Approximation Algorithms for Planar Traveling Salesman Tours and Minimum-Length Triangulations

Kenneth L. Clarkson
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Abstract

This paper gives a partitioning scheme for the geometric, planar traveling salesman problem, under the Euclidean metric: given a set S of n points in the plane, find a shortest closed tour (path) visiting all the points. The scheme employs randomization, and gives a tour that can be expected to be short, if S satisfies the condition that a random subset $R \subset S$ has on average a tour much shorter than an optimal tour of S. This condition holds for points independently, identically distributed in the plane, for example, for which a tour within $1+\epsilon$ of shortest can be found in expected time $nk^2 2^k$, where $k = O(\log\log n)^3/\epsilon^2$. One algorithm employed in the scheme is of interest in its own right: when given a simple polygon P, it finds a Steiner triangulation of the interior of P. If P has n sides and perimeter L_P, the edges of the triangulation have total length $L_P O(\log n)$. If this algorithm is applied to a simple polygon induced by a minimum spanning tree of a point set, the result is a Steiner triangulation of the set with total length within a factor of $O(\log n)$ of the minimum possible.

1 Introduction

The Traveling Salesman Problem (TSP) needs no introduction (but see [LLKS85]). This paper gives an algorithm for the planar geometric case: given a set S of n points in the plane, find a shortest possible path that visits all the points. Even in this special case of the TSP, no known polynomial-time algorithm does better in general than that of Christofides, which finds tours within a factor of $3/2$ of the optimal length [Chr76]. In fact, unless $P = NP$, no approximation scheme can guarantee a tour within $1+\epsilon$ of shortest and run in $(n/\epsilon)^{O(1)}$ time [Pap77]. In this discouraging situation, Karp's approximation schemes are a pleasant surprise: if the points are identically, independently distributed (i.i.d.), one of Karp's methods find a tour almost surely within $1+\epsilon$ of shortest, and requires

$O(n \log n) + O(n)2^{1/\epsilon^2}/\epsilon^2$ time [Kar77, Ste81a, LLKS85]. Such probabilistic results give some idea of the usefulness of Karp's methods, but suffer from some restrictions placed on the probability distribution of the input.

One of Karp's methods, *fixed dissection*, requires the density function to be (approximately) known. Both fixed dissection and Karp's other approach, *adaptive dissection*, require the density function of the input points to be zero outside a bounded region. These limitations are needed for the approximation guarantees for the algorithms, and for the time bounds for fixed dissection. Also, the optimal tour must be of cost $\Omega(\sqrt{n})$ for the approximation guarantees; although this condition holds for independently identically distributed (i.i.d.) points under mild conditions [Ste81b], it is easy to describe families of point sets for which it does not (and for which the scheme given here gives provably good results).

The algorithm given here has an expected time bound holding for any point set, and can give a short tour when, roughly, a tour of a small random subset of S is expected to be much shorter than any tour of S. Such a condition holds for any i.i.d. point set for which the expected tour length $T(n)$ for n points satisfies $T(n) = \Omega(n^\alpha)$ for some $\alpha > 0$. As noted, this is true with $\alpha = 1/2$ for density functions satisfying mild conditions, so the approximation guarantees will be expressed under this assumption; the results can be easily generalized by replacing some constants by functions of α as appropriate.

A key result for the TSP approximation scheme is an algorithm for computing a low-length Steiner triangulation of a simple polygon. Given a simple polygon P with n sides and perimeter L_P, the algorithm returns a triangulation $\Delta^*(P)$ of the interior of P: a collection of edges that divide the interior of the polygon into triangles, so that every side of the polygon is an edge in the triangulation. The triangles meet only at common edges or vertices. The total number of triangles is $O(n)$. The length of the triangulation, the total of the lengths of the edges, is $L_P O(\log n)$. (Sometimes this total length is called the weight or cost.) The algorithm requires $O(n \log^2 n)$ time in the worst case.

This algorithm can be used to triangulate point sets as follows: given a point set S, build a simple polygon $P(S)$ by enclosing the minimum spanning tree of S by a small iso-oriented rectangle with one side containing a point of S (See Figure 1. Here the edges of the tree each give two edges of $P(S)$, and so it is quite degenerate in a sense.) The triangulation $\Delta^*(P(S))$ has a length within $O(\log n)$ of the length of the minimum spanning tree of S, and so is within $O(\log n)$ of a minimum length triangulation of S. (This holds whether the triangulation is Steiner or not, since a Steiner triangulation has at least $\sqrt{3}/2$ times the length of a minimum spanning tree [DH90].)

Plaisted and Hong [PH87] give an algorithm for (non-Steiner) triangulation of a point set, with a length within $O(\log n)$ of optimal. Recently Smith has sped up their algorithm, to a worst-case time bound of $O(n^2 \log n)$ [Smi89]. Thus the new algorithm is faster; also, it may produce much shorter triangulations in some cases. (Applications in finite element analysis and numerical interpolation

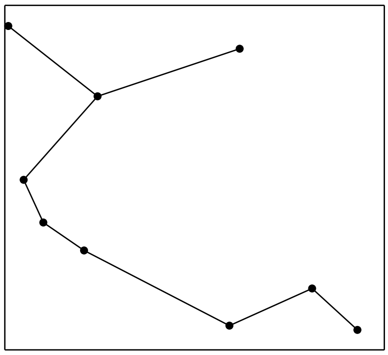

Figure 1: A simple polygon induced by an MST.

seem to require a non-Steiner triangulation, however.)

1.1 The rest of the paper.

The next section gives the main ideas for the basic algorithm, and then a more complex algorithm with a better time bound. Section 3 gives the algorithm for polygon triangulation. The final section gives a few concluding remarks.

2 The basic algorithm

The new algorithm uses Karp's observation that a planar subdivision of low length gives a correspondingly cheap tour. This fact is stated below, using the notation $T(S)$ for the length of an optimal tour of point set S, and L_C for the total length of a set of line segments C. We will view a triangulation Δ either as a collection of line segments, or as a collection of regions, as appropriate.

Theorem 1 *Suppose Δ is a triangulation of length L_Δ. Then for any point set S, the optimal tour length $T(S)$ satisfies*

$$T(S) + 3L_\Delta/2 \geq \sum_{\rho\in\Delta} T(\rho\cap S).$$

Also, there is a tour of length $T^*(S)$, where

$$T^*(S) \leq 3L_\Delta + \sum_{\rho\in\Delta} T(\rho\cap S),$$

obtainable in $O(n)$ time, given optimal tours in each $\rho\cap S$.

Proof. The proof is essentially the same as in [Kar77], and is omitted. The procedure for computing $T^*(S)$ is to use the triangulation and optimal subtours to obtain an Eulerian graph whose vertices include the points of S. A tour can then be obtained via “shortcuts.” □

Given this theorem, and the triangulation procedure for point sets discussed above, the following algorithm for the TSP suggests itself: given a set S of n points, and $\epsilon > 0$, find a random subset $R \subset S$ of size $\hat{r} = \epsilon^2 n / K \log^2 n$, where K is a constant. Using algorithm Δ^* of §3, compute a triangulation $\Delta^*(P(R))$, and then use the procedure of Theorem 1 to compute a tour.

How fast is this algorithm? Plainly $\Delta^*(P(R))$ can be computed in polynomial time; our main concern then is the time for computing the optimal subtours within each triangle in $\Delta^*(P(R))$. The following lemma will help in bounding this cost.

Lemma 2 *If $S \subset E^2$ is a set of n points in general position, and R is a random subset of S of size r, then with probability at least $1 - r^{O(1)} e^{-xr/n}$, every triangle in $\Delta^*(P(R))$ contains no more than x points of S.*

Proof. The lemma is similar to results in [Cla87]; every triangle in $\Delta^*(P(R))$ has vertices that are the intersection of lines that pass through points of S. That is, the triangles in $\Delta^*(P(R))$ are all in a set $\mathcal{F}_S$ of $O(n^{12})$ triangles, each triangle defined by $b \leq 12$ points of S. The probability that a given triangle of $\mathcal{F}$ that contains x points of S and defined by b points will appear in $\Delta^*(P(R))$ is no more than $O(r/n)^b e^{-x(r-b)/n}$; summing this over all triangles in $\mathcal{F}$ gives the bound. □

Thus for the given subset size of the above algorithm, with high probability every $\rho \in \Delta^*(P(R))$ contains $O(\log \hat{r}) n / \hat{r} = O(\log^3 n)/\epsilon^2$ points of S. This is fairly small as a function of n, but unfortunately if we use an exponential algorithm for computing the subtours, the algorithm is superpolynomial in n. To find a polynomial algorithm, we must be a little more devious; the resulting algorithm and its timing analysis are given below.

How good a tour does the algorithm produce? From Theorem 1, it is enough to show that $L_{\Delta^*(P(R))}$ is small. If we assume that the expected cost of a tour of R, a random subset of S, grows as $\Theta(\sqrt{r})$ for $r = |R|$, then the expected length of $\Delta^*(P(R))$ grows at this rate, and so the ratio of expected values $ET(R)/ET(S)$ is $\epsilon / K \log n$; since $L_{\Delta^*(P(R))} = T(R) O(\log n)$, we have $ET^*(S)/ET(S) \leq 1 + \epsilon$ as $n \to \infty$, for suitable choice of K.

2.1 The polynomial-time algorithm.

This section changes the basic algorithm in two ways, to decrease the size of subproblems and to reduce the probability that the given bounds fail. For the former problem, the basic subdivision is refined one step further; for the latter, each triangulation construction is iterated, and the triangulation whose maximum subproblem size is smallest is chosen.

Specifically, iterate the construction of $\Delta^*(P(R))$ for $2n/\hat{r}$ times, taking an independent sample R each time, building $\Delta^*(P(R))$ each time, and determining the maximum size set $\rho \cap S$ for $\rho \in \Delta^*(P(R))$. Now choose the triangulation Δ^c for which that maximum set size is smallest. Now to reduce subproblem sizes, take a random subset $R_\rho \subset S_\rho$ of size $r_\rho = \epsilon^2|S_\rho|/(\log\log n)^2$, and compute $\Delta^*(P(R\rho))$. (Here $P(R_\rho)$ is found by a change to the construction for the basic algorithm: form a simple polygon by attaching an MST of R_ρ not to a bounding rectangle for R_ρ, but to ρ.) Iterate this construction for each ρ for $2|S_\rho|/r_\rho$ times, taking the triangulation of $P(R_\rho)$ whose maximum subproblem size is smallest. The resulting collection of triangulations forms a triangulation Δ^f of $P(R)$, and a TSP can be found as in Theorem 1.

Lemma 2 can be applied to the construction of Δ^c and Δ^f, to readily show that there is a quantity $k = O(\log\log n)^3/\epsilon^2$ so that the probability that $|S \cap \rho| > k + x'$ is not more than $e^{-2x'}$ for all $\rho \in \Delta^f$. With such a high probability of small subproblems, even if we use an algorithm for finding an optimal subtour that requires $x^2 2^x$ time[Bel62, HR62], the resulting expected time is $nk^2 2^k$ for finding subtours; this dominates the total time.

A trickier question concerns the shortness of the subdivision Δ', and so of the resulting tour. The problem here is that we don't have a direct bound on the length of the MST of R_ρ for $\rho \in \Delta$. However, we can obtain a bound on the totals of the lengths using an approximating random subset of S, as follows.

Lemma 3 *For some C_1 and C_2, if $R^* \subset S$ of size $r^* = C_1\epsilon^2 n/(\log\log n)^2$ is a random subset of S then*

$$C_2 ET(R^*) \geq \sum_{\rho\in\Delta} ET(R_\rho).$$

Proof. For $\rho \in \Delta$,

$$\begin{aligned} &ET(R^* \cap \rho) \\ &\quad = \sum_{j\geq 0} E[T(R^* \cap \rho) \mid j = |R^* \cap \rho|]\mathrm{Prob}\{j = |R^* \cap \rho|\}, \end{aligned}$$

and since $E[T(R^* \cap \rho) \mid j = |R^* \cap \rho|]$ is nondecreasing as a function of j, we know that

$$ET(R^* \cap \rho) \geq E[T(R^* \cap \rho) \mid j = r_\rho] \sum_{j \geq r_\rho} \mathrm{Prob}\{j = |R^* \cap \rho|\},$$

and so

$$ET(R^* \cap \rho) \geq ET(R_\rho)\mathrm{Prob}\{|R^* \cap \rho| \geq r_\rho\}.$$

With this, it is enough to show that there is some C_1 so that with the given r^*, we have $\mathrm{Prob}\{|R^* \cap \rho| \leq r_\rho\} \leq 1/C_2$ for some constant C_2. For any j, we have

$$\mathrm{Prob}\{j = |R^* \cap \rho|\}$$

$$
\begin{aligned}
&= \binom{|S_\rho|}{j}\binom{n-|S_\rho|}{r^*-j}\Big/\binom{n}{r^*} \\
&= \binom{r^*}{j}|S_\rho|^{\underline{j}}(n-|S_\rho|)^{\underline{r^*-j}}/n^{\underline{r^*}},
\end{aligned}
$$

where $a^{\underline{b}} = \binom{a}{b}b! \leq a^b$. Since also $n^{\underline{r^*}} = n^{r^*}(1+O(1/n))$,

$$\mathrm{Prob}\{j = |R^* \cap \rho|\} \leq \binom{r^*}{j}p^j(1-p)^{r^*-j}(1+O(1/n)),$$

where $p = |S_\rho|/n$. That is, $\mathrm{Prob}\{j = |R^* \cap \rho|\}$ is bounded above, within $1 + O(1/n))$, by the appropriate probability for a binomial random variable. (Of course, this is not very surprising.) Invoking the Chernoff bound,

$$\mathrm{Prob}\{|R^* \cap \rho| \leq r_\rho\} \leq \exp(-r_\rho C_1^3/2(C_1-1)^2),$$

so $C_1 = 2$ and $C_2 = 1/(1-1/e^4)$ satisfy the conditions of the lemma. (Of course, we must assume that $r_\rho \geq 1$, but otherwise $|S_\rho|$ is small enough that we need not do any subsampling at all.) □

Theorem 4 *Let S be a set of n points in the plane in general position, for which the expected length $ET(R)$ grows as $\Omega(\sqrt{r})$ for a random $R \subset S$ of size r. Then a randomized algorithm can find a tour of S in no more than $nk^2 2^k$ expected time, where $k = O(\log\log n)^3/\epsilon^2$, such that the resulting tour has length $T(S)(1+\epsilon)$.*

Proof. The time required by the algorithm is discussed above; it remains only to show that the tour is short. With high probability, no r_ρ is larger than $\log^3 n/(\log\log n)^2$, so with the algorithm of the following section, and by the lemma above, the expected length of the triangulation Δ^f is within a constant factor of $ET(R^*)\log(\log^3 n/(\log\log n)^2)$, which is no more than $\sqrt{r^*}\log\log n = \epsilon\sqrt{n}$, as desired. □

3 Triangulation of simple polygons

This section gives an algorithm Δ for triangulating simple polygons. Given a simple polygon P with n sides, the algorithm gives a collection of triangles $\Delta(P)$ such that $L_{\Delta(P)} = L_P O(\log n)$.

Two simple examples show the main ideas of the algorithm. In Figure 2, we have a convex polygon triangulated by the algorithm. The idea is the "ring heuristic" [Sup81, PH87], which is simply to progressively "coarsen" the polygon by deleting every other vertex and connecting the remaining vertices, and repeat this until a triangle is obtained. This takes $O(\log n)$ stages of coarsening, and by the triangle inequality, the polygon at a given stage has a perimeter no larger

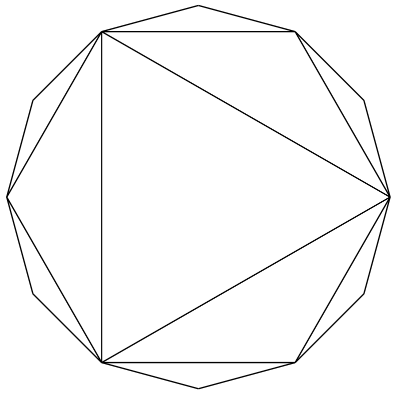

Figure 2: The "ring heuristic" for convex polygons.

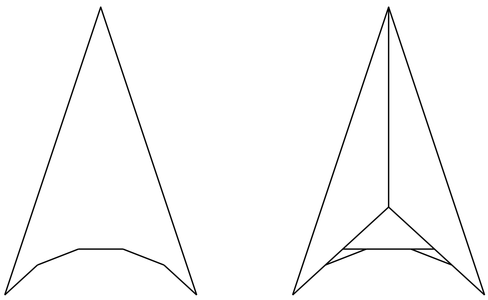

Figure 3: A nonconvex polygon and its triangulation.

than the perimeter of the original. This simple construction solves the problem for convex polygons.

Suppose the polygon is not convex. As an extreme case, consider the polygon in Figure 3, consisting of a *concave chain* spanned by two long edges. A concave chain is chain of edges such that if we walk along the edges keeping the interior of the polygon to our left, we always turn to the right from edge to edge along the chain. Here we cannot apply the same strategy, and indeed any triangulation whose vertices include only the vertices of the polygon can have length $\Omega(n)L_P$.

To tackle this case, we coarsen the polygon by removing edges, as in the right side of Figure 3. At each stage, we delete alternate edges, and extend the remaining edges to give a new polygon. The number of stages is $O(\log n)$; the length of the chain found at each stage is no more than the length of the chain at the next stage. If the angle between the edges at the extreme ends of the chain is small, then the length of the chain at the last stage is within a constant factor of the length of the concave chain. This gives a triangulation of the desired length. (The vertical segment divides the resulting polygon with four sides into two triangles.)

These two coarsening operations are the basis of the general algorithm. To apply these ideas to a general simple polygon, we must generalize them a bit. Consider the operation of adding one edge to a convex polygon, to "cut off" a single vertex. This operation is straightforward for a convex polygon, but what is the analog for a simple polygon? We will use this: given two vertices a and b of the polygon, use the segments of the shortest path between a and b to split up the polygon. The result is a collection of subproblems to be solved recursively; moreover, the total length of the segments used is no larger than the (shortest) path between a and b along the boundary of the polygon.

Unfortunately, we cannot solve resulting subproblems by simply applying this approach recursively; consider again even the convex case. We have then one subproblem to solve after "deleting" a vertex, giving a sequence of n polygons from start to finish. If we make a bad choice for the sequence of vertices to delete, then we will build a lengthy triangulation. A remedy for this problem is to assign weights to edges (or vertices) of the triangulation as we create them; the weight of each edge of the original polygon is one. The weights will help in making the choice of which vertex to delete, to create a series of deletions corresponding to a balanced tree, and giving the desired length for the triangulation.

Before a more precise description of the general algorithm, here is an algorithm for triangulating a concave chain of polygon edges, as above, with the additional complication of taking weights into account. The algorithm applies to concave chains with *rotation* at least $-\pi/2$, where the rotation of a chain is the sum of the angles that we turn as we walk along the chain, keeping the interior of the polygon to our left. Note that the rotation of a concave chain is negative, and the rotation of the whole polygon is 2π, as can be shown using induction on the number of triangles in a triangulation of the polygon. (The rotation of the polygon, possibly divided by 2π, is apparently also called the rotation number, the degree, the winding number, or the tangent winding number: let the reader beware.)

Suppose a concave chain is given, with a rotation at least $-\pi/2$. Suppose edge e in the chain has weight w_e, a positive integer. For a vertex u incident to edges e and f, assign weight $w_u = w_e + w_f$. (If u is an endpoint, let w_u be the weight of the edge incident to u.) The triangulation algorithm repeats the following general step until done: pick an edge e with endpoints u and v so that $w_u + w_v$ is the smallest such sum over all edges. Extend the edges incident to e until they meet above e, creating a new vertex x. Assign the weight $w_u + w_v$ to x.

As stated by the next lemma, this algorithm produces a triangulation for which a given edge contributes a length that is decreasing as a function of its weight.

For a polygonal chain P, let $\operatorname{edge} P$ denote the collection of its edges.

Lemma 5 *Let* $w_c = \sum_{e \in \text{edge } c} w_e$ *for chain* c*. Given a chain* c *of rotation at least* $-\pi/2$*, the algorithm produces a triangulation of length no more than* $K_1 \sum_{e \in \text{edge } c} L_e \lg(2w_c/w_e)$*, for some constant* K_1*.*

Proof. Call two vertices of the triangulation *visible* to each other if the line segment between them doesn't cross the chain c. Call an edge of c visible to a vertex if both of its endpoints are visible to the vertex. When a vertex x and its incident segments are added to the triangulation, the total length of the incident segments is no more than the $\sqrt{2}$ times the total length of the edges of c that are visible to x. Thus if each such visible edge is charged its length when x is added, the total charges will be proportional to the length of the triangulation. The total charge for an edge e is L_e, multiplied by the number of vertices that see it. We need to show that the number of such vertices is proportional to $\lg(2w_c/w_e)$.

Note that we've built a binary tree on the vertices, and the subtree rooted at a vertex contains the vertices that see it; we've seeking to bound the path length from the root of the tree to a given vertex of c: since the tree is roughly weight-balanced, this can be done appropriately.

Suppose u, x, x' and x'' are consecutive vertices on a path to the root node of the binary tree. It is enough to show that $w_u \leq w_{x''}/2$; this implies that a path from chain vertex v to the root has length no more than $2\lg(2w_c/w_v)$, which then also bounds the number of vertices that see an edge incident to v.

To show the inequality, suppose the "current chain" has consecutive vertices u'', u', u, v, v', v'' when x is added to the triangulation above the edge e with endpoints u and v. By the choice of e, we know that $w_u + w_v = w_x \leq w_{u'} + w_{v'}$, $w_x \leq w_{u''} + w_{u'}$, and $w_x \leq w_{v''} + w_{v'}$. Since some pair of vertices $\{u'', u'\}, \{u', v'\}, \{v', v''\}$ contributes to $w_{x''}$, as well as u and v, we know that $w_u \leq w_{x''}/2$. □

In general for simple polygons, we consider a polygon whose edges have positive integral weights, and whose boundary is split into a collection of concave chains. The weight of such a chain is the sum of the weights of the edges in it. In the original polygon, every chain is one edge of weight 1.

First we extend the above claim to simple polygons with boundaries comprising few chains.

Lemma 6 *Let* P *be a simple polygon with weighted edges of total weight* n*, and whose boundary comprises three or four concave chains. Then* P *has a triangulation of length no more than* $K_2 \sum_{e \in \text{edge } P} L_e \log(2n/w_e)$*, where* K_2 *is a constant.*

Proof. The polygon can be split into at most 8 chains each of rotation at least $-\pi/2$: suppose at least 9 are needed. Then there are chains of total rotation less than -2π; since the polygon has rotation 2π, and a pair of edges gives rotation at most π, there must be more than $(2\pi + 2\pi)/\pi = 4$ pairs of edges each of positive rotation, a contradiction.

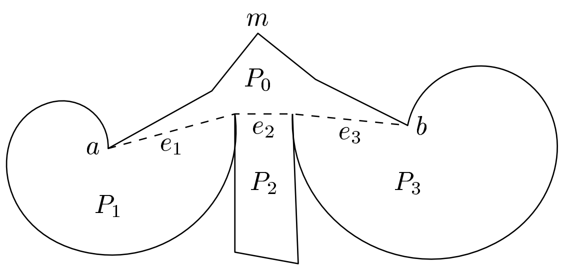

Figure 4: The general step for simple polygons.

We apply the algorithm of Lemma 5 to a chain, and find the intersection of the resulting triangulation with the interior of the polygon. This collection of triangles and quadrilaterals can be easily triangulated; moreover, we obtain either a simple polygon with a concave chain having two edges, or two simple polygons; repeat until polygons with fewer edges than some constant are obtained, and triangulate arbitrarily. The given bound easily follows. □

Now consider a simple polygon whose boundary comprises more than three concave chains. To do the general step, we find a pair of adjacent chains C and D, such that $w_C + w_D$ is minimum over all such pairs of chains. Suppose a and b are the respective distinct endpoints of C and D, and m is their common endpoint. We will include some segments of the shortest path in P from a to b in the triangulation. This will split P up into smaller polygons, using a path of length no more than the length of the chains C and D. To find these shortest paths, we preprocess the original polygon for shortest path queries[GH87], so that queries require time $O(\log n)$ plus the number of segments in the path.

We can characterize the shortest path from a to b within P as follows: the path consists of a chain of line segments, having three subchains, two of which are subchains of C and D, with another subchain E through the interior of P. Any one of these subchains may be empty. Suppose chain E is not empty. Let F denote the portion of C and D not in the shortest path from a to b. The segments of E divide P into pieces; associated with each $e \in \text{edge}\, E$ is a polygon P_e comprising that portion of P separated from m by e. Also, there is the polygon P_0 bounded by E and F. (In Figure 4, E has three edges, and F is the concatenation of C and D.)

The general step of the algorithm is as follows: suppose E is empty, so that the shortest path from a to b is simply the chains C and D together. (This occurs when C and D meet at a concave angle.) Then the general step consists of simply concatenating the chains C and D. Suppose E is not empty. Then recursively triangulate the P_e, for $e \in \text{edge}\, E$, giving every e the weight $n - w'_e$ for this purpose, where w'_e is the sum of the weights of the edges of P_e, not including e. (Note that with this assignment, $w_P = n$ for every polygon P input to the algorithm.) Use the algorithm of Lemma 6 for triangulating polygons with three or four concave chains. This leaves the problem of triangulating P_0. Here if e is not an edge of P, assign it weight w'_e; if e is an edge of P, it retains its weight. With these assignments, apply the algorithm of Lemma 6 to P_0.

Call the edges of E *inward* edges, and the edges made by calls to the algorithm of Lemma 6 *outward* edges. Include the edges of P itself also in the inward edges. We first bound the total length of all the inward edges.

Lemma 7 *The inward edges resulting from the above procedure have total length no more than*

$$K_3 \sum_{g \in \text{edge}\, P} L_g \lg(2n/w_g),$$

for a constant K_3.

Proof. The theorem implies that each edge g of P can be charged $C_g(P) = K_3 L_g \log(n/w_g)$, and the resulting total charges over all edges accounts for the length of the triangulation. We will show inductively that the length of the triangulations of the P_e, and of P_0, can be distributed to the edges of P to satisfy this condition. For this, it is enough to show that the charges for edges of E can be appropriately distributed to the edges of F.

Suppose that edge $e \in \text{edge}\, E$ is charged xL_e for the triangulation of P_i, for some x. The charges to the edges in E must be distributed to the edges of F. Suppose $f \in \text{edge}\, F$ is charged $xL_e L_f/L_F$ for each $e \in \text{edge}\, E$, plus its own length; then the total charge distributed from each e is xL_e. If the edges of P not in F are appropriately charged inductively, the total charge is the length of the inward edges. On the other hand, if every $e \in \text{edge}\, E$ is charged no more than xL_e, then f is charged $xL_f L_E/L_F + L_f \le (x+1)L_f$.

Say that the edges of E are *above* those of F. We can apply charging across several levels of this relation: suppose edges above those of E are charged no more than x times their lengths; then by charging to edges of E and then F, the total charge to $f \in \text{edge}\, F$ is no more than $(x+2)L_f$. In general, if edges at C levels above f are charged no more than x times their lengths, then f is charged $(x+C)L_f$.

We now proceed analogously to the proof of Lemma 5: if an edge g is above an edge above an edge,..., above f, for a sufficient (but fixed) number C of levels, then $w_g > 2w_F$.

Suppose, inductively, that the charge to g is $L_g K_3 \lg(2n/w_g) \le L_g K_3 \lg(n/w_F)$; by the above, the transferred charge to f is $L_f(K_3 \lg(n/w_F) + C)$. This is no more than $L_f K_3 \lg(2n/w_f)$ for $K_3 \ge C$. □

Theorem 8 *The given procedure yields a triangulation with length no more than*

$$K_4 \sum_{g \in \text{edge}\, P} L_g \log(2n/w_g),$$

for some constant K_4.

Proof. As in the previous lemma, we show that each edge g of P can be charged $C_g(P) = K_4 L_g \log(n/w_g)$, and the resulting total charges over all edges bound the length of the triangulation. We will show inductively that the cost of triangulating P_0 and the P_e, for $e \in \text{edge}\, E$, can be distributed to the edges of P to satisfy this condition.

Suppose the theorem holds inductively for the polygons P_e, so each edge g of $P_e \setminus \{e\}$ is charged by the prescribed $C_g(P_e) \leq C_g(P)$; we will show that the charges to edges of P_0 from the triangulation of P_0, and the charge to e for the triangulation of P_e, can be charged to the edges of F in an amount no more than $C_f(P)$ for $f \in \text{edge}\, F$.

The charge due to $e \in \text{edge}\, E$ for triangulation of P_0 is $K_2 L_e \lg((n - w_F)/w'_e)$, by Lemma 6, noting that $n - w_F = \sum_{e \in \text{edge}\, E} w'_e$.

Suppose for $e \in \text{edge}\, E$, the polygon P_e comprises no more than four concave chains, so the algorithm of Lemma 6 is applied. By that lemma, the length of the resulting triangulation is at most $\sum_{g \in \text{edge}\, P_e} K_2 L_g \lg(2n/w_g)$. Adding to this the charge $K_2 L_e \log(2(n - w_F)/w'_e)$ due to $e \in \text{edge}\, E$ for triangulation of P_0, and total

$$K_2 L_e \lg \frac{2(n - w_F)}{w'_e} + \sum_{g \in \text{edge}\, P_e \setminus \{e\}} K_2 L_g \lg \frac{2n}{w_g},$$

which is bounded by

$$\sum_{g \in \text{edge}\, P_e \setminus \{e\}} K_4 L_g \lg \frac{2n}{w_g},$$

for $K_4 > 2K_2$, noting that obviously $w'_e \geq w_g$ for $g \in P_e \setminus \{e\}$, and $L_e \leq L_{P_e}$. Thus the additional charges can be made to the edges of P_e.

If P_e comprises more than four concave edges,— **is this defined?** — the charge to e for triangulation of P_e is $K_4 L_e \lg(2n/(n - w'_e))$, by induction. Adding to this the charge to e for triangulation of P_0, we have

$$L_e \left(K_2 \lg \frac{n - w_F}{w'_e} + K_4 \lg \frac{n}{n - w'_e} \right)$$

to distribute to the edges of F. It is enough to show that these edges are charged at most $K_4 \lg(2n/w_F)$ times their length. Since $L_E \leq L_F$, it would suffice if the above charge for e were less than $L_e K_4 \lg(2n/w_F)$. This holds when the quantity

$$\begin{aligned} & K_4 \lg \frac{2n}{w_F} - K_4 \lg \frac{2n}{n - w'_e} - K_2 \lg \frac{2(n - w_F)}{w'_e} \\ = \quad & K_4 \lg \frac{n - w'_e}{w_F} - K_2 \lg \frac{2(n - w_F)}{w'_e} \qquad (1) \end{aligned}$$

is nonnegative. Since P_e contains at least five concave chains, $w'_e \geq w_F$ by choice of C and D. Therefore (1) is nonnegative for $w'_e \leq n/3$ and $K_4 \geq K_2$.

If $w'_e > n/3$, quantity (1) is at least $-K_2 \lg 6$. That is, each edge $f \in \operatorname{edge} F$ is charged $L_f K_2 \lg 6$ more than can be accounted for inductively. However, such additional charges the previous lemma, and the theorem follows by appropriately adjusting the constant. □

4 Concluding remarks

Two natural questions raised by these results: can something be done for point sets for which $T(R) \approx T(S)$ for small random subsets R? Can an approximation algorithm be found for non-Steiner triangulations that requires $n \log^{O(1)} n$ time?

References

[Bel62] R. E. Bellman. Dynamic programming treatment of the traveling salesman problem. *Journal of the ACM*, 9:61–63, 1962.

[Chr76] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete and Computational Geometry*, 2:195–222, 1987.

[DH90] D. Z. Du and F. K. Hwang. An approach to proving lower bounds: solution of Gilbert-Pollack's conjecture on Steiner ration. In *Proceedings of the 31th Annual IEEE Symposium on Foundations of Computer Science*, pages 76–85, 1990.

[GH87] L. J. Guibas and J. Hershberger. Optimal shortest path queries in a simple polygon. In *Proceedings of the Third Symposium on Computational Geometry*, pages 50–63, 1987.

[HR62] M. Held and M. Karp R. A dynamic programming approach to sequencing problems. *SIAM J.*, 10:196–210, 1962.

[Kar77] R. M. Karp. Probabilistic analysis of partitioning algorithms for the traveling-salesman problem in the plane. *Math. of Operations Research*, 2:209–224, 1977.

[LLKS85] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The traveling salesman problem.* Wiley, New York, 1985.

[Pap77] C. H. Papadamitriou. The Euclidean traveling salesman problem is NP-complete. *Theoretical Computer Science*, 4:237–244, 1977.

[PH87] D. A. Plaisted and J. Hong. A heuristic triangulation algorithm. *Journal of Algorithms*, 8:405–437, 1987.

[Smi89] W. D. Smith. Implementing the Plaisted-Hong min-length plane triangulation heuristic. unpublished, 1989.

[Ste81a] J. M. Steele. Complete convergence of short paths and Karp's algorithm for the TSP. *Math. of Operations Research*, 6:374–378, 1981.

[Ste81b] J. M. Steele. Subadditive euclidean functionals and nonlinear growth in geometric probability. *Annals of Prob.*, 9:365–376, 1981.

[Sup81] K. J. Supowit. Topics in computational geometry. Technical Report UIUCDCS-R-81-1062, University of Illinois, 1981.