotes a fixed arbitrary red point in S_u. For each set S_u that contains at least one blue point, $rep_B(S_u)$ denotes a fixed arbitrary blue point in S_u.*
10. *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.*
11. *Let u be a red-node or a blue-node in T. Consider all pairs $\{S_v, S_w\}$ in the BWSPD, where v is a red-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 1 computes a spanner of a complete bipartite geometric graph. It considers each pair $\{S_u, S_v\}$ of the WSPD, and decides whether or not it adds a red-blue and/or a blue-red edge between S_u and S_v. The outcome of this decision is based on the following three cases.

Case 1: All points of $S_u \cup S_v$ are of the same color. In this case, there is no edge of K_{RB} to approximate, so the algorithm just ignores this pair.

Case 2: Both S_u and S_v are bichromatic. In this case, the algorithm adds the two edges $(\text{rep}_R(S_u), \text{rep}_B(S_v))$ and $(\text{rep}_B(S_u), \text{rep}_R(S_v))$; see line 21. These two edges will allow us to approximate each edge of K_{RB} having one vertex in S_u and the other vertex in S_v.

Case 3: All points in S_u are of the same color, say red. In this case, only the edge $(\text{rep}_R(S_u), \text{rep}_B(S_v))$ is added; see line 11. In order to approximate each edge of K_{RB} having one (red) vertex in S_u and the other (blue) vertex in S_v, other edges have to be added. This is done in such a way that our final graph contains a "short" path between every red point r of S_u and the red representative $\text{rep}_R(S_u)$ of S_u. Observe that this path must contain blue points that are not in S_u. One way to achieve this is to add an edge between each point of S_u and $\text{rep}_B(\text{cl}(S_u))$; we call this construction a *star*. However, since the subtree rooted at u may contain other red-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 red-leaf; see line 7. If u is a red-node, the algorithm only adds the edge $(\text{rep}_R(S_u), \text{rep}_B(\text{cl}(S_u)))$; see line 10. Then, the algorithm links each red-node u'' that is not a red-root to its red-parent u'. This is done through the edge $(\text{rep}_R(S_{u''}), \text{rep}_B(\text{cl}(u')))$; see line 13.

Algorithm 1

Input: $S = R \cup B$, where R and B are two disjoint sets of red and blue points in $\mathbb{R}^d$, respectively.
Output: A spanner $G = (S, E)$ of the complete bipartite graph K_{RB}.

1: compute the split-tree T of $R \cup B$
2: using T, compute the WSPD with respect to a separation constant $s > 0$
3: using the WSPD, compute the BWSPD
4: $E \leftarrow \emptyset$
5: **for** each red-root u in T **do**
6: **for** each red-leaf u' in the subtree of u **do**
7: for each $r \in S_{u'}$, add to E the edge $(r, \text{rep}_B(\text{cl}(S_{u'})))$
8: **end for**
9: **for** each red-node u' that is in the subtree of u (including u) **do**
10: add to E the edge $(\text{rep}_R(S_{u'}), \text{rep}_B(\text{cl}(S_{u'})))$
11: for each pair $\{S_{u'}, S_{v'}\}$ in the BWSPD, add to E the edge $(\text{rep}_R(S_{u'}), \text{rep}_B(S_{v'}))$
12: **for** each red-child u'' of u' **do**
13: add to E the edge $(\text{rep}_R(S_{u''}), \text{rep}_B(\text{cl}(S_{u'})))$
14: **end for**
15: **end for**
16: **end for**
17: **for** each blue-root u in T **do**
18: //do the same as on lines 5–16, with red and blue interchanged
19: **end for**
20: **for** each $\{S_u, S_v\}$ in the BWSPD for which both u and v are bichromatic **do**
21: add to E the edges $(\text{rep}_R(S_u), \text{rep}_B(S_v))$ and $(\text{rep}_B(S_u), \text{rep}_R(S_v))$
22: **end for**
23: return the graph $G = (R \cup B, E)$

4 Analysis of Algorithm 1

Lemma 4.1 *The graph G computed by Algorithm 1 has $O(|R \cup B|)$ edges.*

Proof: See the appendix. □

Lemma 4.2 *Let r be a point of R, let b be a point of B, and let $\{S_u, S_v\}$ be the pair in the BWSPD for which $r \in S_u$ and $b \in S_v$. Assume that u is a red-node. Then there is a path in G between r and $rep_R(S_u)$ whose length is at most $c|rb|$, where*

$$c = 4\sqrt{d}(\mu d + 1)(1 + 4/s)^3, \qquad \mu = \left\lceil \log\left(\sqrt{d}(1 + 4/s)\right)\right\rceil + 1,$$

and s is the separation constant of the WSPD.

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

$$\begin{array}{ccccccccc} r & \to & \mathrm{rep}_B(\mathrm{cl}(S_{w_0})) & \to & \mathrm{rep}_R(S_{w_0}) & \to & \mathrm{rep}_B(\mathrm{cl}(S_{w_1})) & \to & \mathrm{rep}_R(S_{w_1}) \\ & \to & \ldots & \to & \mathrm{rep}_B(\mathrm{cl}(S_{w_k})) & \to & \mathrm{rep}_R(S_{w_k}) & = & \mathrm{rep}_R(S_u). \end{array}$$

The first edge on this path, i.e., $(r, \mathrm{rep}_B(\mathrm{cl}(S_{w_0})))$, is added to the graph G in line 7 of the algorithm. The edges $(\mathrm{rep}_B(\mathrm{cl}(S_{w_i})), \mathrm{rep}_R(S_{w_i}))$, $0 \le i \le k$, are added to G in line 10. Finally, the edges $(\mathrm{rep}_R(S_{w_{i-1}}), \mathrm{rep}_B(\mathrm{cl}(S_{w_i})))$, $1 \le i \le k$, are added to G in line 13. It follows that Π is a path in G. We will show that the length of Π is at most $c|rb|$.

Let $0 \le i \le k$. Recall the definition of $\mathrm{cl}(S_{w_i})$; see Definition 3.1: We consider all pairs $\{S_x, S_y\}$ in the BWSPD, where x is a red-node on the path in T from w_i to the root, and pick the pair for which $\mathrm{dist}(S_x, S_y)$ is minimum. We denote the pair picked by (S_{x_i}, S_{y_i}). Thus, x_i is a red-node on the path in T from w_i to the root, $\{S_{x_i}, S_{y_i}\}$ is pair in the BWSPD, and $\mathrm{cl}(S_{w_i}) = S_{y_i}$. We define

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

Consider the first edge $(r, \mathrm{rep}_B(\mathrm{cl}(S_{w_0})))$ on the path Π. Since $r \in S_{w_0} \subseteq S_{x_0}$ and $\mathrm{rep}_B(\mathrm{cl}(S_{w_0})) \in S_{y_0}$, it follows from Lemma 2.4 that

$$|r, \mathrm{rep}_B(\mathrm{cl}(S_{w_0}))| \le (1 + 4/s)\mathrm{dist}(S_{x_0}, S_{y_0}) = (1 + 4/s)\ell_0.$$

Let $0 \le i \le k$ and consider the edge $(\mathrm{rep}_B(\mathrm{cl}(S_{w_i})), \mathrm{rep}_R(S_{w_i}))$ on Π. Since $\mathrm{rep}_R(S_{w_i}) \in S_{w_i} \subseteq S_{x_i}$ and $\mathrm{rep}_B(\mathrm{cl}(S_{w_i})) \in S_{y_i}$, it follows from Lemma 2.4 that

$$(1) \quad |\mathrm{rep}_B(\mathrm{cl}(S_{w_i})), \mathrm{rep}_R(S_{w_i})| \le (1 + 4/s)\mathrm{dist}(S_{x_i}, S_{y_i}) = (1 + 4/s)\ell_i.$$

Let $1 \le i \le k$ and consider the edge $(\mathrm{rep}_R(S_{w_{i-1}}), \mathrm{rep}_B(\mathrm{cl}(S_{w_i})))$ on Π. Since $\mathrm{rep}_R(S_{w_{i-1}}) \in S_{w_{i-1}} \subseteq S_{x_i}$ and $\mathrm{rep}_B(\mathrm{cl}(S_{w_i})) \in S_{y_i}$, it follows from Lemma 2.4 that

$$|\mathrm{rep}_R(S_{w_{i-1}}), \mathrm{rep}_B(\mathrm{cl}(S_{w_i}))| \le (1 + 4/s)\mathrm{dist}(S_{x_i}, S_{y_i}) = (1 + 4/s)\ell_i.$$

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 $\sum_{i=0}^{k} \ell_i \le 2\sqrt{d}(\mu d + 1)(1 + 4/s)^2|rb|$. It follows from the definition of $\mathrm{cl}(S_u) = \mathrm{cl}(w_k)$ that $\ell_k = \mathrm{dist}(S_{x_k}, S_{y_k}) \le \mathrm{dist}(S_u, S_v)$. Since, by Lemma 2.4, $\mathrm{dist}(S_u, S_v) \le (1 + 4/s)|rb|$, it follows that

$$(2) \quad \ell_k \le (1 + 4/s)|rb|.$$

Thus, it is sufficient to prove that

$$\sum_{i=0}^{k} \ell_i \le 2\sqrt{d}(\mu d + 1)(1 + 4/s)\ell_k. \tag{3}$$

If $k = 0$, then (3) obviously holds. Assume from now on that $k \ge 1$. For each i with $0 \le i \le k$, 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 $0 \le i \le k$. It follows from Lemma 2.4 that $L_{\max}(\beta(S_{x_i})) \le \frac{2}{s}\ell_i$. Since w_i is in the subtree of x_i, we have $L_{\max}(\beta(S_{w_i})) \le L_{\max}(\beta(S_{x_i}))$. Thus, we have

$$a_i \le \frac{2}{s}\ell_i \text{ for } 0 \le i \le k. \tag{4}$$

Lemma 2.6 states that

$$a_i \le \frac{1}{2}a_{i+d} \text{ for } 0 \le i \le k - d. \tag{5}$$

Let $0 \le i \le k - 1$. Since w_i is a red-node, there is a node w_i' such that $\{S_{w_i}, S_{w_i'}\}$ is a pair in the BWSPD. We have $\ell_i = \text{dist}(S_{x_i}, S_{y_i}) \le \text{dist}(S_{w_i}, S_{w_i'})$. By applying Lemma 2.7, we obtain

$$\text{dist}(S_{w_i}, S_{w_i'}) \le \frac{\sqrt{d}(s+4)}{2} L_{\max}(\beta(S_{\pi(w_i)})) \le \frac{\sqrt{d}(s+4)}{2} L_{\max}(\beta(S_{w_{i+1}})) = \frac{\sqrt{d}(s+4)}{2} a_{i+1}.$$

Thus, we have

$$\ell_i \le \frac{\sqrt{d}(s+4)}{2} a_{i+1} \text{ for } 0 \le i \le k - 1. \tag{6}$$

First assume that $1 \le k \le \mu d$. Let $0 \le i \le k - 1$. By using (6), the fact that the sequence $a_0, a_1, \ldots, a_k$ is non-decreasing, and (4), we obtain

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

Therefore,

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

which is less than the right-hand side in (3).

It remains to consider the case when $k > \mu d$. Let $i \ge 0$ and $j \ge 0$ be integers such that $i + 1 + jd \le k$. By applying (6) once, (5) j times, and (4) once, we obtain

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

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

$$\ell_i \leq \frac{1}{2}\ell_{i+1+\mu d}. \quad (7)$$

By re-arranging the terms in the summation in (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 (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), the fact that the sequence $a_0, a_1, \ldots, a_k$ is non-decreasing, followed by (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 (3) holds. □

Lemma 4.3 *Assuming that the separation constant s of the WSPD satisfies $s \geq 8+6/c$, the graph G computed by Algorithm 1 is a t-spanner of the complete bipartite graph K_{RB}, where $t = 2c+1+4/s$ and c is as in Lemma 4.2.*

Proof: It suffices to show that for each edge (r, b) of K_{RB}, the graph G contains a path between r and b of length at most $t|rb|$. We will prove this by induction on the lengths of the edges in K_{RB}.

Let r be a point in R, let b be a point in B, and let $\{S_u, S_v\}$ be the pair in the BWSPD for which $r \in S_u$ and $b \in S_v$.

The base case is when (r, b) is a shortest edge in K_{RB}. Since $s > 2$, it follows from Lemma 2.4 that u is a red-node and v is a blue-node. In line 11 of Algorithm 1, the edge $(\text{rep}_R(S_u), \text{rep}_B(S_v))$ is added to G. By Lemma 2.4, the length of this edge is at most $(1 + 4/s)|rb|$. The claim follows from two applications of Lemma 4.2 to get from r to $\text{rep}_R(S_u)$ and from $\text{rep}_B(S_v)$ to b.

In the induction step, we distinguish four cases.

Case 1: u is a red-node and v is a blue-node. This case is identical to the base case.

Case 2: u is a bichromatic node and v is a blue-node. In the equivalent of line 11 for the blue-nodes, Algorithm 1 adds the edge $(\text{rep}_R(S_u), \text{rep}_B(S_v))$ to G. By Lemma 2.4, the length of this edge is at most $(1 + 4/s)|rb|$. Let b^* be a blue node in S_u. Since $s > 2$, it follows from Lemma 2.4 that $|rb^*| < |rb|$. Thus, by induction, there is a path in G between r and b^* whose length is at most $t|rb^*|$. By a similar argument, there is a path in G between b^* and $\text{rep}_R(S_u)$ whose length is at most $t|b^*, \text{rep}_R(S_u)|$. Applying Lemma 2.4, it follows that there is a path in G between r and $\text{rep}_R(S_u)$ whose length is at most $t(|rb^*| + |b^*, \text{rep}_R(S_u)|) \leq t(2|rb|/s + 2|rb|/s) = 4t|rb|/s$. By Lemma 4.2, there is a path in G between b and $\text{rep}_B(S_v)$ whose length is at most $c|rb|$. We have shown that there is a path in G between r and b whose length is at most $(1 + 4/s)|rb| + 4t|rb|/s + c|rb|$. Since $s \geq 8 + 6/c$, this quantity is at most $t|rb|$.

Case 3: Both u and v are bichromatic nodes. In line 21, Algorithm 1 adds the edge $(\text{rep}_B(S_u), \text{rep}_R(S_v))$ to G. By Lemma 2.4, the length of this edge is at most $(1 + 4/s)|rb|$. By induction, there is a path in G between r and $\text{rep}_B(S_u)$ whose length is at most $t|r, \text{rep}_B(S_u)|$, which, by Lemma 2.4, is at most $2t|rb|/s$. By a symmetric argument, there is a path in G between b and $\text{rep}_R(S_v)$, whose length is at most $2t|rb|/s$. We have shown that there is a path in G between r and b whose length is at most $(1 + 4/s)|rb| + 4t|rb|/s$, which is at most $t|rb|$. □

Lemma 4.4 *The running time of Algorithm 1 is $O(n \log n)$, where $n = |R \cup B|$.*

Proof: See the appendix. □

To summarize, we have shown the following: Algorithm 1 computes a t-spanner of the complete bipartite graph K_{RB} having $O(n)$ edges, where t is given in Lemma 4.3. The running time of this algorithm is $O(n \log n)$. By choosing the separation constant s sufficiently large, the stretch factor t converges to

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

5 An Improved Algorithm

As before, we are given two disjoint sets R and B of red and blue points in $\mathbb{R}^d$. Intuitively, the way to improve the bound of Lemma 4.2 is by adding shortcuts along the path from each red-leaf to the red-root above it. More precisely, from (7) in the proof of Lemma 4.2, we know that if we

go $1 + \mu d$ levels up in the split-tree, then the length of the edge along the path doubles. Thus, for each red-node in T, we will add edges to all $2\delta(1 + \mu d)$ red-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 5.1 *Let u and u' be red-nodes in the split-tree such that u' is in the subtree rooted at u. For an integer $\zeta \geq 1$, we say that u is* ζ-levels above *u', if there are $\zeta - 1$ red-nodes on the path strictly between u and u'. We say that u' is a* ζ-red-child *of u if u is at most ζ-levels above u'. These notions are defined similarly for the blue-nodes.*

The improved algorithm is given as Algorithm 2. The following lemma generalizes Lemma 4.2.

Algorithm 2

Input: $S = R \cup B$, where R and B are two disjoint sets of red and blue points in $\mathbb{R}^d$, respectively, and a real constant $0 < \epsilon < 1$.

Output: A $(5 + \epsilon)$-spanner $G = (S, E)$ of the complete bipartite graph K_{RB}.

1: 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 $\frac{2^\delta}{2^\delta - 1} \leq 1 + \epsilon/36$.

2: The rest of the algorithm is the same as Algorithm 1, except for lines 12–14, which are replaced by the following:

let $\zeta = 2\delta(\mu d + 1)$

for each ζ-red-child u'' of u' **do**

add to E the edges $(\text{rep}_R(S_{u''}), \text{rep}_B(\text{cl}(S_{u'})))$ and $(\text{rep}_B(\text{cl}(S_{u''})), \text{rep}_R(S_{u'}))$

end for

Lemma 5.2 *Let r be a point of R, let b be a point of B, and let $\{S_u, S_v\}$ be the pair in the BWSPD for which $r \in S_u$ and $b \in S_v$. Assume that u is a red-node. Let G be the graph computed by Algorithm 2. There is a path in G between r and $rep_R(S_u)$ whose length is at most $(2 + \epsilon/3)|rb|$.*

Proof: The complete proof is given in the appendix. Here, we only state how the path can be obtained. Let w be the red-leaf such that $r \in S_w$, and let $w = w_0, w_1, \ldots, w_k = u$ be the sequence of red-nodes that are on the path in T from w to u. Throughout the proof, we will 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 4.2.

If $0 \leq k \leq 2\delta(\mu d + 1)$, then the path

$$r \to \text{rep}_B(\text{cl}(S_w)) \to \text{rep}_R(S_u)$$

satisfies the condition in the lemma. 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 red-nodes

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

The path

$$\begin{array}{rcccl}
r & \to & \text{rep}_B(\text{cl}(S_{w_0})) & \to & \text{rep}_R(S_{w_{\delta(\mu d+1)+m}}) \\
& \to & \text{rep}_B(\text{cl}(S_{w_{2\delta(\mu d+1)+m}})) & \to & \text{rep}_R(S_{w_{2\delta(\mu d+1)+m}}) \\
& \to & \text{rep}_B(\text{cl}(S_{w_{3\delta(\mu d+1)+m}})) & \to & \text{rep}_R(S_{w_{3\delta(\mu d+1)+m}}) \\
& \vdots & & \vdots & \\
& \to & \text{rep}_B(\text{cl}(S_{w_k})) & \to & \text{rep}_R(S_{w_k}) \quad = \quad \text{rep}_R(S_u)
\end{array}$$

satisfies the condition in the lemma. □

Lemma 5.3 *Let $n = |R \cup B|$. The graph G computed by Algorithm 2 is a $(5+\epsilon)$-spanner of the complete bipartite graph K_{RB} and the number of edges of this graph is $O(n)$. The running time of Algorithm 2 is $O(n \log n)$.*

Proof: See the appendix. □

We have proved the following result.

Theorem 5.4 *Let S be a set of n points in $\mathbb{R}^d$ which is partitioned into two subsets R and B, 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 bipartite graph K_{RB} having $O(n)$ edges.*

6 Improving the Stretch Factor

We have shown how to compute a $(5+\epsilon)$-spanner with $O(n)$ edges of any complete bipartite graph. In this section, we show that if we are willing to use $O(n \log n)$ edges, the stretch factor can be reduced to $3+\epsilon$. We start by showing that a stretch factor less than 3 using a subquadratic number of edges is not possible.

Theorem 6.1 *For every real number $t < 3$, there is no algorithm that, when given as input two arbitrary disjoint sets R and B of points in $\mathbb{R}^d$, computes a t-spanner of the complete bipartite graph K_{RB} having less than $|R| \cdot |B|$ edges.*

Proof: Let us assume by contradiction that there exists such an algorithm $\mathcal{A}$ for some real number $t < 3$. Let $\epsilon = 3 - t$, let B_1 and B_2 be two balls of diameter $\epsilon/6$ such that the distance between their centers is $1 + \epsilon/6$. Let R be a set of points that are contained in B_1 and let B be a set of points that are contained in B_2. Let G be the graph obtained by running algorithm $\mathcal{A}$ on R and B. By our hypothesis, G has less than $|R| \cdot |B|$ edges. Thus, there exist a point r in R and a point b in B, such (r, b) is not an edge in G. Since any path in G between r and b contains at least three edges, the length of the shortest path in G between r and b in G is at least three. Since $|rb| \leq 1 + \epsilon/3$, it follows that the stretch factor of G is at least $\frac{3}{1+\epsilon/3}$, which is greater than $t = 3 - \epsilon$, contradicting the existence of $\mathcal{A}$. □

Theorem 6.2 *Let S be a set of n points in $\mathbb{R}^d$ which is partitioned into two subsets R and B, 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 bipartite graph K_{RB} 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 $\{\{x\}, B\}$, where x ranges over all points of X. Thus, in this new WSPD, each pair contains at least one singleton set. Callahan and Kosaraju [4] showed that this new WSPD consists of $O(n \log n)$ pairs.

We run Algorithm 2 on R and B, using this new WSPD. Let G be the graph that is computed by this algorithm. Observe that Lemma 5.2 still holds for G. In the proof of Lemma 5.3 of the upper bound on the stretch factor of G, we apply Lemma 5.2 only once. Therefore, the stretch factor of G is at most $3 + \epsilon$. □

References

[1] 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.

[2] Prosenjit Bose, Paz Carmi, Mathieu Couture, Anil Maheshwari, Michiel Smid, and Norbert Zeh. Geometric spanners with small chromatic number. In *Proceedings of the 5th Workshop on Approximation and Online Algorithms*, Lecture Notes in Computer Science, Berlin, 2007. Springer-Verlag.

[3] 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*, pages 291–300, 1993.

[4] Paul B. Callahan and S. Rao Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. ACM*, 42(1):67–90, 1995.

[5] J. Gudmundsson and M. Smid. On spanners of geometric graphs. In *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory*, volume 4059 of *Lecture Notes in Computer Science*, pages 388–399, Berlin, 2006. Springer-Verlag.

[6] Giri Narasimhan and Michiel Smid. *Geometric Spanner Networks*. Cambridge University Press, New York, NY, USA, 2007.

[7] Bhaskaran Raman and Kameswari 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*, pages 156–169, New York, NY, USA, 2005. ACM Press.

[8] J. S. Salowe. Constructing multidimensional spanner graphs. *International Journal of Computational Geometry & Applications*, 1:99–107, 1991.

[9] 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.

Appendix

Proof of Lemma 4.1: For each red-leaf u', the algorithm adds $|S_{u'}|$ edges to G in line 7. Since the sets $S_{u'}$, where u' ranges over all red-leaves, are pairwise disjoint, the total number of edges that are added in line 7 is $O(|R|)$.

The total number of edges that are added in lines 11 and 21 is at most twice the number of pairs in the BWSPD. Since the WSPD contains $O(|R \cup B|)$ pairs (see [4]), the same upper bound holds for the number of edges added in lines 11 and 21.

It remains to consider the edges that are added in lines 10 and 13. The total number of edges added in these lines is at most twice the number of nodes in the split-tree, which is $O(|R \cup B|)$.

We have shown that the total number of edges added to G, due to the red-nodes, is $O(|R \cup B|)$. Since the algorithm works in a completely symmetric way for the blue-nodes, the proof is complete.

Proof of Lemma 4.4: Using the results of Callahan and Kosaraju [4], the split-tree T and the WSPD can be computed in $O(n \log n)$ time. The time for the rest of the algorithm, i.e., lines 3–23, 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.

Proof of Lemma 5.2: Let w be the red-leaf such that $r \in S_w$, and let $w = w_0, w_1, \ldots, w_k = u$ be the sequence of red-nodes that are on the path in T from w to u.

Throughout the proof, we will 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 4.2.

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

$$r \to \mathrm{rep}_B(\mathrm{cl}(S_w)) \to \mathrm{rep}_R(S_u).$$

It follows from Algorithm 2 that Π is a path in G. Since $r \in S_w = S_{w_0} \subseteq S_{x_0}$ and $\mathrm{rep}_B(\mathrm{cl}(S_w)) = \mathrm{rep}_B(\mathrm{cl}(S_{w_0})) \in S_{y_0}$, it follows from Lemma 2.4 that

$$|r, \mathrm{rep}_B(\mathrm{cl}(S_w))| \leq (1 + 4/s)\mathrm{dist}(S_{x_0}, S_{y_0}) = (1 + 4/s)\ell_0. \tag{8}$$

Since $\{S_u, S_v\}$ is one of the pairs that is considered in the definition of $\mathrm{cl}(S_{w_0})$, we have $\mathrm{dist}(S_{x_0}, S_{y_0}) \leq \mathrm{dist}(S_u, S_v)$. Again by Lemma 2.4, we have $\mathrm{dist}(S_u, S_v) \leq (1 + 4/s)|rb|$. Thus, we have shown that

$$|r, \mathrm{rep}_B(\mathrm{cl}(S_w))| \leq (1 + 4/s)^2|rb|.$$

By the triangle inequality, we have

$$|\mathrm{rep}_B(\mathrm{cl}(S_w)), \mathrm{rep}_R(S_u)| \leq |\mathrm{rep}_B(\mathrm{cl}(S_w)), r| + |r, \mathrm{rep}_R(S_u)|.$$

Since r and $\mathrm{rep}_R(S_u)$ are both contained in S_u, it follows from Lemma 2.4 that $|r, \mathrm{rep}_R(S_u)| \leq (2/s)|rb|$. Thus, we have

$$|\mathrm{rep}_B(\mathrm{cl}(S_w)), \mathrm{rep}_R(S_u)| \leq (1 + 4/s)^2|rb| + (2/s)|rb|.$$

We have shown that the length of the path Π is at most

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

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

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 red-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

$$\begin{array}{cccccccc}
r & \to & \mathrm{rep}_B(\mathrm{cl}(S_{w_0})) & \to & \mathrm{rep}_R(S_{w_{\delta(\mu d+1)+m}}) & & \\
 & \to & \mathrm{rep}_B(\mathrm{cl}(S_{w_{2\delta(\mu d+1)+m}})) & \to & \mathrm{rep}_R(S_{w_{2\delta(\mu d+1)+m}}) & & \\
 & \to & \mathrm{rep}_B(\mathrm{cl}(S_{w_{3\delta(\mu d+1)+m}})) & \to & \mathrm{rep}_R(S_{w_{3\delta(\mu d+1)+m}}) & & \\
 & \vdots & & \vdots & & & \\
 & \to & \mathrm{rep}_B(\mathrm{cl}(S_{w_k})) & \to & \mathrm{rep}_R(S_{w_k}) & = & \mathrm{rep}_R(S_u).
\end{array}$$

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

We have shown already (see (8)) that the length of the first edge on Π satisfies

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

The length of the second edge satisfies

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

Since r and $\mathrm{rep}_R(S_{w_{\delta(\mu d+1)+m}})$ are both contained in S_u, it follows from Lemma 2.4 that

$$|r, \mathrm{rep}_R(S_{w_{\delta(\mu d+1)+m}})| \le (2/s)|rb|.$$

Thus, the length of the second edge on Π satisfies

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

Let $2 \le j \le m'$. We have seen in (1) in the proof of Lemma 4.2 that the length of the edge

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

satisfies

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

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

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

and

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

it follows from Lemma 2.4 that the length of the edge

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

satisfies

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

We have shown that the length of Π is at most

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

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

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

Recall inequality (7) in the proof of Lemma 4.2, 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 (2) in the proof of Lemma 4.2, we have

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

We have shown that the length of the path Π is at most

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

Our choices of s and δ (see line 1 in Algorithm 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

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

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

Proof of Lemma 5.3: The proof for the upper bound on the stretch factor is similar to the one of Lemma 4.3. The difference is that instead of the value c that was used in the proof of Lemma 4.3, we now use the value $c = 2 + \epsilon/3$. Thus, the stretch factor for the base case of the induction and for Case 1 is

$$1 + 4/s + 2c = 1 + 4/s + 4 + 2\epsilon/3,$$

which is at most $5 + \epsilon$, because of our choice for s. For Case 2, the stretch factor is at most

$$1 + 4/s + 4t/s + c = 3 + \epsilon/3 + (4/s)(6 + \epsilon),$$

which is at most $5 + \epsilon$, again because of our choice for s. Finally, the stretch factor for Case 3 is at most that of Case 2; thus, it is at most $5 + \epsilon$.

The analysis for the number of edges is the same as in Lemma 4.1, except that the number of edges that are added to each red-node on line 13 is $\delta(\mu d + 1)$ instead of one as is in Algorithm 1. Finally, the analysis of the running time is the same as in Lemma 4.4.