Approximating the average stretch factor of geometric graphs*

Siu-Wing Cheng† Christian Knauer‡ Stefan Langerman§ Michiel Smid¶

Abstract

Let G be a geometric graph whose vertex set S is a set of n points in $\mathbb{R}^d$. The stretch factor of two distinct points p and q in S is the ratio of their shortest-path distance in G and their Euclidean distance. We consider the problem of approximating the sum of all $\binom{n}{2}$ stretch factors determined by all pairs of points in S. We show that for paths, cycles, and trees, this sum can be approximated, within a factor of $1+\epsilon$, in $O(n\,polylog(n))$ time. For plane graphs, we present a $(2+\epsilon)$-approximation algorithm with running time $O(n^{5/3}polylog(n))$, and a $(4+\epsilon)$-approximation algorithm with running time $O(n^{3/2}polylog(n))$.

1 Introduction

Let S be a set of n points in $\mathbb{R}^d$ and let G be a connected graph with vertex set S in which the weight of any edge (p,q) is equal to the Euclidean distance $|pq|$ between p and q. The length of a path in G is defined to be the sum of the weights of the edges on the path. For any two points p and q of S, we denote by $|pq|_G$ the minimum length of any path in G between p and q. If $p \neq q$, then the *stretch factor* of p and q is defined to be $|pq|_G/|pq|$. If $t \geq 1$ is a real number such that each pair of distinct points in S has stretch factor at most t, then we say that G is a *t-spanner* of S. The smallest value of t such that G is a t-spanner of S is called the *stretch factor* of G.

The problem of computing, given any set S of points in $\mathbb{R}^d$ and any $t > 1$, a t-spanner of S, has been well-studied; see the book by Narasimhan and Smid [8].

For the related problem of computing, or approximating, the stretch factor of a given geometric graph, much less is known. Narasimhan and Smid [7] show that the problem of approximating the stretch factor of any geometric graph on n vertices can be reduced to performing approximate shortest-path queries for $O(n)$ pairs of points. Agarwal *et al.* [1] show that the exact stretch factor of a geometric path, tree, and cycle on n points in the plane can be computed in $O(n \log n)$, $O(n \log^2 n)$, and $O(n\sqrt{n} \log n)$ expected time, respectively. They also present algorithms for the three-dimensional versions of these problems. Klein *et al.* [6] consider the problem of reporting all pairs of vertices whose stretch factor is at least some given value t; they present efficient algorithms for the cases when the input graph is a geometric path, tree, or cycle.

Given a method to compute the stretch factor of a graph, a natural question is whether the graph connectivity can be adjusted to lower the stretch factor. For instance, this would be helpful in reducing the maximum commute time in a road network and related problems

*Research of Cheng was supported by Research Grant Council, Hong Kong, China (project no. 612107). Research of Smid was supported by NSERC.

†Department of Computer Science and Engineering, HKUST, Hong Kong
‡Institute of Computer Science, Universität Bayreuth
§Département d'Informatique, Université Libre de Bruxelles. Maître de Recherches du F.R.S.-FNRS.
¶School of Computer Science, Carleton University, Ottawa

have been considered (e.g. [4]). However, the stretch factor can be high just because the stretch factors of a few pairs of points are high while the stretch factors of the other pairs are low. A more robust measure is the *average* stretch factor which we define as follows. Let $SSF(G)$ denote the sum of all stretch factors, i.e.,

$$SSF(G) = \sum_{\{p,q\} \in \mathcal{P}_2(S)} \frac{|pq|_G}{|pq|},$$

where $\mathcal{P}_2(S)$ denotes the set of all $\binom{n}{2}$ unordered pairs of distinct elements in S. The value $SSF(G)/\binom{n}{2}$ is equal to the *average* stretch factor of the graph G.

To the best of our knowledge, even for a simple graph G such as a path, it is not known if $SSF(G)$ can be computed in $o(n^2)$ time. We remark that Wulff-Nilsen shows in [9] that the related problem of computing the Wiener index (i.e., $\sum_{\{p,q\} \in \mathcal{P}_2(S)} |pq|_G$) of an unweighted planar graph can be solved in $O(n^2 \log\log n / \log n)$ time.

In this paper, we consider the problem of approximating $SSF(G)$. We start in Section 2 by showing that, not surprisingly, the well-separated pair decomposition (WSPD) of Callahan and Kosaraju [3] can be used to approximate $SSF(G)$. In Section 3, we apply this general approach to compute a $(1+\epsilon)$-approximation to $SSF(G)$ in $O(n \log^2 n)$ time, for the cases when G is a path or a cycle. In Section 4, we modify the general approach of Section 2 and show how to compute a $(1+\epsilon)$-approximation to $SSF(G)$ in $O(n \log^2 n / \log\log n)$ time for the case when G is a tree. Finally, in Section 5, we consider plane graphs. We further modify the general approach of Section 2 and obtain a $(2+\epsilon)$-approximation to $SSF(G)$ in $O((n \log n)^{5/3})$ time, and a $(4+\epsilon)$-approximation in $O(n^{3/2} \log^2 n)$ time.

2 The general approach using well-separated pairs

Let $s > 0$ be a real number, called the *separation ratio*. We say that two point sets A and B in $\mathbb{R}^d$ are *well-separated* with respect to s, if there exist two balls, one containing A and the other containing B, of the same radius, say ρ, which are at least $s\rho$ apart. If A and B are well-separated, a and a' are points in A, and b and b' are points in B, then it is easy to verify that

$$|ab| \leq (1 + 4/s)|a'b'|. \tag{1}$$

Let S be a set of n points in $\mathbb{R}^d$. A *well-separated pair decomposition (WSPD)* of S is a sequence $\{A_1, B_1\}, \ldots, \{A_m, B_m\}$ of well-separated pairs of subsets of S, such that, for any two distinct points p and q in S, there is a unique index i such that $p \in A_i$ and $q \in B_i$ or $p \in B_i$ and $q \in A_i$.

Callahan and Kosaraju [3] have shown that a WSPD can be obtained from the *split-tree* $T(S)$ of the point set S. This tree is defined as follows: If $n = 1$, then $T(S)$ consists of one single node storing the only element of S. Assume that $n \geq 2$. Consider the bounding box $\mathcal{B}$ of S. By splitting the longest edge of $\mathcal{B}$ into two parts of equal size, we obtain two boxes $\mathcal{B}_1$ and $\mathcal{B}_2$. The split tree $T(S)$ consists of a root having two subtrees, which are recursively defined split trees $T(S_1)$ and $T(S_2)$ for the point sets $S_1 = S \cap \mathcal{B}_1$ and $S_2 = S \cap \mathcal{B}_2$, respectively.

Given a separation ratio $s > 0$, the split tree $T(S)$ can be used to compute a WSPD of S, where each subset A_i (and each subset B_i) corresponds to a node v of the split-tree: A_i equals the set S_v of all points that are stored at the leaves of the subtree rooted at v.

Theorem 1 (Callahan and Kosaraju [3]) *Let S be a set of n points in $\mathbb{R}^d$ and let $s > 0$ be a real constant. In $O(n \log n)$ time, the split tree $T(S)$ and a corresponding WSPD*

$\{A_1, B_1\}, \ldots, \{A_m, B_m\}$ *of* S *can be computed, such that* $m = O(n)$ *and* $\sum_{i=1}^{m} \min(|A_i|, |B_i|) = O(n \log n)$.

If G is a connected graph with vertex set S, then

$$SSF(G) = \sum_{i=1}^{m} \sum_{p \in A_i, q \in B_i} \frac{|pq|_G}{|pq|}.$$

Let $s = 4/\epsilon$. By (1), all distances $|pq|$, where $p \in A_i$ and $q \in B_i$, are within a factor of $1 + \epsilon$ of each other. For each i, choose an arbitrary point x_i in A_i and an arbitrary point y_i in B_i, and consider the summation

$$SSF'(G) = \sum_{i=1}^{m} \frac{1}{|x_i y_i|} \sum_{p \in A_i, q \in B_i} |pq|_G.$$

Then $1/(1+\epsilon) \leq SSF'(G)/SSF(G) \leq 1+\epsilon$. In order to compute $SSF'(G)$, we need to compute the values

$$\sum_{p \in A_i, q \in B_i} |pq|_G. \tag{2}$$

3 Paths and cycles

Assume that the graph G is a path $(p_1, p_2, \ldots, p_n)$ on the points of the set S. For two indices i and j with $1 \leq i < j \leq n$, we say that p_i is to the *left* of p_j in G, and p_j is to the *right* of p_i in G.

Before we present the algorithm that approximates $SSF(G)$, we describe the main idea. Consider a pair $\{A_i, B_i\}$ of the WSPD. Let p be an arbitrary point in A_i, let $b_1, \ldots, b_k$ be the points in B_i that are to the left of p in G, and let $b'_1, \ldots, b'_{k'}$ be the points in B_i that are to the right of p in G. Then

$$\sum_{q \in B_i} |pq|_G = (k - k')|p_1 p|_G + \sum_{j=1}^{k'} |p_1 b'_j|_G - \sum_{j=1}^{k} |p_1 b_j|_G. \tag{3}$$

Let v be the node in the split tree such that $B_i = S_v$, i.e., B_i is the subset of S that is stored in the subtree rooted at v. Assume that we have a balanced binary search tree $\mathcal{T}_v$ storing the points of S_v at its leaves, sorted according to their indices in the path G. Also assume that each node u of this tree stores (i) the number of points stored in the subtree of u and (ii) the sum of the path lengths $|p_1 q|_G$, where q ranges over all points stored in the subtree of u. Then by searching in $\mathcal{T}_v$ for p, we obtain, in $O(\log |B_i|) = O(\log n)$ time, (i) a partition, into $O(\log n)$ canonical subsets, of all points in B_i that are to the left of p in G, and (ii) a partition, into $O(\log n)$ canonical subsets, of all points in B_i that are to the right of p in G. From the information stored at the canonical nodes, we can compute the summation in (3) in $O(\log n)$ time.

Based on this discussion, we obtain the following algorithm.

Step 1: Compute the split tree $T(S)$ and the corresponding WSPD $\{A_1, B_1\}, \ldots, \{A_m, B_m\}$ of Theorem 1, with separation ratio $s = 4/\epsilon$. Assume that $|A_i| \leq |B_i|$ for all $1 \leq i \leq m$.

Step 2: Traverse the path G and store with each point p_i $(1 \leq i \leq n)$ the path length $|p_1 p_i|_G$.

Step 3: Traverse the split tree $T(S)$ in post-order, maintaining the following invariant: After having just visited node v, this node contains a pointer to the above data structure $\mathcal{T}_v$ storing the set S_v. Let v be the node of $T(S)$ that is currently visited.

1. If v is a leaf of $T(S)$, then initialize $\mathcal{T}_v$ such that it contains only the point stored at v. Otherwise, let v_1 and v_2 be the two children of v. If the size of $\mathcal{T}_{v_1}$ is at most that of $\mathcal{T}_{v_2}$, then insert all elements of $\mathcal{T}_{v_1}$ into $\mathcal{T}_{v_2}$, discard $\mathcal{T}_{v_1}$, and rename $\mathcal{T}_{v_2}$ as $\mathcal{T}_v$. Otherwise, insert all elements of $\mathcal{T}_{v_2}$ into $\mathcal{T}_{v_1}$, discard $\mathcal{T}_{v_2}$, and rename $\mathcal{T}_{v_1}$ as $\mathcal{T}_v$.

2. For each pair $\{A_i, B_i\}$ in the WSPD for which $B_i = S_v$, do the following: Let w be the node of the split tree such that $A_i = S_w$. Traverse the subtree rooted at w and for each point p stored in this subtree, use $\mathcal{T}_v$ to compute the value in (3). The sum of all these values (over all p in A_i) gives the summation in (2).

Theorem 2 *Let G be a path on n points in $\mathbb{R}^d$ and let $\epsilon > 0$ be a real constant. In $O(n \log^2 n)$ time, we can compute a real number that lies between $SSF(G)/(1+\epsilon)$ and $(1+\epsilon)SSF(G)$.*

By using a slight modification of the above algorithm, we can prove the following result:

Theorem 3 *Let G be a cycle on n points in $\mathbb{R}^d$ and let $\epsilon > 0$ be a real constant. In $O(n \log^2 n)$ time, we can compute a real number that lies between $SSF(G)/(1+\epsilon)$ and $(1+\epsilon)SSF(G)$.*

4 Trees

Let S be a set of n points in $\mathbb{R}^d$ and let G be a spanning tree of S. Assume that $n \geq 3$. Let c be a *centroid* of G, i.e., c is a node whose removal from G (together with its incident edges) results in two forests G'_1 and G'_2, each one having size at most $2n/3$. It is well known that such a centroid always exists and can be computed in $O(n)$ time. Let G_1 be the tree obtained by adding c to G'_1, together with the edges of G between c and G'_1. Define G_2 similarly with respect to G'_2. We have

$$SSF(G) = SSF(G_1) + SSF(G_2) + \sum_{p \in G'_1} \sum_{q \in G'_2} \frac{|pq|_G}{|pq|}. \tag{4}$$

We will show that the summation in (4) can be approximated in $O(n \log n)$ time. Therefore, by recursively approximating the values $SSF(G_1)$ and $SSF(G_2)$, we obtain an approximation of $SSF(G)$ in $O(n \log^2 n)$ time.

We color each point of G'_1 *red* and each point of G'_2 *blue*. The centroid c does not get a color. Consider the split tree $T(S)$ and the corresponding WSPD of Theorem 1, where $s = 4/\epsilon$. For each i, let A_i^r and A_i^b be the set of red and blue points in A_i, respectively, and let B_i^r and B_i^b be the set of red and blue points in B_i, respectively. Then (4) is equal to

$$\sum_{i=1}^{m} \left(\sum_{p \in A_i^r} \sum_{q \in B_i^b} \frac{|pq|_G}{|pq|} + \sum_{p \in B_i^r} \sum_{q \in A_i^b} \frac{|pq|_G}{|pq|} \right).$$

For each i, fix $x_i \in A_i$ and $y_i \in B_i$. Then

$$\sum_{i=1}^{m} \left(\frac{1}{|x_i y_i|} \left(\sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G + \sum_{p \in B_i^r} \sum_{q \in A_i^b} |pq|_G \right) \right)$$

approximates the summation in (4) within a factor of $1+\epsilon$. Observe that

$$\sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G = |B_i^b| \sum_{p \in A_i^r} |pc|_G + |A_i^r| \sum_{q \in B_i^b} |cq|_G$$

and

$$\sum_{p \in B_i^r} \sum_{q \in A_i^b} |pq|_G = |A_i^b| \sum_{p \in B_i^r} |pc|_G + |B_i^r| \sum_{q \in A_i^b} |cq|_G.$$

This leads to the following algorithm for approximating the summation in (4):

Traverse the tree G in postorder (assuming it is rooted at the centroid c) and store with each point p the path length $|pc|_G$.

Traverse the split tree $T(S)$ in postorder and store with each node v the number of red points in S_v and the number of blue points in S_v.

For each leaf v of the split tree $T(S)$, do the following: Let p be the point stored at v. If p is red, then set $\mathit{redsum}(v) = |pc|_G$ and $\mathit{bluesum}(v) = 0$. If p is blue, then set $\mathit{redsum}(v) = 0$ and $\mathit{bluesum}(v) = |pc|_G$. If p is the centroid, then set $\mathit{redsum}(v) = 0$ and $\mathit{bluesum}(v) = 0$.

Traverse the split tree $T(S)$ in postorder. For each internal v, with children v_1 and v_2, set $\mathit{redsum}(v) = \mathit{redsum}(v_1) + \mathit{redsum}(v_2)$ and $\mathit{bluesum}(v) = \mathit{bluesum}(v_1) + \mathit{bluesum}(v_2)$.

Consider a pair $\{A_i, B_i\}$ in the WSPD, and let v and w be the nodes in the split tree such that $A_i = S_v$ and $B_i = S_w$. Node v stores the values $|A_i^r|$ and $|A_i^b|$. Also, the values of $\mathit{redsum}(v)$ and $\mathit{bluesum}(v)$ are equal to $\sum_{p \in A_i^r} |pc|_G$ and $\sum_{q \in A_i^b} |cq|_G$, respectively. Similarly, from the information stored at w, we obtain the values of $|B_i^r|$, $|B_i^b|$, $\sum_{p \in B_i^r} |pc|_G$, and $\sum_{q \in B_i^b} |cq|_G$.

Theorem 4 *Let G be a tree on n points in $\mathbb{R}^d$ and let $\epsilon > 0$ be a real constant. In $O(n \log^2 n)$ time, we can compute a real number that lies between $SSF(G)/(1+\epsilon)$ and $(1+\epsilon)SSF(G)$.*

We now show how the running time can be improved by a doubly-logarithmic factor. Consider the recursion tree of the above divide-and-conquer algorithm, and consider a node in this tree. Let S' be the set of points in S that are involved in the call at this node, and let n' be the size of S'. The total time spent at this node is equal to the sum of (i) $O(n' \log n')$, which is the time to compute the split tree and the WSPD of S', and (ii) $O(n')$, which is the time for the rest of the algorithm at this node of the recursion tree. Assume that, at this node, we do not compute the split tree and the WSPD of S', but use the split tree and the WSPD for the entire point set S. Consider a centroid c' of the subtree of G that corresponds to S'. This centroid splits the set S' into two subsets, which we color red and blue, whereas the centroid c' does not get a color. Also, no point of $S \setminus S'$ gets a color. Now we can use the split tree $T(S)$ to compute an approximation of the summation in (4) in $O(n)$ time.

Let h be a positive integer such that $h = O(\log n)$. By using the split tree $T(S)$ and the corresponding WSPD of the entire set S at the levels $0, 1, \ldots, h-1$ of the recursion tree, the total time spent at these levels is $O(n \log n + 2^h n)$. At each node at level h of the recursion tree, we compute the split tree and the WSPD for the points involved in the recursive call at this node. In this way, the total time of our algorithm is $O((n \log n + 2^h n)\frac{\log n}{h})$. For $h = \log \log n$, this gives the following result:

Theorem 5 *Let G be a tree on n points in $\mathbb{R}^d$ and let $\epsilon > 0$ be a real constant. In $O(n \log^2 n / \log \log n)$ time, we can compute a real number that lies between $SSF(G)/(1+\epsilon)$ and $(1+\epsilon)SSF(G)$.*

5 Plane graphs

Let G be a plane connected graph whose vertex set is a set S of n points in $\mathbb{R}^d$, and let C be a *separator* of G. That is, C is a subset of the point set S, such that the following is true: By removing the points of C (together with their incident edges) from G, we obtain two graphs, with vertex sets, say, A and B, such that G does not contain any edge joining some point of A with some point of B.

For any point p in $S \setminus C$, let p' be a point of C for which $|pp'|_G$ is minimum. The following lemma appears in Arikati *et al.* [2].

Lemma 1 *Let p be a point in A, let q be a point in B, and assume that $|pp'|_G \leq |qq'|_G$. Then*

$$|pp'|_G + |p'q|_G \leq 2|pq|_G.$$

The following notions were introduced by Frederickson [5]. A *division* of G is a sequence $R_1, \ldots, R_k$ of subsets of S (called *regions*), for some $k \geq 1$, such that $\cup_{i=1}^{k} R_i = S$ and for each i and each p in R_i,

1. either p is an *interior* point of R_i, i.e., (i) p is not contained in any other subset in the sequence and (ii) for every edge (p, q) in G, the point q also belongs to R_i
2. or p is a *boundary* point of R_i, i.e., p is contained in at least one other subset in the sequence.

A division $R_1, \ldots, R_k$ is called an *r-division*, if $k = O(n/r)$ and each region R_i contains at most r points and $O(\sqrt{r})$ boundary points. Frederickson [5] has shown that such an r-division can be computed in $O(n \log n)$ time. The total number of boundary points in an r-division is $k \cdot O(\sqrt{r}) = O(n/\sqrt{r})$. Also, for any i, the boundary points of R_i form a separator of the graph G.

5.1 Approximating $SSF(G)$ within $2 + \epsilon$

Consider an r-division $R_1, \ldots, R_k$ of G. It partitions the pairs in $\mathcal{P}_2(S)$ into two groups: The pair $\{p, q\}$ is of *type 1*, if p and q belong to the same region; otherwise, $\{p, q\}$ is of *type 2*. For $j \in \{1, 2\}$, we define

$$SSF_j(G) = \sum_{\{p,q\} \text{ of type } j} \frac{|pq|_G}{|pq|}.$$

Then $SSF(G) = SSF_1(G) + SSF_2(G)$. We start by showing how a 2-approximation of $SSF_1(G)$ can be computed. Using the algorithm of [2], we preprocess the graph G in $O(n^{3/2})$ time, after which, for any two points p and q, a 2-approximation of $|pq|_G$ can be computed in $O(\log n)$ time. Since the total number of pairs of type 1 is at most $kr^2 = O(rn)$, this leads to a 2-approximation of $SSF_1(G)$ in $O(n^{3/2} + rn \log n)$ time. In the rest of this section, we will show how to compute a $(2 + \epsilon)$-approximation of $SSF_2(G)$ in time

$$O\left(\frac{n^2 \log^2 n}{\sqrt{r}}\right). \tag{5}$$

By choosing $r = (n \log n)^{2/3}$, we obtain the following result:

Theorem 6 *Let G be a plane graph on n points in $\mathbb{R}^d$ and let $\epsilon > 0$ be a real constant. In $O((n \log n)^{5/3})$ time, we can compute a real number that lies between $SSF(G)/(2 + \epsilon)$ and $(2 + \epsilon)SSF(G)$.*

A $(2 + \epsilon)$-approximation of $SSF_2(G)$ is obtained in the following way. For each boundary point p, we run Dijkstra's shortest-path algorithm with source p. In this way, we obtain the shortest-path lengths for all pairs p, q of points, where p ranges over all boundary points and q ranges over all points of S. We store the values $|pq|_G$ in a table so that we can access any one

of them in $O(1)$ time. This part of the algorithm takes $O((n^2 \log n)/\sqrt{r})$ time, which is within the time bound in (5).

Nest, we compute the split tree $T(S)$ and the corresponding WSPD $\{A_1, B_1\}, \ldots, \{A_m, B_m\}$ of Theorem 1, with separation ratio $s = 8/\epsilon$. Assume that $|A_i| \leq |B_i|$ for all $1 \leq i \leq m$.

We repeat the following for each region R in the r-division $R_1, \ldots, R_k$. We color each point of R *red* and color each point of $S \setminus R$ *blue*. For each i with $1 \leq i \leq m$, let A_i^r and A_i^b be the set of red and blue points in A_i, respectively, and let B_i^r and B_i^b be the set of red and blue points in B_i, respectively.

As in Section 4, in order to obtain a $(2+\epsilon)$-approximation of $SSF_2(G)$, it is sufficient to compute a 2-approximation of the values

$$\sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G \text{ and } \sum_{p \in B_i^r} \sum_{q \in A_i^b} |pq|_G, \tag{6}$$

for all i with $1 \leq i \leq m$. We will show how a 2-approximation of the first summation in (6) can be computed. The second summation can be approximated in a symmetric way.

Let $b_1, \ldots, b_\ell$ be the boundary points of the region R. Recall that $\ell = O(\sqrt{r})$. For each point p of S, let p' be a point in $\{b_1, \ldots, b_\ell\}$ for which $|pp'|_G$ is minimum. Observe that, since we have run Dijkstra's algorithm from every boundary point, each point p "knows" the closest boundary point p'.

Consider a pair $\{A_i, B_i\}$ in the WSPD, let v be the node in the split tree $T(S)$ such that $B_i = S_v$, and let S_v^b be the set of blue points in S_v. Assume that v stores the following information:

1. Balanced binary search trees $\mathcal{T}_{v,j}$ for $1 \leq j \leq \ell$. Each such tree $\mathcal{T}_{v,j}$ stores the points q of S_v^b at its leaves, sorted according to the values $|qq'|_G$. Moreover, each node u in this tree stores
   (a) the number of leaves in the subtree of u and
   (b) the sum of the values $|qb_j|_G$, where q ranges over all points in the substree of u.
2. Balanced binary search trees $\mathcal{T}'_{v,j}$ for $1 \leq j \leq \ell$. Each such tree $\mathcal{T}'_{v,j}$ stores the points q of $\{q \in S_v^b : q' = b_j\}$ at its leaves, sorted according to the values $|qq'|_G$. Each node u in this tree stores
   (a) the number of leaves in the subtree of u and
   (b) the sum of the values $|qb_j|_G$, where q ranges over all points in the substree of u.

Let us see how these trees can be used to obtain a 2-approximation of the summation in (6). Let p be a point in A_i^r and let q be a point in B_i^b.

1. Assume that $|pp'|_G \leq |qq'|_G$. By Lemma 1, $|pp'|_G + |p'q|_G$ is a 2-approximation of $|pq|_G$. Recall that we know the value $|pp'|_G$. If j is the index such that $p' = b_j$, then the value $|p'q|_G = |b_j q|_G$ is stored in the tree $\mathcal{T}_{v,j}$.
2. Assume that $|pp'|_G > |qq'|_G$. By Lemma 1, $|qq'|_G + |q'p|_G$ is a 2-approximation of $|pq|_G$. We know the value $|q'p|_G$. If j' is the index such that $q' = b_{j'}$, then the value $|qq'|_G = |qb_{j'}|_G$ is stored in the tree $\mathcal{T}_{v,j'}$.

Based on this, we do the following, for each point p in A_i^r: Let j be the index such that $p' = b_j$. By searching in $\mathcal{T}_{v,j}$ with the value $|pp'|_G$, we compute, in $O(\log n)$ time,

1. the number N of points q in B_i^b for which $|pp'|_G \leq |qq'|_G$,
2. the summation
$$X = \sum_{q \in B_i^b, |pp'|_G \leq |qq'|_G} |p'q|_G,$$
3. the value $N|pp'|_G + X$.

Observe that
$$N|pp'|_G + X = \sum_{q \in B_i^b, |pp'|_G \leq |qq'|_G} (|pp'|_G + |p'q|_G).$$

Next, for all j' with $1 \leq j' \leq \ell$, by searching in the trees $\mathcal{T}'_{v,j'}$ with the value $|pp'|_G$, we compute, in $O(\ell \log n) = O(\sqrt{r} \log n)$ total time,

1. the numbers $N_{j'}$ of points q in $\{q \in B_i^b : q' = b_{j'}\}$ for which $|pp'|_G > |qq'|_G$,
2. the summations
$$X_{j'} = \sum_{q \in B_i^b, q' = b_{j'}, |pp'|_G > |qq'|_G} |qq'|_G,$$
3. the summation
$$\sum_{j'=1}^{\ell} (N_{j'}|pq'|_G + X_{j'}).$$

Observe that this last summation is equal to
$$\sum_{q \in B_i^b, |pp'|_G > |qq'|_G} (|pq'|_G + |q'q|_G).$$

Thus, for a fixed point p in A_i^r, we have computed, in $O(\sqrt{r} \log n)$ time, a 2-approximation of the summation $\sum_{q \in B_i^b} |pq|_G$. Therefore, in $O(|A_i^r|\sqrt{r} \log n)$ time, we have computed a 2-approximation of the summation in (6). (Recall that this assumes that we have the trees $\mathcal{T}_{v,j}$ and $\mathcal{T}'_{v,j}$ for all j with $1 \leq j \leq \ell$.)

By traversing the split tree $T(S)$ in post-order, as we did in Step 3 of the algorithm in Section 3, we obtain 2-approximations of the summations in (6), for all i with $1 \leq i \leq m$, in total time which is the sum of

1. $O(\sqrt{r} n \log^2 n)$: This is the total time to compute all binary search trees $\mathcal{T}_{v,j}$ and $\mathcal{T}'_{v,j}$.
2. $O(\sqrt{r} n \log^2 n)$: This is the total time to search in all these binary search trees.

Recall that we repeat this algorithm for each of the $O(n/r)$ regions R in the r-division. It follows that the total time used to compute a $(2+\epsilon)$-approximation of $SSF_2(G)$ is within the time bound in (5). This completes the proof of Theorem 6.

5.2 Approximating $SSF(G)$ within $4+\epsilon$

In this section, we improve the running time in Theorem 6, while increasing the approximation factor to $4+\epsilon$. Since the algorithm is recursive, a generic call solves a more general problem.

Let R be a subset of S, which we can think of to be a region in a division of the graph G. Recall that a point p of R is an interior point, if for every edge (p, q) in G, the point q is also in R. All other points of R are boundary points. We denote the sets of interior and boundary

points of R by $int(R)$ and ∂R, respectively. The subgraph of G that is induced by R is denoted by $G[R]$.

The input for the algorithm consists of a subset R of S such that $|int(R)| = r$ and $|\partial R| = O(\sqrt{r})$. The output will be a real number that is between $SSF(R)/(4+\epsilon)$ and $(4+\epsilon)SSF(R)$, where

$$SSF(R) = \sum_{\{p,q\} \in \mathcal{P}_2(int(R))} \frac{|pq|_G}{|pq|}.$$

By running this algorithm with $R = S$ (in which case $int(R) = S$ and $\partial R = \emptyset$), we obtain a $(4+\epsilon)$-approximation of $SSF(G)$.

We assume that the entire graph G has been preprocessed using the algorithm of Arikati *et al.* [2]. Recall that this preprocessing takes $O(n^{3/2})$ time, after which, for any two points p and q, a 2-approximation of $|pq|_G$ can be computed in $O(\log n)$ time.

If r is less than some constant, we use the data structure of [2] to compute a 2-approximation of $SSF(R)$ in $O(\log n)$ time. For a large value of r, the algorithm does the following.

Let $r' = r/2$. Use the algorithm of Frederickson [5] to compute an r'-division of the graph $G[R]$. Since $G[R]$ has size $O(r)$, this takes $O(r \log r)$ time and produces $k = O(r/r') = O(1)$ regions, each one having size at most $r' = r/2$ and $O(\sqrt{r'}) = O(\sqrt{r})$ boundary points. Thus, the total number of boundary points in the r'-division is $O(\sqrt{r})$.

The r'-division partitions the pairs in $\mathcal{P}_2(int(R))$ into three groups:

1. The pair $\{p, q\}$ is of *type 0*, if at least one of p and q is a boundary point of some region.
2. The pair $\{p, q\}$ is of *type 1*, if p and q are interior points of the same region.
3. The pair $\{p, q\}$ is of *type 2*, if p and q are interior points of different regions.

For $j \in \{0, 1, 2\}$, we define

$$SSF_j(R) = \sum_{\{p,q\} \text{ of type } j} \frac{|pq|_G}{|pq|},$$

so that

$$SSF(R) = SSF_0(R) + SSF_1(R) + SSF_2(R).$$

We obtain a 2-approximation of $SSF_0(R)$ by querying the data structure of [2] with each pair of type 0. Since the number of such pairs is $O(r^{3/2})$, this takes time

$$O\left(r^{3/2} \log n\right). \tag{7}$$

We obtain a $(4+\epsilon)$-approximation of $SSF_1(R)$, by running the algorithm recursively on each region in the r'-division. Recall that k denotes the number of regions in the r'-division. For $1 \le i \le k$, we denote by r_i the number of interior points of the i-th region and by $\mathcal{T}(r_i)$ the running time of the recursive call on the i-th region. Then, the total time to approximate $SSF_1(R)$ is

$$O(r) + \sum_{i=1}^{k} \mathcal{T}(r_i). \tag{8}$$

Observe that $r_1 + \ldots + r_k \le r$ and each value r_i is at most $r/2$.

It remains to show how to approximate $SSF_2(R)$. We first do the following for each region R' in the r'-division. Consider the graph $G[R']$. We add a dummy vertex and connect it by an edge to every point of $\partial R'$; each such edge gets weight, say, one. Then we run Dijkstra's algorithm on the resulting graph with the source being the dummy vertex. This gives, for

each point p in $int(R')$ a point p' of $\partial R'$ such that $|pp'|_{G[R']}$ is minimum, together with the shortest-path length $|pp'|_{G[R']}$. Observe that $|pp'|_{G[R']} = |pp'|_G$. Since the number of regions R' is $O(1)$ and each one has size $O(r)$, this part of the algorithm takes $O(r \log r)$ time.

The value of $SSF_2(R)$ is approximated, by doing the following for each pair R', R'' of distinct regions in the r'-division. We compute the split tree and the corresponding WSPD $\{A_1, B_1\}$, $\ldots$, $\{A_m, B_m\}$ of Theorem 1 for the point set $int(R') \cup int(R'')$, with separation ratio $s = 16/\epsilon$. We color the points of $int(R')$ and $int(R'')$ red and blue, respectively. For each pair $\{A_i, B_i\}$, we define A_i^r, A_i^b, B_i^r, and B_i^b as in Section 5.1. As before, we want to approximate the values

$$\sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G \text{ and } \sum_{p \in B_i^r} \sum_{q \in A_i^b} |pq|_G,$$

for all i with $1 \leq i \leq m$.

Recall that, during the approximation of $SSF_0(R)$, we have computed a 2-approximation $\delta(b, p)$ of $|bp|_G$, for each b in $\partial R' \cup \partial R''$ and each p in $int(R') \cup int(R'')$. Consider a point p in $int(R')$ and a point q in $int(R'')$. Since any path in G between p and q passes through a point of $\partial R' \cup \partial R''$, we can use Lemma 1 to approximate $|pq|_G$:

1. If $|pp'|_G \leq |qq'|_G$, then $|pp'|_G + \delta(p', q)$ is a 4-approximation of $|pq|_G$.
2. If $|pp'|_G > |qq'|_G$, then $|qq'|_G + \delta(q', p)$ is a 4-approximation of $|pq|_G$.

Let $b_1, \ldots, b_\ell$ be the elements of $\partial R' \cup \partial R''$. We now use the binary search trees $\mathcal{T}_{v,j}$ and $\mathcal{T}'_{v,j}$ as in Section 5.1. The only difference is that each node u in any of these trees stores the sum of the values $\delta(b_j, q)$, where q ranges over all points in the substree of u. By using the same algorithm as in Section 5.1, we obtain 4-approximations of the summations $\sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G$ and $\sum_{p \in B_i^r} \sum_{q \in A_i^b} |pq|_G$ in total time $O(r^{3/2} \log^2 r)$.

Thus, since the number of pairs of distinct regions is $O(1)$, the total time for computing a $(4+\epsilon)$-approximation of $SSF_2(R)$ is

$$O\left(r^{3/2} \log^2 r\right). \tag{9}$$

If we denote the total running time of the algorithm by $\mathcal{T}(r)$, then it follows from (7), (8), and (9) that

$$\mathcal{T}(r) = O\left(r^{3/2}(\log n + \log^2 r)\right) + \sum_{i=1}^{k} \mathcal{T}(r_i).$$

Recall that $r_1 + \ldots + r_k \leq r$ and each value r_i is at most $r/2$. A straightforward inductive proof shows that

$$\mathcal{T}(r) = O\left(r^{3/2}(\log n + \log^2 r)\right).$$

As mentioned before, we obtain a $(4+\epsilon)$-approximation of $SSF(G)$, by running this algorithm with $R = S$. We have proved the following result:

Theorem 7 *Let G be a plane graph on n points in $\mathbb{R}^d$ and let $\epsilon > 0$ be a real constant. In $O(n^{3/2} \log^2 n)$ time, we can compute a real number that lies between $SSF(G)/(4+\epsilon)$ and $(4+\epsilon)SSF(G)$.*

References

[1] P. K. Agarwal, R. Klein, C. Knauer, S. Langerman, P. Morin, M. Sharir, and M. Soss. Computing the detour and spanning ratio of paths, trees, and cycles in 2D and 3D. *Discrete & Computational Geometry*, 39:17–37, 2008.

[2] S. Arikati, D. Z. Chen, L. P. Chew, G. Das, M. Smid, and C. D. Zaroliagis. Planar spanners and approximate shortest path queries among obstacles in the plane. In *ESA*, volume 1136 of *LNCS*, pages 514–528. Springer-Verlag, 1996.

[3] 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.

[4] M. Farshi, P. Giannopoulos and J. Gudmundsson. Improving the stretch factor of a geometric graph by edge augmentation. *SIAM Journal on Computing*, 38:226–240, 2008.

[5] G. N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM Journal on Computing*, 16:1004–1022, 1987.

[6] R. Klein, C. Knauer, G. Narasimhan, and M. Smid. On the dilation spectrum of paths, cycles, and trees. *Computational Geometry: Theory and Applications*, 42:923–933, 2009.

[7] G. Narasimhan and M. Smid. Approximating the stretch factor of Euclidean graphs. *SIAM Journal on Computing*, 30:978–989, 2000.

[8] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge University Press, Cambridge, UK, 2007.

[9] C. Wulff-Nilsen. Wiener index and diameter of a planar graph in subquadratic time. In *EuroCG*, pages 25–28, 2009.