A Polynomial-Time Approximation Scheme for Weighted Planar Graph TSP

Sanjeev Arora* Princeton U.

Michelangelo Grigni† Emory U.

David Karger‡

Philip Klein§ Brown U.

Andrzej Woloszyn Emory U.

Abstract

Given a planar graph on n nodes with costs (weights) on its edges, define the distance between nodes i and j as the length of the shortest path between i and j. Consider this as an instance of *metric* TSP. For any $\varepsilon > 0$, our algorithm finds a salesman tour of total cost at most $(1+\varepsilon)$ times optimal in time $n^{O(1/\varepsilon^2)}$.

We also present a quasi-polynomial time algorithm for the Steiner version of this problem.

1 Introduction

The TSP has been a testbed for virtually every algorithmic idea in the past few decades [7]. Most interesting sub-cases of the problem are NP-hard, so attention has turned to other notions of a "good solution." An early *approximation algorithm* by Christofides achieves an approximation ratio 1.5 on every instance of *metric TSP* (namely, one in which the internode distances form a metric). This algorithm has not been improved upon, and a general approximation scheme is unlikely since metric TSP is MAXSNP-hard [13]. Nevertheless, there has been recent progress for certain special classes of metrics.

Grigni, Koutsoupias, and Papadimitriou [6] showed that if the metric is the shortest-path metric of an unweighted planar graph (that is, all edge costs are one), then a *polynomial time approximation scheme* (or PTAS) exists. It achieves an approximation ratio $(1+\varepsilon)$ in $n^{O(1/\varepsilon)}$ time, where $\varepsilon > 0$ is any constant. Then Arora [2] (and soon after, Mitchell [11]) showed that a PTAS also exists for *Euclidean TSP* (i.e., the sub-case in which the points lie in $\Re^2$ and distance is measured using the Euclidean metric). This PTAS also achieves an approximation ratio $(1+\varepsilon)$ in $n^{O(1/\varepsilon)}$ time. More recently, Arora [3] improved the running time of his algorithm to $O(n \cdot (\log n)^{O(1/\varepsilon)})$, using randomization.

The PTAS's for Euclidean TSP and planar-graph TSP, though discovered within a year of each other, are quite different. Interestingly enough, Grigni et al. had conjectured the existence of a PTAS for Euclidean TSP, and suggested one line of attack: to design a PTAS for the case of a shortest-path metric of a *weighted* planar graph. (We note later in the paper that Euclidean TSP can be viewed as a *Steiner* sub-case of weighted planar graph TSP.)

In this paper we present an $n^{O(1/\varepsilon^2)}$ time PTAS scheme for weighted-planar-graph TSP. Our approximation scheme starts by extracting a spanner of the input graph; the spanner preserves distances between vertices but has a lower sum of edge-costs. Then, like some previous results about approximation schemes on planar (unweighted) graphs [8, 6], our approximation scheme employs a technique for obtaining a *hierarchical decomposition* of the input graph. By "hierarchical" we mean that the decomposition is done in steps. A step partitions every current component into two or more pieces, each of which contains at most a constant fraction of the vertices of its parent. Hence the decomposition tree has $O(\log n)$ levels, where n is the number of vertices. The useful aspect of our decomposition is that we show the existence of a $(1+\varepsilon)$-approximate salesman tour that crosses the boundary of each region in the decomposition $O(\log n/\varepsilon^2)$ times. This is reminiscent of the approximation scheme for the unweighted case [6]. In particular, we ensure that after contracting the edges that lie on the boundaries of the regions, we obtain a graph where each region is bounded by $O(\log n/\varepsilon^2)$ ver-

*`arora@cs.princeton.edu`

Supported by NSF CAREER Award, Alfred Sloan Fellowship, and Packard Fellowship.

†Emory Dept. of Math. & Computer Sci., Atlanta GA 30322. E-mail: `mic@mathcs.emory.edu`.

‡MIT Laboratory for Computer Science, Cambridge, MA 02138.

Supported by NSF contract CCR-9624239 and an Alfred P. Sloane Foundation Fellowship.

E-mail: `kargerlcs.mit.edu`.

URL: `http://theory.lcs.mit.edu/~/karger`

§`klein@cs.brown.edu`

Supported by NSF grant CCR-97000146

tices. Dynamic programming can be used to find the optimal tour in this contracted graph. We show how to lift this tour to a tour in the uncontracted graph without increasing the cost by much.

In Section 6, we also present a quasi-polynomial time algorithm for the Steiner version of this problem; that is, when we want a near-optimal tour on some given subset S of the vertices.

2 Definitions and procedures

2.1 Spanner Our approach requires that the sum of the edge-costs of the graph be within a constant factor of the cost of the minimum tour. We use the following spanner result due to Althofer, Das, Dobkin, Joseph, and Soares to obtain such a low-cost subgraph of the input graph that also approximately preserves distances.

Theorem 1 ([1]) *For every planar graph G with edge-costs and every $\varepsilon > 0$, there is a subgraph G' with the same vertex set such that for every pair of vertices i and j in V, their distance in G' is at most $(1+\varepsilon)$ times their distance in G. Furthermore, the sum of costs of edges appearing in G' is $O(1/\varepsilon)$ times the cost of a minimum spanning tree in G. The subgraph G' can be found in polynomial time.*

2.2 Separating the graph Now we give our separator theorem, which can be viewed as a generalization of Miller's simple-cycle-separator theorem. The proof of the theorem appears in Section 5.

It is useful to ensure that our separator, while not a cycle in the original graph, does trace out a Jordan curve. To capture the resemblance to a cycle, we use the notion of *face edges*. Face edges are artificial edges added to the graph while preserving planarity; in other words, the cycle may cross through the interior of some faces. Our separator will be a cycle in the graph obtained by adding some of these artificial edges. We call it a *cycle separator*. Such a cycle traverses certain vertices, edges, and faces of the graph; the rest of the graph divides into an *interior* and an *exterior*.

The separator is supposed to achieve some sort of balance. We specify what is to be balanced by assigning weights to vertices and faces; the separator then has the property that the sum of weights of vertices and faces in the interior is only a constant fraction of the sum of all the weights, and similarly for the exterior.

We measure the quality of the separator in two ways: the number of face edges it uses, and the sum of costs of the ordinary edges it uses.

Theorem 2 (Separator Theorem) *There is an* $\mathrm{poly}(n)$*-time algorithm that, given a parameter k and a planar embedded graph G with edge-costs, vertex-weights, and face-weights, finds a Jordan curve $\mathcal{C}$ such that*

balance condition: *the interior and exterior of $\mathcal{C}$ have weight at most $2/3$ of the total weight;*

face-edge condition: *$\mathcal{C}$ uses at most k face edges,*

ordinary-edge condition: *$\mathcal{C}$ uses ordinary edges of total cost $O(1/k)$ times the total cost of all the ordinary edges.*

Remark: *The separating cycle also has the property that the ordinary edges comprise at most two paths, but the algorithm does not use this property.*

Given a planar embedded graph G and a separator $\mathcal{C}$ satisfying the conditions of Theorem 2, we *split* the graph into two or more derived subgraphs, called *children*, as follows.

First, in G we contract the ordinary edges that are in $\mathcal{C}$, giving us a contracted planar embedded graph G' (the ordinary-edge condition ensures that we do not contract much cost, so we do not dramatically perturb the solution). Let $\mathcal{C}'$ be the contracted version of the separating cycle; it has no ordinary edges so the vertices it contains, if removed, separate the graph (the number of vertices is equal to the number of face edges in the separator, so the face-edge condition ensures that there are not too many). We call these vertices the *boundary vertices.*

We define the interior piece to be the interior of $\mathcal{C}'$, together with the boundary vertices. The exterior piece is similarly defined (it too contains the boundary vertices). The interior piece and exterior piece each satisfy the following property.

> **Boundary-vertices property:** The boundary vertices lie on the boundary of a single face

We call the single face a *hole* because it results from the removal of some of the graph.

The connected components of the interior piece and the exterior piece are the *children* of G. Because of the balance condition on the separator, the interior of $\mathcal{C}'$ has vertex weight at most two-thirds of the vertex weight of G. We obtain the following property.

> **Child-weight property:** Each child has vertex weight at most a constant fraction of that of the parent graph.

Because all ordinary edges on the boundary between the interior and exterior were contracted, we obtain the following property.

edge-disjointness property: No edge of G appears in more than one child.

Moreover, the boundary-vertices property still holds for each child: the boundary vertices that separate the child from its siblings all lie on the boundary of a single face, a hole.

3 A simpler algorithm

To introduce our techniques, we begin with a simpler algorithm that runs in quasipolynomial time. In the following section, we will show how to modify it to achieve a polynomial running time.

Our goal is to find a tour in the input graph that has cost at most $1+\varepsilon$ times that of the optimal tour. Let OPT denote the cost of the optimal tour in the input graph. The first step of our algorithm is to let G be a spanner of the input graph that preserves distances up to a factor of $1+\varepsilon/2$, using Theorem 1. The sum of edge-costs of G is $O(1/\varepsilon)$ times the cost of a minimum spanning tree in the input graph, which is in turn at most OPT. The cost of an optimal tour in G is at most $1+\varepsilon/2$ times OPT. Our goal is now to find a tour in G whose cost is at most $\varepsilon OPT/2$ more than the optimal tour.

Next, we fix the parameter $k = c\log n/\varepsilon^2$, where n is the number of vertices in the given graph and c is a constant to be determined. We assign weight one to every vertex in G.

Next, we find a recursive decomposition of G using the separator algorithm of Theorem 2: we find a separator in G, determine the children of G, find a separator in each child and determine its children, and so on, until each remaining graph has a small number of vertices. By the child-weight property, the depth of the recursion is $O(\log n)$.

Consider the graphs occuring at a particular level of the recursive decomposition. By the edge-disjointness property, these graphs are edge-disjoint. Hence the sum of edge-costs over all these graphs is at most the sum of edge-costs in G. For each graph, the (contracted) edges of the separating cycle used to divide the graph have total cost $O(1/k)$ times the sum of edge-costs in that graph. Summing over all these graphs, the costs of edges appearing in all these separators is $O(1/k)$ times the total cost in G, i.e. $O(OPT/k\varepsilon)$. Summing over all levels of the decomposition, we obtain a bound of $O(OPT\log n/k\varepsilon)$ on the sum of costs of all edges contracted. By appropriate choice of the constant c in the definition of k, we ensure that this bound is at most $\varepsilon OPT/4$.

Let G' be the graph obtained from G by contracting all the primal separator edges in all the separators. We obtain the following lemma.

Lemma 3 *The edges of G that do not appear in G' have total cost at most $\varepsilon OPT/4$.*

Clearly the optimal tour in G' has length at most OPT. Below we show how to use dynamic programming to compute the *optimal* tour for G'. We then extend this tour back to the original graph as follows. We uncontract the contracted paths and use the classical double-MST heuristic on each path to get a tour for those vertices of cost at most $2\cdot$ path length. We then "splice" the path tour into the global tour. Doing this for all contracted paths raises the tour cost by at most $\varepsilon/2\cdot OPT$, so the the cost of the final tour is at most $(1+\varepsilon/2)\cdot OPT$ (the other $\varepsilon/2$ is taken by the spanner subgraph approximation).

3.1 Finding the optimal tour in G' in quasipolynomial time

Now we describe how to compute the optimum salesman tour on the contracted graph. We show that the running time is $n^{O((\log n\log\log n)/\varepsilon^2)}$. In the next section, we use a more careful analysis to show that polynomial running time can be achieved.

The recursive decomposition of G using the cycle separators induces a recursive decomposition of G'; namely, the decomposition obtained by using the contracted versions $\mathcal{C}'$ of the separators. After the contractions, the number of vertices in a separator is equal to the number of face edges in the separator. So (by the face-edge condition) each separator consists of at most k vertices. Consider a subgraph H of G' that appears at level d of the recursive decomposition. It contains at most dk boundary vertices, and they collectively separate H from the rest of G'. Note that $dk = O(\log^2 n/\varepsilon^2)$ because the depth of the decomposition is $O(\log n)$. The optimal tour in G' winds in and out of H via the boundary vertices. The next lemma guarantees that there is an optimal tour that passes through each boundary vertex at most most twice.

Lemma 4 (Patching Lemma) *Let J be a Jordan curve in the plane. Let π be a closed curve in the plane. Then we can modify π into another closed curve π' that traverses the same points as π but such that at every point p where π' intersects J, π' crosses J at most twice.*

Proof: First by eliminating crossings, we may assume that π does not self-intersect. Then at a point of intersection with J, the curve π alternates going in and out of J. We may then reconnect the segments of the curve so that at most two crossings are necessary. See Figure 1. $\square$

Thus the contracted graph G' has a simple structure: it possesses a hierarchical decomposition such that

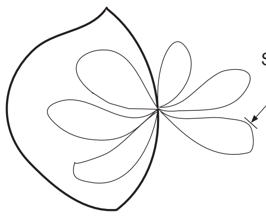


Figure 1: If a planar salesman tour crosses a Jordan curve more than two times at a boundary vertex, then we can clearly reconnect the edges incident to the boundary vertex so that the tour crosses at most twice.

each subgraph H arising in the decomposition is delimited by $p = O(\log^2 n/\varepsilon^2)$ boundary vertices, and the optimum salesman tour enters or leaves H at most twice via each boundary vertex.

We use a simple dynamic program to find the optimum tour as follows. We build a lookup table for each subgraph arising in the decomposition, starting from the leaf subgraphs of the decomposition, and working up to larger subgraphs. For a subgraph H, the corresponding table contains an entry for every possible way the tour can pass through H, which can be specified by an ordered list of boundary vertices through which the tour enters and exits H. Each boundary vertex can appear at most twice, so the number of entries is bounded by the number of ways of ordering at most $2p$ items, which is $p^{O(p)}$.

The value of an entry is the minimum cost of a set of paths that realize the corresponding pairing while visiting all vertices inside H. This value can be calculated from similar tables for the children of H: Given the contracted cycle separator $\mathcal{C}'$ used to separate H, we enumerate the ways the set of paths could cross $\mathcal{C}'$. For each, we look up the minimum-cost realization within each child, and then add these costs. The minimum over all the ways the paths could cross $\mathcal{C}'$ is the desired value.

To calculate each entry in H's table thus requires $p^{O(p)}$ time, and there are $p^{O(p)}$ entries to fill, for a total of $p^{O(p)}$ time. We have to build a table for each subgraph in the decomposition. The total time (and space) required is $n^{O(1)} \cdot p^{O(p)} = n^{O((\log n \log\log n)/\varepsilon^2)}$.

4 A polynomial algorithm

We now improve the preceding algorithm to achieve a polynomial running time. We modify the assignment of weights used in finding the cycle separators in order to ensure that, for every subgraph H arising in the decomposition, the number of boundary vertices in H separating it from the rest of the graph is small, and that these boundary vertices lie on the boundaries of only a constant number of holes. This last fact, together with the planarity of an optimal tour, allows us to prove a Catalan-like $2^{O(k)}$ upper bound on the number of subproblems that we must actually consider at H. Putting this together, we will have running time $n^{O(1)}2^{O(k)} = n^{O(1/\varepsilon^2)}$ as claimed.

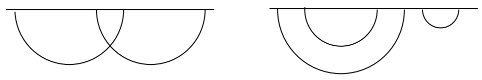
The new assignment of weights is actually done dynamically as we construct the recursive decomposition. Suppose we are separating a graph H with total weight w. When we find a separator in H, contract its edges, and construct its children, we create some new boundary vertices for the children. We assign weight $w/12k$ to each new boundary vertex in a child. Since each child has only k boundary vertices, this adds no more than $w/12$ to a child's total weight. We also assign weight $w/12$ to the hole in each child on which the boundary vertices lie.

Before additional weight was assigned to boundary vertices, each child's total weight was at most $\frac{2}{3}w$. The additional weight increases the child's weight by at most $\frac{1}{6}w$, so the child's weight is now at most $\frac{5}{6}w$. Thus the child-weight property still holds: each child's weight is still a constant fraction of its parent's.

Now consider the hierarchical decomposition of the graph G' obtained by the contractions. Let H be a subgraph arising in the decomposition. First we bound the number of boundary vertices in H. The weight of every boundary vertex in H is at least $1/12k$ times the weight of H's parent. Since H has weight at most five-sixths that of its parent, it must contain at most $10k$ boundary vertices.

Next we show that the boundary vertices lie on a constant number of holes. As the recursive decomposition takes place (a cycle separator is found for a subgraph, and its children are constructed), each existing hole is either preserved (if the cycle separator does not pass through the face) or destroyed (if the cycle passes through the face). In the former case, the hole retains the weight originally assigned to it when it was formed, and it appears in one of the children of the subgraph. In the latter case, the boundary of the hole is broken into two pieces, and each piece makes up part of the boundary of the *new* hole being formed in the children. Thus in this case the old boundary vertices (which lined the boundary of the old hole) now lie on the boundaries of the new holes appearing in the children (one new hole per child). We preserve the property that all boundary vertices lie on the boundaries of holes.

We can now use an argument like that used above in bounding the number of boundary vertices. Let H be a subgraph arising in the hierarchical decomposition of the contracted graph. Each hole has weight at least



(A) (B)

Figure 2: The order in which a planar tour traverses m vertices on the boundary of a Jordan curve corresponds to to balanced arrangements of m pairs of parenthesis. Each boundary vertex can contribute two parentheses. (A) shows an invalid pairing and (B) shows a valid pairing.

1/12 times the weight of H's parent, and H's weight is at most five-sixths that of its parent, so H contains at most 10 holes.

(Better constants can be obtained by slightly more complicated ways of breaking up the graph and assigning weights.)

Enumerating the pairings To finish the description of the algorithm, we need to bound the number of ways in which an optimum salesman tour could link up the (at most) $10k$ boundary vertices in a subgraph. Note that we can assume that the optimal tour does not cross itself (though it can overlap itself) and passes through each boundary vertex at most twice.

If in fact all the boundary vertices were on a single face, then we could finish with a Catalan bound. That is, suppose we have a subgraph with a unique hole with m boundary vertices. The portion of the tour on the subgraph (that is, the exterior of the hole) consists of a non-crossing set of paths which matches pairs of boundary vertices (some more than once, but none more than twice). The number of such matchings is $2^{O(m)}$, as may be proven by relating this quantity to Catalan numbers (Figure 2).

However, instead of a disk, we have the subgraph embedded on a sphere with $h = O(1)$ holes in it. The number of boundary vertices on the boundary of each hole is $O(k)$. Again, we want an upper bound on the number of topologically distinct ways to embed a non-crossing matching on this surface.

We claim an upper bound of $(h)^{O(h)}2^{O(hk)}$, which is $2^{O(k)}$ in our case since h is constant. This is seen as follows. Any embedded matching $\mathcal{M}$ is a non-crossing multigraph with the holes as its vertices. The multigraph is connected because it represents a salesman tour. If we cut the surface along any spanning subtree of this multigraph, then the h holes merge into a single hole. This single hole has $O(hk)$ boundary vertices, and our earlier analysis bounds the number of noncrossing matchings among them by $2^{O(hk)}$. In other words, the number of choices for the matching $\mathcal{M}$ on the original set of boundary vertices is at most

$$(\text{number of nonself-crossing trees on } h \text{ points}) \times 2^{O(hk)},$$

which is at most $h^{O(h)} \cdot 2^{O(hk)}$.

5 The separator theorem

Now we prove Theorem 2. Our proof uses ideas from Grigni, Koutsoupias, and Papadimitriou [6] as well as Lipton and Tarjan's planar separator theorem [8] and Miller's simple cycle separator theorem [10]. The performance criteria in our case are different from those of Lipton and Tarjan and those of Miller, so we must use these techniques differently.

By rescaling, we can assume without loss of generality that all vertex and face weights sum to 1.

The next lemma is borrowed (with slight modification) from Lipton and Tarjan.

Lemma 5 (Lipton and Tarjan) *Let T be a spanning tree of a planar triangulated graph G with weights on vertices and faces totaling at most 1. Then there is an edge not belonging to T such that the interior and exterior of the simple cycle in $T \cup e$ each contain weight no more than 2/3.*

Furthermore, the same result holds even if not every face is triangular, as long as each non-triangular face F satisfies the following two properties.

- *F has weight less than 2/3.*
- *The boundary of F is the unique simple cycle in $T \cup e$ for some edge e.*

The first step of the separator algorithm is to find the shortest-path tree rooted at some vertex r. In the second step, described below, we extract a subgraph of the graph (and a corresponding subgraph of the shortest-path tree); we also add some face edges. Finally, we apply the lemma of Lipton and Tarjan to obtain a simple cycle separator.

5.1 Extracting a subgraph

Note that any positive value x determines a partition of the vertices: those whose distance from r is less than x, and those whose distance is at least x. For any subset S of the edges of the graph, let $f_S(x)$ denote the number of edges in S that cross this partition.

Lemma 6 *For any subset S of edges and any distances $d_1 < d_2$, if $f_S(x) \geq k/2$ for every $d_1 < x \leq d_2$ then $d_2 - d_1 \leq 2L/k$, where L is the total cost of the edges in S.*

Proof: An edge of length ℓ can only contribute to $f_S(x)$ for an interval of length ℓ. The lemma follows by simple averaging. □

Let $(S_d, \overline{S_d})$ denote the partition induced by cutting the shortest-path tree at distance d. The separator algorithm will use the topology of these partitions. Note that there are at most n such partitions, as we can always take d to be the distance of some vertex from the root.

The set of edges crossing the partition $(S_d, \overline{S_d})$ is a *co-cycle*, which means that in the planar dual graph the duals of these edges form a collection of edge-disjoint simple cycles. Moreover, we can view the embedding of the planar graph in such a way that the *interiors* of these dual cycles contain the nodes at distance d or greater. Let $\mathcal{C}_d$ denote this set of dual cycles. (Essentially, we choose as our infinite region some face whose boundary contains the root r.)

As shown in Figure 3, the dual cycles are nested: for $d_1 < d_2$, each cycle of $\mathcal{C}_{d_2}$ lies in some cycle of $\mathcal{C}_{d_1}$. The minimal nontrivial dual cycles each enclose a single vertex of the primal graph.

Define the *depth* of a dual cycle in $\cup_d C_d$ to be the maximum d for which the cycle belongs to $\mathcal{C}_d$. Define its *edge-cardinality* to be the number of dual edges comprising it. For a collection of such cycles, the *total edge-cardinality* is the sum of the edge-cardinalities of the individual cycles, i.e. the total number of dual edges appearing in all of them. Define the *weight contained* by a dual cycle to be the sum of weights of vertices and faces embedded in the interior of the cycle.

Let C denote a deepest dual cycle containing weight more than 1/2. Let B the the deepest ancestor of C whose edge-cardinality is less than $k/2$. (See Figure 4.) Let d_B be the depth of B. Note that B contains weight more than 1/2. Define $\hat{d}$ as

$$\hat{d} := \min \{d > d_B \ : \mathcal{C}_d \text{ has total edge-cardinality } < k/2\}$$

and let $C_1, \ldots, C_r$ be the cycles in $\mathcal{C}_{\hat{d}}$ that are in the interior of B.

The next lemma is a consequence of Lemma 6.

Lemma 7 *$d_B - \hat{d} \leq 2L/k$, where L is the total cost of edges in the graph.* □

Lemma 8 *For each i, C_i contains weight at most 1/2.*

Proof: Suppose some dual cycle C_i contained more than half the weight. It would then contain in particular the dual cycle C, since more than half of the weight is contained in C. Thus C_i is an ancestor of C. Since C_i has edge-cardinality less than $k/2$, C_i is an ancestor of B, the deepest ancestor of C having such a small edge-cardinality. This contradicts the fact that C_i's depth exceeds that of B. □

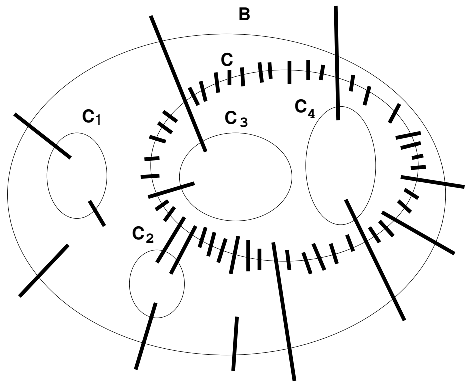


Figure 4: C is the deepest dual cycle containing weight more than 1/2. Note that C may have high edge-cardinality. B is C's deepest ancestor having edge-cardinality less than $k/2$. It contains weight more than 1/2. The dual cycles $C_1, \ldots, C_r$ are at the first level deeper than B such that they have edge-cardinality less than $k/2$.

Obtain a graph $\hat{G}$ from G as follows. For each i, consider the dual cycle C_i. Corresponding to this dual cycle, place a circuit of face edges in G (just inside the dual cycle), where each face edge connects the endpoints of a primal edge. Then delete all other vertices enclosed by this new circuit, thus turning the circuit into a face. Assign the weight of the deleted nodes to this face. We call the new face a *leaf face.* (This process is depicted in Figure 5.)

Note that the circuits are vertex-disjoint because each one connects vertices within a different dual cycle C_i, and the interiors of the C_i's are disjoint. Moreover, by Lemma 8, each leaf face has weight at most 1/2.

The dual cycle B is treated similarly; add a circuit of face edges just *outside* this dual cycle, connecting together those endpoints of the primal edges lying outside the dual cycle. Then delete the other vertices in the exterior of the dual cycle. These deletions cause the circuit to be the boundary of the infinite region, which we call the *root face.* The weight assigned to the root face is the total weight of the exterior vertices deleted. Note that since B contained more than weight 1/2, the weight of the root face is at most 1/2.

5.2 Extracting the spanning tree for the subgraph

Starting with the shortest-path tree T, we obtain a spanning tree $\hat{T}$ for the subgraph $\hat{G}$ we have extracted. For each vertex that lies inside B and outside

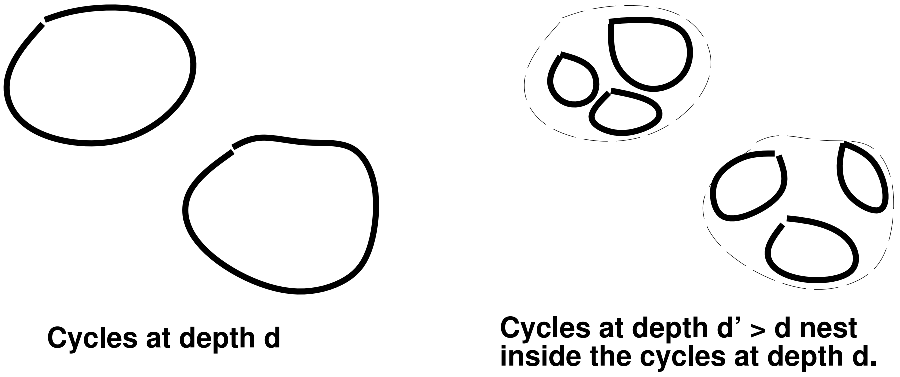


Figure 3: This figure shows how cycles nest as distance increases.

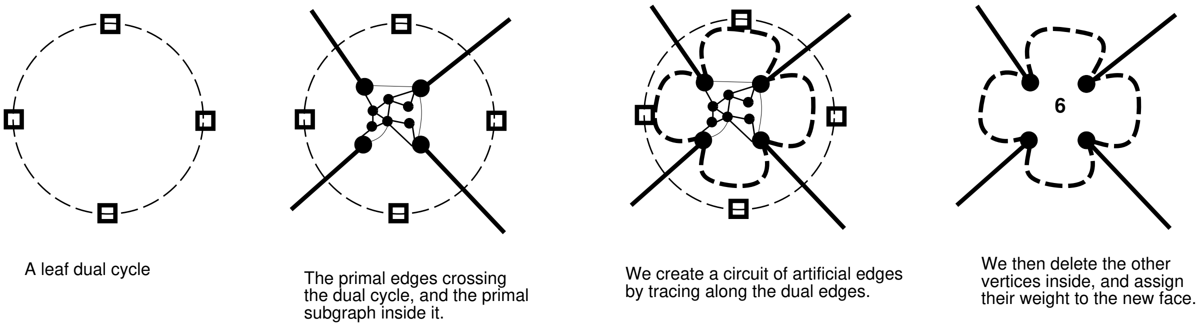


Figure 5: The construction applied to each dual cycle C_i to obtain a leaf face.

every C_i, the vertex's parent edge in $\hat{T}$ is defined to be the vertex's parent edge in T. For each but one of the vertices in the circuit outside B, we take its parent edge to be the face edge connecting it to the next vertex in counterclockwise order. The one exception is to be the root of $\hat{T}$.

We similarly treat the dual cycles C_i. For each, we determine the interior vertex closest to the root, and we assign that vertex's parent edge to be its parent edge in T. For each of the other interior vertices, we take its parent edge to be the face edge connecting it to the next vertex in clockwise order.

5.3 Applying Lipton and Tarjan's lemma We have obtained a subgraph $\hat{G}$ and a spanning tree $\hat{T}$. We add more face edges to $\hat{G}$ to triangulate every face except the root and leaf faces just created, and we apply Lemma 5 to obtain a cycle separator in $T^* \cup e$ where e is a non-tree edge.

We claim that the cycle separator uses face edges from at most one leaf face. To see this, note that the face edges from each leaf face form a terminal path in T^*: no real edge has such an edge as an ancestor. Furthermore, since the leaf faces correspond to edge-disjoint co-cycles, no edge connects two vertices from different leaf faces.

Thus the cycle separator consists of two paths of original edges, joined by at most k face edges: fewer than $k/2$ from a leaf face, fewer than $k/2$ from the root face, and possibly the non-tree edge e. By Lemma 7, each of the paths of original edges has length at most $2L/k$. Thus the separator has all the promised properties.

6 The Steiner version of planar TSP

An obvious next problem to study is a "Steiner"-type weighted-planar-graph TSP problem: given a weighted planar graph and a set of terminals (which is a subset of the set of vertices), find a minimum-cost circuit that visits all the terminals. This problem is an obvious generalization of the planar-graph TSP, and it also generalizes the Euclidean TSP. To see the latter, note that given n points in $\Re^2$ on which we desire a salesman tour, we can draw all $\binom{n}{2}$ line segments between them and add a new node at each intersection. This gives a weighted planar graph on $O(n^4)$ vertices (the costs on the edges are the Euclidean lengths) in which it suffices to find a minimum-cost circuit for the original set of n vertices.

Most of our techniques generalize to the Steiner-type problem, except for the spanner result (Theorem 1). Thus the existence of a PTAS would follow if the following conjecture is true.

Conjecture 1 *There exists a function $f(\cdot)$ such that: given $\varepsilon > 0$, a weighted planar graph G, and a subset S of vertices, there exists an edge-induced subgraph G' which $(1+\varepsilon)$-approximates all internode distances in S, and furthermore G' has total edge weight at most $f(\varepsilon)$ times the minimum Steiner tree weight for S.*

Even in the absence of a spanner result, we can give a quasipolynomial-time approximation scheme for the Steiner variant by combining the techniques of this paper with that of Arora's original PTAS for Euclidean TSP [2]. In each step the algorithm tries a small (poly(n) size) family of separators, one of which is guaranteed to work. The approximation scheme runs in $n^{\mathrm{poly}(\log n, 1/\varepsilon)}$ time.

The approximation scheme uses a separator algorithm that identifies not one but many separators; one of them must be good in a sense to be described. First we describe the algorithm for finding the set of separators for a subgraph H of the input graph.

The separator algorithm first finds a shortest-path tree of H rooted at an arbitrary vertex. For each vertex v in H, let $d(v)$ denote the distance of v from the root. Let $v_1, v_2, \ldots, v_n$ be the vertices of H in increasing order of distance from the root. For each pair of vertices v_i, v_j $(i < j)$, define $H(v_i, v_j)$ to be the graph obtained by coalescing $v_1, \ldots, v_i$ to a single vertex, and deleting $v_j, \ldots, v_n$. The coalescing can be done via edges of the shortest-path tree, so planarity is preserved.

Apply Lemma 5 to this graph, with artificial edges added temporarily to triangulate every face. The result is a cycle separator consisting of two paths in the shortest-path tree and a (possibly artificial) non-tree edge. Let $P(v_i, v_j)$ be the set of edges in these paths. Let $C(v_i, v_j)$ denote the vertex-partition $(\{v_1, \ldots, v_i\}$, *Interior*, *Exterior*, $\{v_j, \ldots, v_n\})$, where *Interior* and *Exterior* denote the set of vertices of $H(v_i, v_j)$ in the interior and exterior of the cycle separator. Note we have $O(n^2)$ such partitions.

The following lemma can be proved using Lemma 6.

Lemma 9 *Let OPT_H be the set of those edges of the optimal tour that lie in H, and let $k > 0$. There exists a pair v_i, v_j of vertices such that*

- *the number of edges in OPT_H having precisely one endpoint in $\{v_1, \ldots, v_i\}$ is at most $k/2$,*
- *the number of edges in OPT_H having precisely one endpoint in $\{v_j, \ldots, v_n\}$ is at most $k/2$,*
- *the total cost of the edges in $P(v_i, v_j)$ is at most $4/k$ times the cost of all edges in OPT_H.*
- *Each part of $C(v_i, v_j)$ has at most $2/3$ of the vertices of H.*

Now we describe the approximation scheme. Let k be $c \log n/\varepsilon$, where c is a constant to be determined and n is the number of vertices in the input graph. The approximation scheme makes use of a recursive algorithm that takes three inputs: a subgraph H of the input graph G, and an ordered set S of edges and vertices that connect H to the remainder of G. The output is an approximately min-cost way of routing the tour within H in such a way that it enters and leaves H via the elements of S in the specified order.

The algorithm first applies the separator algorithm to H, which yields $O(n^2)$ separators specified by a 4-way vertex partition $C(v_i, v_j)$ and a set $P(v_i, v_j)$ of edges forming two paths.

For each such 4-way partition $(A, \mathit{Interior}, \mathit{Exterior}, B)$, the algorithm first contracts the edges in the corresponding two paths, obtaining two vertices x and y. Let H' be the contracted graph. Next, the algorithm enumerates orderings S' of k-subsets of edges within H (together with the vertices x and y). For each block of the 4-way partition, the algorithm makes a recursive call to find a subtour within the corresponding subgraph of H' that is consistent with the orderings S and S'. The subtours of H' are combined, and then lifted to get a subtour of H, using the same patching scheme as was sketched in the beginning of Section 3.

The best subtour of H obtained in this way is returned. Lemma 9 guarantees that one of the choices of 4-way partition and choices of S' will lead to a good subtour being returned.

The error analysis again comes down to bounding the total cost of all edges contracted in the recursive calls contributing to the best tour found. The choice of k (including the constant c) ensures that the total error is at most $\varepsilon\, OPT$, where OPT is the cost of the optimal tour.

The depth of our recursion is again $O(\log n)$, and we consider $n^{O(k)}$ subproblems for every problem, so the total running time is $n^{O(k \log n)}$, or $n^{O(\log^2 n/\varepsilon)}$. Note that dynamic programming won't help us much with this algorithm, since the size of S may reach $O(k \log n)$ (perhaps a weighting scheme would help again here).

Open problems

We suspect that there is a polynomial-time algorithm for the Steiner version of the planar TSP.

Another open problem is to find a nearly linear time approximation scheme for the planar-graph TSP. The "charging" argument in [3] strongly uses the geometry of the plane and does not seem to apply to the graph metric case.

References

[1] I. Althofer, G. Das, D. Dobkin, D. Joseph, L. Soares. On sparse spanners of weighted graphs. *Disc. Computational Comp. Geo.*, **9**:1, 1993.

[2] S. Arora. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of 37th IEEE Symposium on Foundations of Computer Science*, pp. 2-12, 1996.

[3] S. Arora. Nearly linear time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of 38th IEEE Symposium on Foundations of Computer Science*, pp. 554–563, 1997.

[4] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *JACM*, **41**(1), 1994. Preliminary version in IEEE FOCS 1983.

[5] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. In J.F. Traub, editor, *Symposium on new directions and recent results in algorithms and complexity*, page 441. Academic Press, NY, 1976.

[6] M. Grigni, E. Koutsoupias, and C. H. Papadimitriou. An approximation scheme for planar graph TSP. In *Proc. IEEE Symposium on Foundations of Computer Science,* pp 640–645, 1995.

[7] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys. The traveling salesman problem. John Wiley, 1985.

[8] R. Lipton and R. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math*, **36**:177-189, 1979.

[9] R. Lipton and R. Tarjan. Applications of a planar separator theorem. *SIAM J. Comp.*, **9**(3):615–627, 1980.

[10] G. Miller. Finding small simple cycle separators for 2-connected planar graphs. *JCSS*, **32**:265–279, 1986.

[11] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: Part II– A simple PTAS for geometric k-MST, TSP, and related problems. Preliminary manuscript, April 30, 1996. *To appear in SIAM J. Computing.*

[12] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *J. of Computer and System Sciences* **43**, pp. 425–440, 1991.

[13] C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research* **18**, pp. 1–11, 1993.