APPROXIMATING THE AVERAGE STRETCH FACTOR OF GEOMETRIC GRAPHS*

Siu-Wing Cheng,† *Christian Knauer,*‡ *Stefan Langerman,*§ *and 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 average of the $\binom{n}{2}$ stretch factors determined by all pairs of points in S. We show that for paths, cycles, and trees, this average can be approximated, within a factor of $1+\epsilon$, in $O(n \cdot \mathit{polylog}(n))$ time. For plane graphs in $\mathbb{R}^2$, we present a $(2+\epsilon)$-approximation algorithm with running time $O(n^{5/3} \cdot \mathit{polylog}(n))$, and a $(4+\epsilon)$-approximation algorithm with running time $O(n^{3/2} \cdot \mathit{polylog}(n))$. Finally, we show that, for any tree in $\mathbb{R}^2$, the exact average of the squares of the $\binom{n}{2}$ stretch factors can be computed in $O(n^{1.8335})$ time.

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 [12].

For the related problem of computing, or approximating, the stretch factor of a given geometric graph, much less is known. Narasimhan and Smid [11] 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; they also show that

*SWC was supported by the Research Grant Council, Hong Kong, China (project no. 612107). MS was supported by NSERC. A preliminary version of this paper appeared in the Proceedings of the 21st Annual International Symposium on Algorithms and Computation (ISAAC), Part I, Lecture Notes in Computer Science, Vol. 6506, Springer-Verlag, Berlin, 2010, pp. 37–48.

†*Department of Computer Science and Engineering, HKUST, Hong Kong*, scheng@cse.ust.hk

‡*Institute of Computer Science, Universität Bayreuth, Germany*, christian.knauer@uni-bayreuth.de

§*Maître de Recherches du F.R.S.-FNRS. Département d'Informatique, Université Libre de Bruxelles*, stefan.langerman@ulb.ac.be

¶*School of Computer Science, Carleton University, Ottawa, Canada*, michiel@scs.carleton.ca

such a sequence of queries can be answered efficiently for paths, cycles, trees, and plane graphs. Gao and Zhang [8] present a fast algorithm for answering such a sequence of queries for unit-disk graphs; thus they obtain a fast algorithm for approximating the stretch factor of such a graph.

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.* [10] 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; related problems have been considered (see, e.g., Farshi *et al.* [6]). 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, which is an interesting quantity associated with 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 [14] 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 [5] can be used to reduce the problem of approximating $SSF(G)$ to that of approximating $O(n)$ different sums of shortest-path distances $\sum_{p \in A_i} \sum_{q \in B_i} |pq|_G$, with one such double-summation for each pair $\{A_i, B_i\}$ in the WSPD. Note that this is different from the general approach of Narasimhan and Smid [11] for approximating the (worst) stretch factor: For that problem, it is sufficient to take, for each of the $O(n)$ pairs $\{A_i, B)\}$, one point p in A_i and one point q in B_i, and approximate the shortest-path distance $|pq|_G$.

In Section 3, we apply the general approach of Section 2 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. 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.

As mentioned above, it is not known whether the exact value of $SSF(G)$ can be computed in subquadratic time, even for a simple graph G such as a path. In Section 6, we consider the related problem of computing the exact sum of all *squared* stretch factors, i.e., the quantity

$$SSF^{(2)}(G) = \sum_{\{p,q\} \in \mathcal{P}_2(S)} \left(\frac{|pq|_G}{|pq|} \right)^2.$$

Thus, the value $SSF^{(2)}(G)/\binom{n}{2}$ is equal to the *average* squared stretch factor of the graph G. For trees in $\mathbb{R}^2$, we use an appoach based on arithmetization and fast polynomial multipoint-evaluation, inspired by a technique developed by Ajwani *et al.* [2], to show that the exact value of $SSF^{(2)}(G)$ can be computed in $O(n^{1.8335})$ time.

2 The General Approach using Well-Separated Pairs

Our algorithms are based on Callahan and Kosaraju's well-separated pair decomposition; see [5]. We start by reviewing this decomposition and some of the relevant properties that will be used in the rest of this paper.

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 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. Observe that the height of $T(S)$ can be as large as $\Omega(n)$.

Given a separation ratio $s > 0$, Callahan and Kosaraju showed that the split tree $T(S)$ can be used to compute a WSPD of S, consisting of $m = O(n)$ pairs, where each subset A_i (and, similarly, each subset B_i) corresponds to a node v of the split-tree, in the sense that A_i equals the set S_v of all points that are stored at the leaves of the subtree rooted at v. Callahan [4, Section 4.5] showed that, for this particular WSPD, $\sum_{i=1}^{m} \min(|A_i|, |B_i|) = O(n \log n)$.

Theorem 1 ([4, 5]). *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 and using $O(n)$ space, 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*

1. *$m = O(n)$ and*
2. *$\sum_{i=1}^{m} \min(|A_i|, |B_i|) = O(n \log n)$.*

Consider a connected graph G with vertex set S, the split tree $T(S)$, and the corresponding WSPD $\{A_1, B_1\}, \ldots, \{A_m, B_m\}$ of Theorem 1. It follows from the definition of the WSPD that

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

By (1), all distances $|pq|$, for any p in A_i and any q in B_i, are within a factor of $1 + 4/s$ of each other. Therefore, for each i with $1 \leq i \leq m$, we choose an arbitrary point x_i in A_i and an arbitrary point y_i in B_i, and consider the summation

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

Using (1), it follows that

$$\frac{1}{1 + 4/s} \leq \frac{SSF'(G)}{SSF(G)} \leq 1 + 4/s.$$

Thus, by choosing $s = 4/\epsilon$, the value $SSF'(G)$ approximates the sum of all stretch factors within a factor of $1 + \epsilon$.

In order to compute $SSF'(G)$, we need to compute the values

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

for all i with $1 \leq i \leq m$. In Section 3, we will show that by traversing the split tree in post-order, these values can be computed efficiently for the cases when G is a path or a cycle. In Sections 4 and 5, we will follow a similar, but slightly different, approach for the cases when G is a tree or a plane graph.

3 Approximating *SSF* for 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 we have

$$\sum_{j=1}^{k} |pb_j|_G = k|p_1p|_G - \sum_{j=1}^{k} |p_1b_j|_G$$

and

$$\sum_{j=1}^{k'} |pb'_j|_G = \sum_{j=1}^{k'} |p_1b'_j|_G - k'|p_1p|_G.$$

It follows that

$$\sum_{q \in B_i} |pq|_G = (k - k')|p_1p|_G + \sum_{j=1}^{k'} |p_1b'_j|_G - \sum_{j=1}^{k} |p_1b_j|_G. \tag{3}$$

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

1. the number of points stored in the subtree of u and
2. the sum of the path lengths $|p_1q|_G$, where q ranges over all points stored in the subtree of u.

Then by searching in $\mathcal{T}_v$ for p (more precisely, for the index of p in G), we obtain, in $O(\log |B_i|) = O(\log n)$ time,

1. a sequence of $O(\log n)$ *canonical nodes* in $\mathcal{T}_v$ whose subtrees partition the set of all points in B_i that are to the left of p in G, and
2. a sequence of $O(\log n)$ canonical nodes in $\mathcal{T}_v$ whose subtrees partition the set 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, without loss of generality, 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 $|p_1p_i|_G$.

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. (a) If v is a leaf of $T(S)$, then initialize $\mathcal{T}_v$ such that it contains only the point stored at v.
   (b) Otherwise, let v_1 and v_2 be the two children of v.
      i. 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 $\mathcal{T}_v$.
      ii. 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).

The correctness of this algorithm follows from the discussion above. By Theorem 1, Step 1 takes $O(n \log n)$ time. Step 2 can be done in $O(n)$ time. Consider Step 3. The total time spent in computing the trees $\mathcal{T}_v$ is equal to $O(\log n)$ times the total number of insertions that take place. Since the algorithm always inserts the elements of the smaller of $\mathcal{T}_{v_1}$ and $\mathcal{T}_{v_2}$ into the larger of these two, each point is inserted $O(\log n)$ times. Therefore, the total time for computing the trees $\mathcal{T}_v$ is $O(n \log^2 n)$. Finally, the total time for searching in the trees $\mathcal{T}_v$ is $\sum_{i=1}^{m} |A_i| \log n$, which, by Theorem 1, is $O(n \log^2 n)$. We have proved the following result:

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 and using $O(n)$ space, we can compute a real number that lies between $SSF(G)/(1+\epsilon)$ and $(1+\epsilon)SSF(G)$.*

Remark 1. *In Section 4, we will present an algorithm that approximates $SSF(G)$ for an arbitrary tree G in $O(n \log^2 n)$ time. If G is a path, then this algorithm is slightly different from the one presented above. In fact, we will even show in Section 4 that for any tree G, $SSF(G)$ can be approximated in $O(n \log^2 n / \log\log n)$ time; thus, this will improve Theorem 2.*

We now consider the case when the graph G is a cycle $(p_1, p_2, \ldots, p_n, p_1)$ on the points of the set S. We will refer to the ordered sequence $p_1, \ldots, p_n$ as the *clockwise* order of the points.

For two points p and q of S, we denote by $G[p, q]$ the path that starts at p, follows the cycle G in clockwise order, and ends at q. The length of this path is denoted by $|G[p, q]|$.

Let L be the total length of all edges on the cycle. For each point p of S, define $\nu(p)$ to be the last point of G, in clockwise order from p, for which $|p\nu(p)| \leq L/2$.

Consider a pair $\{A_i, B_i\}$ of the WSPD and let p be an arbitrary point in A_i. Assume that p is contained in $G[p_1, \nu(p)]$. (The case when $\nu(p)$ is contained in $G[p_1, p]$ can be handled in a similar way.) The points p and $\nu(p)$ partition the set B_i into three subsets (some of which may be empty):

1. the points $b_1, \ldots, b_k$ that are on the path $G[p_1, p]$,

2. the points $b'_1, \ldots, b'_{k'}$ that are on the path $G[p, \nu(p)]$, and
3. the points $b''_1, \ldots, b''_{k''}$ that are on the path $G[\nu'(p), p_n]$, where $\nu'(p)$ is the clockwise neighbor of $\nu(p)$ on the cycle G.

We have

$$\begin{aligned}\sum_{q \in B_i} |pq|_G &= k|G[p_1, p]| - \sum_{j=1}^{k} |G[p_1, b_j]| \\ &\quad + \sum_{j=1}^{k'} |G[p_1, b'_j]| - k'|G[p_1, p]| \\ &\quad + \sum_{j=1}^{k''} (L - |G[p_1, b''_j]|) + k''|G[p_1, p]|.\end{aligned}$$

Let v be the node in the split tree such that $B_i = S_v$. By using the same balanced binary search tree $\mathcal{T}_v$ that we used before, and by searching in $\mathcal{T}_v$ with the indices (in G) of the points p and $\nu(p)$, we can compute the value of $\sum_{q \in B_i} |pq|_G$ in $O(\log n)$ time. Thus, by using a slight modification of the algorithm in Section 3, we obtain 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 and using $O(n)$ space, we can compute a real number that lies between $SSF(G)/(1+\epsilon)$ and $(1+\epsilon)SSF(G)$.*

4 Approximating *SSF* for Trees

Let S be a set of n points in $\mathbb{R}^d$ and let G be a spanning tree of S. In this section, we present a divide-and-conquer algorithm that approximates the value $SSF(G)$.

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 S_1 and S_2 be the vertex sets of G'_1 and G'_2, respectively. Thus, $S = S_1 \cup S_2 \cup \{c\}$. 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

$$SSF(G) = SSF(G_1) + SSF(G_2) + \sum_{p \in S_1} \sum_{q \in S_2} \frac{|pq|_G}{|pq|}. \tag{4}$$

We will show that the double-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. (The base case is when $n = 2$; in this case, $SSF(G) = 1$.)

We color each point of S_1 *red* and each point of S_2 *blue*. The centroid c does not get a color. Consider the split tree $T(S)$ and the corresponding WSPD $\{A_1, B_1\}, \ldots, \{A_m, B_m\}$

of Theorem 1, with separation ratio $s = 4/\epsilon$. 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. The double-summation in (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 with $1 \leq i \leq m$, let x_i be an arbitrary point in A_i and let y_i an arbitrary point in B_i. Then, the summation

$$\sum_{i=1}^{m} \left(\frac{1}{|x_i y_i|} \left(\sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G + \sum_{p \in B_i^r} \sum_{q \in A_i^b} |pq|_G \right) \right) \tag{5}$$

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

$$\begin{aligned} \sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G &= \sum_{p \in A_i^r} \sum_{q \in B_i^b} (|pc|_G + |cq|_G) \\ &= |B_i^b| \sum_{p \in A_i^r} |pc|_G + |A_i^r| \sum_{q \in B_i^b} |cq|_G. \end{aligned}$$

By a symmetric argument, we get

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

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

1. Traverse the tree G in postorder (assuming it is rooted at the centroid c) and store with each point p the path length $|pc|_G$.
2. 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.
3. For each leaf v of the split tree $T(S)$, do the following: Let p be the point stored at v.
   (a) If p is red, then set $redsum(v) = |pc|_G$ and $bluesum(v) = 0$.
   (b) If p is blue, then set $redsum(v) = 0$ and $bluesum(v) = |pc|_G$.
   (c) If p is the centroid, then set $redsum(v) = 0$ and $bluesum(v) = 0$.
4. Traverse the split tree $T(S)$ in postorder. For each internal node v, with children v_1 and v_2, set $redsum(v) = redsum(v_1) + redsum(v_2)$ and $bluesum(v) = bluesum(v_1) + bluesum(v_2)$.

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 $redsum(v)$ and $bluesum(v)$ 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 $\sum_{q\in B_i^b} |cq|_G$.

Thus, after having computed the split tree and the WSPD (which takes $O(n \log n)$ time), we obtain an approximation of the double-summation in (4) in $O(n)$ time. We have proved the following result:

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 and using $O(n)$ space, 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'. As we have seen above, 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 (which was computed during the call at level zero of the recursion tree). 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

$$O\left(\left(n \log n + 2^h n\right) \frac{\log n}{h}\right).$$

For $h = \log \log n$, this quantity is $O(n \log^2 n / \log \log n)$. Thus, we have proved 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 and using $O(n)$ space, we can compute a real number that lies between $SSF(G)/(1+\epsilon)$ and $(1+\epsilon)SSF(G)$.*

5 Approximating *SSF* for Plane Graphs

In this section, G denotes a plane connected graph whose vertex set is a set S of n points in $\mathbb{R}^2$.

In Section 5.2, we will show that $SSF(G)$ can be approximated within a factor of $2+\epsilon$ in $O(n^{5/3} \cdot polylog(n))$ time. In Section 5.3, we will obtain a $(4+\epsilon)$-approximation in $O(n^{3/2} \cdot polylog(n))$ time. We start by reviewing separators and Frederickson's r-divisions (see [7]).

5.1 Separators and r-Divisions

Let C be a *separator* of the graph 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.* [3]. For completeness, we include a proof.

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

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

Proof. First observe that the shortest path in G between p and q must visit some point in the separator C. Let c be such a point. By the definition of p', we have $|pp'|_G \leq |pc|_G$ and, similarly, $|qq'|_G \leq |qc|_G$. By combining these inequalities with the triangle inequality and the assumption that $|pp'|_G \leq |qq'|_G$, we obtain

$$\begin{aligned} |pp'|_G + |p'q|_G &\leq |pc|_G + (|p'p|_G + |pc|_G + |cq|_G) \\ &\leq |pc|_G + (|q'q|_G + |pc|_G + |cq|_G) \\ &\leq |pc|_G + (|qc|_G + |pc|_G + |cq|_G) \\ &= 2|pq|_G. \end{aligned}$$

□

The following notions were introduced by Frederickson [7]. A *division* of G is a sequence $R_1, \ldots, R_k$ of subsets of S, for some $k \geq 1$, such that

1. $\cup_{i=1}^{k} R_i = S$ and
2. for each i with $1 \leq i \leq k$ and for each p in R_i,
   (a) 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
   (b) or p is a *boundary* point of R_i, i.e., p is contained in at least one other subset in the sequence.

Each subset R_i in the sequence is called a *region*.

Let r be an integer with $1 \leq r \leq n$. A division $R_1, \ldots, R_k$ of G is called an *r-division*, if

1. $k = O(n/r)$ and
2. for each i with $1 \leq i \leq k$, the region R_i contains at most r points and $O(\sqrt{r})$ boundary points.

Frederickson [7] has shown that such an r-division can be computed in $O(n \log n)$ time, using $O(n)$ space. Observe that the total number of boundary points in an r-division is $k \cdot O(\sqrt{r}) = O(n/\sqrt{r})$. Also, for any i with $1 \leq i \leq k$, the boundary points of R_i form a separator of the graph G.

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

We choose an integer r with $1 \leq r \leq n$; the precise value of r will be determined later. We compute, in $O(n \log n)$ time, an r-division $R_1, \ldots, R_k$ of G. This r-division partitions the pairs $\{p, q\}$ in $\mathcal{P}_2(S)$ into two groups:

1. The pair $\{p, q\}$ is of *type 1*, if p and q belong to the same region.
2. The pair $\{p, q\}$ is of *type 2*, if p and q belong to different regions.

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 Arikati *et al.* [3], we preprocess the graph G in $O(n^{3/2})$ time and using $O(n^{3/2})$ space, 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 time

$$O\left(n^{3/2} + rn \log n\right),$$

while using $O(n^{3/2})$ space.

In the rest of this section, we will show how to compute a $(2 + \epsilon)$-approximation of $SSF_2(G)$ in time

$$O\left(\frac{n^2 \log^2 n}{\sqrt{r}}\right), \tag{6}$$

while using

$$O\left(\frac{n^2}{\sqrt{r}} + \sqrt{r}n\right)$$

space. By adding the approximations of $SSF_1(G)$ and $SSF_2(G)$, and 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 $\mathbb{R}^2$ and let $\epsilon > 0$ be a real constant. In $O((n \log n)^{5/3})$ time and using $O(n^{5/3}/(\log n)^{1/3})$ space, 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. Since there are $O(n/\sqrt{r})$ boundary points, and since Dijkstra's algorithm takes $O(n \log n)$ time, this part of the algorithm takes $O((n^2 \log n)/\sqrt{r})$ time, which is within the time bound in (6). This part of the algorithm uses $O(n^2/\sqrt{r})$ space.

Next, 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 $s = 8/\epsilon$. (In earlier sections, we used the separation ratio $s = 4/\epsilon$. In the current section, we compute 2-approximations of shortest-path lengths. Because of this, we have to double the value of s.) Assume, without loss of generality, 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 (see (5)), in order to obtain a $(2+\epsilon)$-approximation of $SSF_2(G)$, it is sufficient to compute a 2-approximation of the values

$$\sum_{p \in A_i^r} \sum_{q \in B_i^b} |pq|_G \tag{7}$$

and

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

for all i with $1 \leq i \leq m$. We will show how a 2-approximation of the summation in (7) 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 (which uses $O(\ell n) = O(\sqrt{r} n)$ space):

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 subtree 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 subtree of u.

Let us see how these trees can be used to obtain a 2-approximation of the summation in (7). Let p be a point in A_i^r. For any point q in B_i^b, we have $|pp'|_G \leq |qq'|_G$ or $|pp'|_G > |qq'|_G$.

1. Consider a point q in B_i^b for which $|pp'|_G \leq |qq'|_G$. 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 $|p'q|_G = |b_jq|_G$ is implicitly stored in the tree $\mathcal{T}_{v,j}$. In fact, we can obtain, from this tree, the sum of all values $|p'q|_G$, where q ranges over all points in B_i^b for which $|pp'|_G \leq |qq'|_G$.

2. Consider a point q in B_i^b for which $|pp'|_G > |qq'|_G$. 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 implicitly stored in the tree $\mathcal{T}'_{v,j'}$. Thus, we can obtain, from all these trees, the sum of all values $|qq'|_G$, where q ranges over all points in B_i^b for which $|pp'|_G > |qq'|_G$.

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 (7). (Recall that this assumes that we have the trees $\mathcal{T}_{v,j}$ and $\mathcal{T}'_{v,j}$ for all j with $1 \le j \le \ell$.)

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 (7), for all i with $1 \le i \le m$, 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 (6). This completes the proof of Theorem 6.

5.3 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 $SSF(int(R))/(4+\epsilon)$ and $(4+\epsilon)SSF(int(R))$, where

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

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.* [3]. Recall that this preprocessing takes $O(n^{3/2})$ time and uses $O(n^{3/2})$ space, 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 [3] to compute a 2-approximation of $SSF(int(R))$ 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 [7] to compute an r'-division of the graph $G[R]$. Since $G[R]$ has size $O(r)$, this takes

$$O(r \log r) \tag{8}$$

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

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

so that

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

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

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

We obtain a $(4+\epsilon)$-approximation of $SSF_1(int(R))$, 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 $SSF_1(int(R))$ is

$$O(r) + \sum_{i=1}^{k} \mathcal{T}(r_i). \tag{10}$$

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

It remains to show how to approximate $SSF_2(int(R))$. 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 and uses $O(r)$ space.

The value of $SSF_2(int(R))$ 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 5.2. As before, we want to approximate the values

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

and

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

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

Recall that, during the approximation of $SSF_0(int(R))$, 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 5.2. 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 subtree of u. By using the same algorithm as in Section 5.2, 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 $O(r^{3/2} \log^2 r)$, using $O(r^{3/2})$ space.

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

$$O\left(r^{3/2} \log^2 r\right). \tag{11}$$

If we denote the total running time of the algorithm by $\mathcal{T}(r)$, then it follows from (8), (9), (10), and (11) 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 $r/2$. Observe that

$$\begin{aligned}\sum_{i=1}^{k} r_i^{3/2}(\log n + \log^2 r_i) &= \sum_{i=1}^{k} r_i\sqrt{r_i}(\log n + \log^2 r_i) \\ &\leq \sum_{i=1}^{k} r_i\sqrt{r/2}(\log n + \log^2 r) \\ &\leq \frac{r^{3/2}}{\sqrt{2}}(\log n + \log^2 r)\end{aligned}$$

and $1/\sqrt{2} < 1$. Using this, a straightforward inductive proof shows that

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

The space used by the entire algorithm is $O(n^{3/2} + r^{3/2})$.

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 $\mathbb{R}^2$ and let $\epsilon > 0$ be a real constant. In $O(n^{3/2}\log^2 n)$ time and using $O(n^{3/2})$ space, we can compute a real number that lies between $SSF(G)/(4+\epsilon)$ and $(4+\epsilon)SSF(G)$.*

6 Computing the Average Squared Stretch Factor for Trees

Recall that $SSF^{(2)}(G)$ denotes the sum of the squares of all $\binom{n}{2}$ stretch factors, i.e.,

$$SSF^{(2)}(G) = \sum_{\{p,q\}\in\mathcal{P}_2(S)} \left(\frac{|pq|_G}{|pq|}\right)^2.$$

Thus, the value $SSF^{(2)}(G)/\binom{n}{2}$ is equal to the *average* squared stretch factor of the graph G.

In this section, we present a subquadratic algorithm that computes the exact value of $SSF^{(2)}(G)$ for the case when G is a tree in $\mathbb{R}^2$:

Theorem 8. *Let G be a tree on n points in $\mathbb{R}^2$. The value of $SSF^{(2)}(G)$ can be computed in $O(n^{1.8335})$ time.*

The proof of this theorem will be based on fast polynomial multipoint-evaluation. Before we present the proof in Section 6.2, we recall some relevant results.

6.1 Tools for Polynomial Multipoint-Evaluation

A *bi-variate polynomial* of degree at most k is a function $f(x, y)$ of the two variables x and y that can be written as

$$f(x,y) = \sum_{i=0}^{k} \sum_{j=0}^{k} c_{ij} x^i y^j,$$

where the coefficients a_{ij} are real numbers. We say that f is represented in *coefficient-form*, if all these $(k+1)^2$ coefficients are given.

Let $f_1, f_2, \ldots, f_m$ be a sequence of m bi-variate polynomials, where each polynomial has degree 2 and is given in coefficient-form. There exist two bi-variate polynomials F and G, both of degree $O(m)$, such that

$$\sum_{k=1}^{m} \frac{1}{f_k(x,y)} = \frac{F(x,y)}{G(x,y)}.$$

Lemma 2. The coefficient-form representations of the polynomials F and G can be computed in $O(m^2 \cdot \mathit{polylog}(m))$ time.

Proof. If m is less than some constant, we use a brute-force algorithm. Assume that m is a large integer. Also, assume for simplicity that m is even. The following divide-and-conquer algorithm computes the polynomials F and G:

1. By running the algorithm recursively on $f_1, \ldots, f_{m/2}$, we obtain coefficient-form representations of two bi-variate polynomials F_1 and G_1, both of degree $O(m)$, such that
$$\sum_{k=1}^{m/2} \frac{1}{f_k(x,y)} = \frac{F_1(x,y)}{G_1(x,y)}.$$
2. By running the algorithm recursively on $f_{1+m/2}, \ldots, f_m$, we obtain coefficient-form representations of two bi-variate polynomials F_2 and G_2, both of degree $O(m)$, such that
$$\sum_{k=1+m/2}^{m} \frac{1}{f_k(x,y)} = \frac{F_2(x,y)}{G_2(x,y)}.$$
3. Observe that $F = F_1G_2 + F_2G_1$ and $G = G_1G_2$. Using an algorithm of Nüsken and Ziegler [13][1], we obtain, in $O(m^2 \cdot \mathit{polylog}(m))$ time, the coefficient-form representations of F and G.

Thus, the entire divide-and-conquer algorithm takes $O(m^2 \cdot \mathit{polylog}(m))$ time. □

Define ω_2 to be the infimum of all real numbers w such that any $n \times n$ matrix can be multiplied by any $n \times n^2$ matrix in $O(n^{w+\epsilon})$ time, for any constant $\epsilon > 0$. It is known that $\omega_2 < 3.334$; see Huang and Pan [9].

The following lemma appears as Result 4 in Nüsken and Ziegler [13].

[1] See Question 2(i) on page 545 and its answer on page 546.

Lemma 3. Assume we are given the coefficient-form representation of the bi-variate polynomial F, whose degree is at most m, and a set V of n points in the plane. For any constant $\epsilon > 0$, the sequence of all values $F(p)$, where p ranges over all points in V, can be computed in total time

$$O\left(\left(n + m^2\right) m^{\omega_2/2-1+\epsilon}\right).$$

6.2 Proof of Theorem 8

Let S be a set of n points in $\mathbb{R}^2$ and let G be a tree with vertex set S. As in Section 4, let c be a centroid of G. Thus, by removing c, we obtain two forests G'_1 and G'_2, each one having at most $2n/3$ vertices. Recall that c can be computed in $O(n)$ time. Let S_1 and S_2 denote the vertex sets of G'_1 and G'_2, respectively, so that the vertex set S of G is equal to $S_1 \cup S_2 \cup \{c\}$. 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 will apply the divide-and-conquer technique to compute the value $SSF^{(2)}(G)$. To this end we first observe that

$$SSF^{(2)}(G) = SSF^{(2)}(G_1) + SSF^{(2)}(G_2) + \sum_{p\in S_1}\sum_{q\in S_2}\left(\frac{|pq|_G}{|pq|}\right)^2.$$

The values $SSF^{(2)}(G_1)$ and $SSF^{(2)}(G_2)$ will be computed recursively. Below, we will show how to compute

$$SSF^{(2)}(G, S_1, S_2) = \sum_{p\in S_1}\sum_{q\in S_2}\left(\frac{|pq|_G}{|pq|}\right)^2$$

in $O(n^{1.8335})$ time. From this, it will follow that the entire divide-and-conquer algorithm has running time $O(n^{1.8335})$.

For any p in S_1, we define

$$f(p) = \sum_{q\in S_2}\left(\frac{|pq|_G}{|pq|}\right)^2.$$

Thus,

$$SSF^{(2)}(G, S_1, S_2) = \sum_{p\in S_1} f(p).$$

We now show how to compute the values $f(p)$ for all $p \in S_1$ *simultaneously* in $O(n^{1.8335})$ time. Adding up those $O(n)$ many values will give us $SSF^{(2)}(G, S_1, S_2)$.

For any vertex v of G, let ℓ_v denote the length of the path in the tree G between v and the centroid c. Observe that all values ℓ_v, where v ranges over all vertices of G, can be computed in $O(n)$ total time.

If p is a vertex of S_1 and q is a vertex of S_2, then $|pq|_G = \ell_p + \ell_q$. It follows that

$$f(p) = \sum_{q\in S_2}\left(\frac{\ell_p + \ell_q}{|pq|}\right)^2 = \ell_p^2 \sum_{q\in S_2}\frac{1}{|pq|^2} + 2\ell_p \sum_{q\in S_2}\frac{\ell_q}{|pq|^2} + \sum_{q\in S_2}\frac{\ell_q^2}{|pq|^2}.$$

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

$$f^{(i)}(p) = \sum_{q \in S_2} \frac{\ell_q^i}{|pq|^2}.$$

Thus,

$$f(p) = \ell_p^2 \cdot f^{(0)}(p) + 2\ell_p \cdot f^{(1)}(p) + f^{(2)}(p).$$

We partition S_2 arbitrarily into $N = \Theta(\sqrt{n})$ subsets $S_2^1, \ldots, S_2^N$, each subset having size $\Theta(\sqrt{n})$, and define, for $j \in \{1, 2, \ldots, N\}$ and $i \in \{0, 1, 2\}$,

$$f_j^{(i)}(p) = \sum_{q \in S_2^j} \frac{\ell_q^i}{|pq|^2}.$$

Then we have

$$f^{(i)}(p) = f_1^{(i)}(p) + f_2^{(i)}(p) + \ldots + f_N^{(i)}(p).$$

Consider the coordinates (x, y) of the vertex p as *symbolic variables* and interpret $f_1^{(i)}, \ldots, f_N^{(i)}$ as bi-variate rational functions in those variables.

Let j be an integer with $1 \leq j \leq N$, let $i \in \{0, 1, 2\}$, and consider $f_j^{(i)}$. By Lemma 2, we can use the summands $\ell_q^i/|pq|^2$ to compute, in $O(n \cdot \mathit{polylog}(n))$ time, the coefficient representations of two bi-variate polynomials $n_j^{(i)}(p)$ and $d_j^{(i)}(p)$ such that

$$f_j^{(i)}(p) = \frac{n_j^{(i)}(p)}{d_j^{(i)}(p)}.$$

Both of these polynomials have degree $O(\sqrt{n})$ and, consequently, $O(n)$ coefficients. By Lemma 3, we can evaluate both polynomials $n_j^{(i)}(p)$ and $d_j^{(i)}(p)$ at all points p of S_1 simultaneously in

$$O\left(n\left(\sqrt{n}\right)^{\omega_2/2-1+\epsilon}\right) = O\left(n^{1/2+\omega_2/4+\epsilon/2}\right)$$

time.

Thus, by doing this for all $N = \Theta(\sqrt{n})$ values of j, we obtain $SSF^{(2)}(G, S_1, S_2)$ in

$$O\left(n^{1+\omega_2/4+\epsilon/2}\right)$$

time. Since $\omega_2 < 3.334$, we can take ϵ to be sufficiently small such that the running time is $O(n^{1.8335})$.

7 Concluding Remarks

We have presented a general approach, based on well-separated pairs, for approximating the average stretch factor of geometric graphs. For paths, cycles, trees, and plane graphs, we obtained subquadratic algorithms. We leave as an open problem to improve the running time of our algorithms for paths, cycles, and trees. For plane graphs, it would be interesting

to improve the running time or the approximation factor. Moreover, we leave as an open problem to obtain subquadratic approximation algorithms for other classes of geometric graphs.

Gao and Zhang [8] present a fast algorithm for computing a WSPD for any set of points, where distances are measured in the unit-disk graph shortest-path metric. Using this WSPD, they show that approximate shortest-path queries in unit-disk graphs can be answered efficiently. We leave as an open problem to combine their results with those in the present paper to approximate the average stretch factor of a unit-disk graph in subquadratic time.

All our results for computing the average stretch factor are approximate. Can the exact average stretch factor be computed in subquadratic time, for simple classes of graphs such as geometric paths?

Using fast polynomial multi-point evaluation, we showed that the exact average squared stretch factor of a tree in $\mathbb{R}^2$ can be computed in subquadratic time. Can this result be generalized to higher dimensions? Can a similar result be obtained for other classes of geometric graphs?

Acknowledgements

We thank the participants of the 2009 Korean Workshop on Computational Geometry (held in Kanazawa and Ishikawa, Japan) for fruitful discussions. We thank Antoine Vigneron for pointing us to reference [13]. We thank the referees for their useful comments.

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] D. Ajwani, S. Ray, R. Seidel, and H. R. Tiwary. On computing the centroid of the vertices of an arrangement and related problems. In *Proceedings of the 10th Workshop on Algorithms and Data Structures*, volume 4619 of *Lecture Notes in Computer Science*, pages 519–528, Berlin, 2007. Springer-Verlag.

[3] 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 *Proceedings of the 4th European Symposium on Algorithms*, volume 1136 of *Lecture Notes in Computer Science*, pages 514–528, Berlin, 1996. Springer-Verlag.

[4] P. B. Callahan. *Dealing With Higher Dimensions: The Well-Separated Pair Decomposition and Its Applications*. Ph.D. thesis, Department of Computer Science, Johns Hopkins University, Baltimore, Maryland, 1995.

[5] 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.

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

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

[8] J. Gao and L. Zhang. Well-separated pair decomposition for the unit-disk graph metric and its applications. *SIAM Journal on Computing*, 35:151–169, 2005.

[9] X. Huang and V. Y. Pan. Fast rectangular matrix multiplication and applications. *Journal of Complexity*, 14:257–299, 1998.

[10] 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.

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

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

[13] M. Nüsken and M. Ziegler. Fast multipoint evaluation of bivariate polynomials. In *Proceedings of the 12th European Symposium on Algorithms*, volume 3221 of *Lecture Notes in Computer Science*, pages 544–555, Berlin, 2004. Springer-Verlag.

[14] C. Wulff-Nilsen. Wiener index and diameter of a planar graph in subquadratic time. In *Proceedings of the 25th European Workshop on Computational Geometry*, pages 25–28, 2009.