An Approximation Scheme for Planar Graph TSP

Michelangelo Grigni[*]
mic@mathcs.emory.edu
Dept. of Mathematics and C.S.
Emory University
Atlanta, GA 30322

Elias Koutsoupias[†]
elias@cs.ucla.edu
Computer Science Department
Univ. of California, Los Angeles
Los Angeles, CA 90095

Christos Papadimitriou[‡]
christos@cs.ucsd.edu
Computer Science Department
Univ. of California, San Diego
La Jolla, CA 92093–0114

November 22, 1995

Abstract

We consider the special case of the traveling salesman problem (TSP) in which the distance metric is the shortest-path metric of a planar unweighted graph. We present a polynomial-time approximation scheme (PTAS) for this problem.

[*]Initial work performed at UCSD, supported by NSF grant No. DMS-9206251.
[†]Work supported by NSF grant No. CCR-9521606.
[‡]Work supported by NSF grant No. CCR-9301031.

1 Introduction

The traveling salesman problem [5] (TSP) has been the testbed of every new algorithmic idea during the past half-century: linear programming, cutting planes and polyhedral combinatorics, probabilistic algorithms, local search (and more recently Boltzmann machines, genetic algorithms, simulated annealing, and neural nets), dynamic programming, Lagrangean relaxation, even NP-completeness. Approximability was no exception: The TSP was shown early to be non-approximable in its general case, and the 3/2 approximation algorithm for the triangle inequality case due to Christofides [2] is well-known. It has been open for two decades whether this bound can be improved, even in two important special cases: The *Euclidean TSP*, and the *graph TSP* —in the latter the distance metric is the shortest-path metric of an unweighted graph. It was shown in [11] that another special case, in which all distances are either one or two (1-2 TSP), can be approximated within 7/6, and also that it is MAXSNP-hard [10] (and, as it easily follows from the proof, so is the graph TSP).

It has also been open whether the Euclidean TSP is MAXSNP-hard[1]. The current work was motivated by unsuccessful attempts to prove that it is. *We now conjecture that the Euclidean TSP has a polynomial-time approximation scheme.* The results in the current paper lead us to this conjecture in two ways: First, the planar graph TSP (defined in the next paragraph), for which we provide a PTAS, seems a necessary intermediate step in any MAXSNP-hardness reduction to the Euclidean TSP. Second, it seems plausible that the techniques used for deriving our PTAS, appropriately extended, would be of use in a PTAS for the Euclidean case.

In the *planar graph TSP* we are given a planar graph and we are looking for the shortest closed walk that visits all nodes at least once. Equivalently, we are considering the TSP with distance metric the shortest-path metric of this graph. Our main result is an algorithm which, for any given planar graph with n nodes and error bound $\epsilon > 0$, returns a tour of length at most $\epsilon \cdot n$ above the optimum (and thus within an $(1+\epsilon)$ multiplicative factor of the optimum), in time $n^{O(1/\epsilon)}$. Our algorithm is very much unlike the approximation algorithms discovered in the wake of the original planar separator

[1] Due to a widespread misunderstanding of the results in [11] it has been often reported that it is.

theorem [6, 7], and the approximations by outerplanar decomposition [1]. It is based on dynamic programming and a novel planar separation theorem. To introduce the basic ideas, we describe below a simpler *quasipolynomial-time approximation scheme* for this problem.

Fix some parameter f depending only on n (the number of vertices in G) and ϵ. Consider the $f/2$ largest faces of the given planar graph G, and triangulate them (each from a vertex). We now have a planar graph with largest face $d = O(n/f)$. According to Miller's planar separator theorem [9] (Theorem 3.2), there is a simple cycle of length $\sqrt{nd} = O(n/\sqrt{f})$ that separates G into two graphs with no more than $2n/3$ nodes each. Now if the triangulation edges are removed, this cycle (with at most two triangulation edges per face) breaks up into f paths that still separate G. We contract these path segments into single nodes, and call the result G'. The set I of contracted nodes in G' is shared by the interior and exterior subgraphs G_1 and G_2.

Consider the optimum tour of G. It crosses each path segment of the separator either an even or an odd number of times, and these parities are preserved in G'. Since we do not know which of the 2^{f-1} cases prevails, we must solve each of the following 2^{f-1} subproblems in G_1 and G_2, one for each even subset X of I: "Cover the nodes of G_i with paths between the endpoints in X." We solve each of these problems recursively, and select the set X that minimizes the sum of costs of the two path covers. Our approximate tour for G' is now the union of the two path covers, which may however contain up to $f/2$ cycles. The approximate tour in G uses the edges of the approximate tour in G', plus at most two copies of each contracted edge to cover the contracted vertices. Finally we patch these $f/2$ cycles together into a single tour, at an additional cost of at most f edges.

This argument is repeated down through a graph decomposition, stopping at subgraphs of size $S = O(f^2)$, which are solved exhaustively. We may then show that the overall additive error is $O(n/f + (n \log n)/\sqrt{f})$ (terms from patching edges and contracted edges, respectively), which forces us to take $f = \Theta((\log^2 n)/\epsilon^2)$. The running time obeys the recurrence

$$T(n) = 2^f \cdot T(\frac{2n}{3}) + n,$$

with base case $T(S) = 2^{O(S)}$, with the solution $T(n) = 2^{O(S)} = n^{O((\log^3 n)/\epsilon^4)}$.

Three ideas are needed to turn this quasipolynomial-time approximation scheme into a true PTAS (each idea removes a factor of $\log n$ from the exponent). First, instead of adapting Miller's planar separator theorem to our needs, we prove our own planar separator theorem (Theorem 3.1), possibly of interest in its own right; this allows us to balance the patching and contraction error terms with a smaller f parameter. Our theorem separates a planar graph by a cycle of A ordinary edges and B "faces-edges," subject to a multiplicative trade-off between the two—Miller's theorem provides only an $A \cdot \sqrt{B}$ trade-off. Second, we use a weighing scheme to avoid crowding too many collapsed "constraint points" into any one subproblem; this makes dynamic programming useful. Third, we use a modified algorithm to handle the base case problems, in roughly $2^{O(\sqrt{S})}$ time.

2 The Algorithm

We are given a connected planar graph G on n vertices, with some embedding. We are also given a constant $\epsilon > 0$. Our goal is an algorithm that runs in time $n^{O(1/\epsilon)}$ and outputs a tour T whose total weight $w(T)$ is within a factor of $1 + \epsilon$ of the weight $w(T^*)$ of the minimum weight tour. In fact we know $n \leq w(T^*) < 2n$, so it suffices to stay within an additive error of $\epsilon\, n$.

Our algorithm uses divide and conquer, together with dynamic programming. First, in the *decomposition stage*, we build a decomposition tree of the graph G into contracted connected subgraphs, each subgraph with $O((\log n)/\epsilon)$ *constraint points*. This part of the algorithm is fully polynomial-time (in fact nearly linear), independent of ϵ. Second, in the *approximation stage*, in bottom-up order we compute for each subgraph a table of $n^{O(1/\epsilon)}$ approximate solutions. Each table is computed from the tables of the two children subgraphs.

2.1 Decomposition Stage

For our decomposition, we need the following theorem for finding cycle separators in planar graphs. This is a special case of Theorem 3.1, together with a tree-separator argument on the 2-connected components and bridge-points of graph H. Let $|H|$ denote the *size* of H, its number of vertices. A *face-edge* in a planar graph is a simple arc that connects two vertices through the interior of a face.

Theorem 2.1 *Given a connected embedded planar graph H with weights on its vertices and a parameter f such that $1 \leq f \leq \sqrt{|H|}$, there is a simple planar cycle C through $O(|H|/f)$ vertices of H such that at most f of the arcs of C are face-edges, the remaining arcs are ordinary edges, and the interior and exterior of C both have at most 2/3 of the total weight. Such a C may be found in polynomial time (in fact nearly linear time).*

Starting from the input graph G, we iteratively apply Theorem 2.1 to construct a binary decomposition tree $\mathcal{T}$ where each node is a contracted subgraph of G. At the beginning we fix two parameters depending only on the initial size $|G| = n$ and the target approximation ratio ϵ. Parameter $f = \Theta((\log n)/\epsilon)$ is used as in Theorem 2.1, and parameter $S = \Theta(f^2)$ is a stopping size used to define the leaves of $\mathcal{T}$.

At the root of $\mathcal{T}$ we have G itself. Let all the vertices of G have weight one, so the total weight of G is $W(G) = n$. Theorem 2.1 gives a simple cycle C, consisting of $p \leq f$ paths in G connected by p "face-edges" jumping across some faces of G. We construct two children graphs from G as follows (see Figure 1). First contract each path to a point, resulting in graph G' (remove any self-loops and multiple edges to keep G' simple). The cycle C becomes a simple cycle C' in G', consisting of p vertices connected by face-edges. Let G_1 be the subgraph of G' induced by vertices on or interior to C'. Similarly define G_2 by vertices on or exterior to C'. These graphs G_1 and G_2 are the children of G in $\mathcal{T}$. There is one exception to this construction: if some vertex on C' would become isolated in G_i, then we do not include it in G_i.

The following lemma simplifies our discussion; the proof is delayed to the end of Section 3.

Lemma 2.2 *When C is constructed as in Section 3, the graphs G_1 and G_2 are connected planar graphs.*

Graphs G_1 and G_2 each have at most f *constraint points* (the white circles in Figure 1). Every vertex of G was either contracted into a constraint point, or inherited as an *original vertex* in exactly one of G_1 or G_2. In each of the two children, we assign weight $W(G)/6f$ to each constraint point.

We repeat this construction recursively. That is, let H be a graph in the tree. If H has size at most S vertices (counting both original and constraint

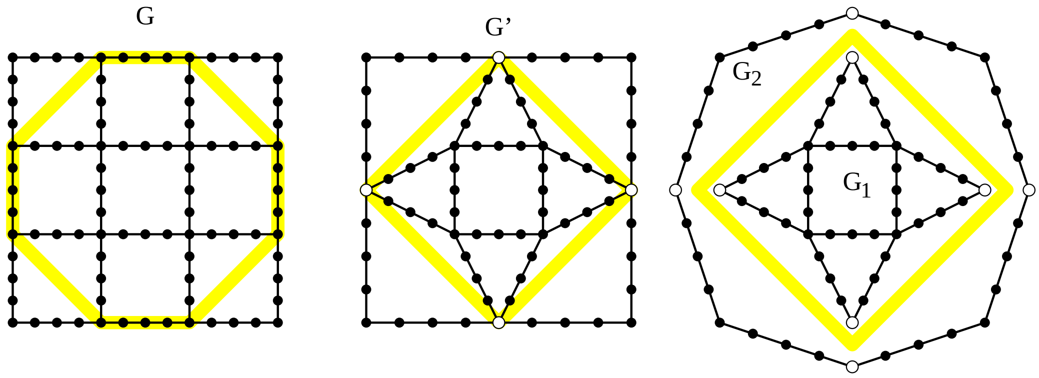

Figure 1: A separating cycle in G (shaded), the contracted graph G', and the two child graphs G_1 and G_2.

points), it is a leaf of $\mathcal{T}$. Otherwise, we already have weights on the vertices of H defined during the split at its parent; let $W(H)$ be the sum of those weights. Apply Theorem 2.1 again (with the same f) to construct the separator C. Contract the paths in C and split H to construct H_1 and H_2 as above; they are the children of H in $\mathcal{T}$. Note that each H_i $(i = 1, 2)$ contains both old constraint points (that were already present in H) and new constraint points (created during this split). Give every constraint point in H_i the weight $W(H)/6f$. That is, old constraint points from H get new weights when inherited by a child.

Claim 2.3 *Suppose H_i is a child of H in $\mathcal{T}$, then $W(H_i) \leq 5/6\ W(H)$.*

Proof: This is clearly true at the root when $H = G$ has no constraint points: the interior (respectively exterior) vertices inherited by H_1 (resp. H_2) have weight at most $2/3\ W(G)$, and the new constraint points (at most f of them) add weight at most $f \cdot (W(H)/6f) = W(H)/6$. Now by induction down the tree, old constraint points inherited from H are reweighted with smaller weights in H_i, thus the reweighted interior (or exterior) vertices in H_i still have weight at most $2/3\ W(H)$, so the same inequality works at every level. $\square$

This has two immediate consequences. First, since constraint points are so heavy, any H in $\mathcal{T}$ has at most $5f = O((\log n)/\epsilon)$ constraint points. Second, since $S > 5f$, every non-leaf contains some original vertex (a vertex from G) with weight one, thus the tree has depth at most $D = \log_{(6/5)} n < 4 \lg n$.

We need to bound the sum of the sizes of all the leaf graphs in $\mathcal{T}$. Clearly every original node from G appears in at most one leaf, so our problem is to bound the total number of *constraint points* that appear in the leaves.

Consider tree $\mathcal{T}'$, which is tree $\mathcal{T}$ with all its leaves removed; $\mathcal{T}'$ may have many nodes with only one child. We want to bound the total size (in vertices) of all the leaves of $\mathcal{T}'$. Every leaf of $\mathcal{T}'$ has at least S vertices, of which at most $5f$ are constraint points, and hence not from the original n vertices from G. Set $S = 10Dcf/\epsilon$, for some large enough constant c (since $D = \Theta(\log n)$, this is consistent with our earlier statement that $S = \Theta(f^2)$). The constraint points are at most an $\epsilon/2cD$ fraction of all vertices in each leaf graph, so there are at most $n\epsilon/cD = o(n)$ constraint points in all the leaves of $\mathcal{T}'$.

Now consider the leaves of $\mathcal{T}$. We simply bound the number of leaves of $\mathcal{T}$ by one plus the number of all nodes in $\mathcal{T}'$, which is at most D times the number of leaves in $\mathcal{T}'$. Thus by charging D leaves of $\mathcal{T}$ to each leaf of $\mathcal{T}'$, we see that in all of the leaves of $\mathcal{T}$ there are at most $(\epsilon/c)n$ constraint points. Thus there are at most $(1 + \epsilon/c)n$ vertices in all the leaves of $\mathcal{T}$.

Note finally that the construction of $\mathcal{T}$ took polynomial time (in fact nearly linear), independent of the parameter ϵ.

2.2 Approximation Stage

We build approximate solutions up from the leaves of $\mathcal{T}$, using a simple form of dynamic programming. First we define a slightly more general optimization problem that we must approximately solve.

Suppose H is a connected graph, and X is a subset of its vertices, with $|X|$ even. An (H, X)-*solution* is a collection $\mathcal{P}$ of paths in H that visit every vertex of H at least once, and such that their boundary is exactly X. That is, their endpoints are in X, and each vertex in X is an endpoint exactly once. Or in the special case that X is empty, $\mathcal{P}$ should contain a single tour that covers H (the usual TSP). Let $c^*(H, X)$ be the minimum cost (total length of all the paths) of any (H, X)-solution.

For each graph H in $\mathcal{T}$, let $c(H) \leq 5f$ be the number of constraint points in H. For every subset X of these constraint points, we want to find an (H, X)-solution $\mathcal{P}$ with total cost approximating $c^*(H, X)$. We will store these approximate solutions in a table $T[H, X]$. For a fixed H, we have a subtable of size $2^{c(H)-1} = n^{O(1/\epsilon)}$.

We begin at the leaves. A leaf graph H has size at most $S = \Theta((\log^2 n)/\epsilon^2)$. We use a simpler approximation scheme described in Section 2.4, returning a solution within $\epsilon/4$ fraction of the optimum. We repeat this for every subset X of the the constraint points. The base case computation takes time $n^{O(1/\epsilon)}$.

At an internal graph H with children H_1 and H_2, consider the children as subgraphs of H', which is H with its separating paths contracted. The set N of at most f new constraint points is common to both children. For each subset X of constraint points of H, define X' to be the "mod-2 contraction" of X. That is, we include those vertices in X' which correspond to an odd number of vertices from X. In particular X' outside N is identical to X outside the cycle. Now define X_1 on H_1 as the restriction of X' to H_1, and define $X_2 = X' - X_1$ (which we think of as a constraint set in H_2). Thus $X' = X_1 \oplus X_2$ in H' ($\oplus$ denotes symmetric difference).

Now consider all subsets Y of N of the same parity as X_1 (and hence X_2). Choose Y^* to be the Y that minimizes the sum of the costs of the two solutions $T[H_1, X_1 \oplus Y]$ and $T[H_2, X_2 \oplus Y]$. We construct our approximate (H, X)-solution $T[H, X]$ as follows:

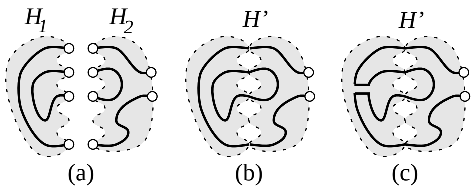

Figure 2: Patching solutions from H_1 and H_2 in H'.

- Put the two child solutions $T[H_1, X_1 \oplus Y^*]$ and $T[H_2, X_2 \oplus Y^*]$ together in H'. Their endpoints match up so that we can join paths and exactly meet the endpoint constraints of X' on H'. However, we may have created at most $f/2$ extra tours in H'. Since H' is connected, we may splice these loops back into a proper (H, X)-solution using at most

f edges and some Eulerian tour techniques. This is an approximate solution for H', costing at most f more than the sum of the two child solutions.

This patching process is illustrated in Figure 2. In (a) we have approximate solutions in H_1 and H_2, meeting the endpoint constraints indicated by open circles. When these solutions are joined in (b), the resulting tours in H' require further patching along an edge of H', resulting in single tour (c). In this example, H' itself has two endpoint constraints.

- Given the approximate (H', X')-solution, extend it to an (H, X)-solution by using each contracted edge at most twice, and meeting the original contracted constraints of X. That is, each contracted path can is doubled to make a tour, and that tour can be patched in to the solution in H' (which visits at least one vertex of this tour). Furthermore we can meet any contracted endpoint constraints by appropriately breaking the tour.

  The total cost over all the contracted paths in H is $2|C| = O(|H|/f)$. The result is our approximate solution stored in $T[H, X]$.

We repeat the above for every subset X of constraint points in H, before working further up the tree. Finally we output the approximate solution $T[G, \emptyset]$.

2.3 Analysis

The running time of the above dynamic program is clearly $n^{O(1/\epsilon)}$, times a factor of $n^{O(1/\epsilon)}$ to handle the base cases (the base case time is slightly different for small n, see the end of Section 2.4). The main problem is to analyze the quality of this approximation. There are three sources of error to bound:

Leaf Error: At leaf problems, it suffices to use an approximation algorithm with relative error $\epsilon/4$ (Section 2.4).

Patching Error: When patching the child solutions together in H', the cost is $O(f)$ units. This includes $f/2$ patching edges to join tours (formed by merging solutions) into adjacent paths, and $O(f)$ further inefficiency because cut points were visited by both child solutions, but they only need to be visited once.

Contraction Error: When uncontracting the solution from H' to H, we used up to $2|C| = O(|H|/f)$ new edges.

The patching error allows us to recover an approximate solution in H' from approximate solutions in H_1 and H_2, with additive error $O(f)$. Let X' be a subset of constraint points of H'. An unknown optimal solution $\mathcal{P}^*$ of cost $c^*(H', X')$ crosses each new cut point (contracted path) an even or odd number of times. By simple planar tour techniques, we may rearrange $\mathcal{P}^*$ so that each cut-point is crossed either once or zero times. Note that $\mathcal{P}^*$ restricted to H_1 and H_2 creates valid solutions of the corresponding constrained problems; actually these solutions may not be entirely valid, because they may fail to visit the cut points in H_1 or H_2. Since $2f$ additional edges can fix this, we have that $c^*(H_1, X_1 \oplus Y) + c^*(H_2, X_2 \oplus Y) - 2f \leq c^*(H', X')$. On the other hand, our algorithm will eventually try the correct subset Y and therefore its cost $c(H, X)$ is at most $c(H_1, X_1 \oplus Y) + c(H_2, X_2 \oplus Y) + f$, where the final f term is what we may have to pay to splice any unwanted tours into adjacent paths. Therefore, the additional patching error is at most $3f$.

The contraction error lets us recover an approximate (H, X)-solution from an approximate (H', X')-solution (here X' is the contraction of the constraint set X). The error bounds are summarized in the following.

Lemma 2.4 *Suppose G is a connected graph, X is a subset of its vertices, and P is a path in G. Let $G \backslash P$ denote graph G with path P contracted to a single point p.*

1. *If $|P \cap X|$ is even, then $c^*(G\backslash P, X - P) \leq c^*(G, X) \leq c^*(G\backslash P, X - P) + 2|P|$.*
2. *If $|P \cap X|$ is odd, then $c^*(G\backslash P, X - P + \{p\}) \leq c^*(G, X) \leq c^*(G\backslash P, X - P + \{p\}) + 2|P|$.*

Now to bound the total error of our final solution, it suffices to sum the additive errors at each step.

For the leaf error, we use our estimate that the sum of the sizes of all leaves of $\mathcal{T}$ is at most $(1 + \epsilon/c)n$, where c is some constant. Thus for large enough c, the total leaf error is at most $\epsilon n/2$.

For the patching error, we note $|H| \geq S = \Theta(f^2)$, so again a large enough c will keep the patching error at each H less than the contraction error at H.

For the contraction error, consider a single level of $\mathcal{T}$. The total size of subgraphs at this level is $n + o(n)$, and for each split H the size of the cycle

C_H is $O(1/f)$ (the hidden constant is from Theorem 3.1). Thus the total error for one level is $O(n/f)$. Finally, the tree has depth $D < 4 \lg n$, so the total contraction error over the entire tree is $O(n \log n/f)$. By an appropriate choice of $f = \Theta((1/\epsilon)\log n)$, this is at most $\epsilon n/4$.

Thus the total additive error is at most ϵn, which is close enough. Note there was no circularity in choosing the constants. First of all D did not depend on any constant choices. Second f is chosen depending only on D, ϵ, and the hidden constant in Theorem 3.1. Finally S is chosen large enough to make everything else work.

Remark. The algorithm could further decompose each subgraph H into its 2-connected components, and then solve the constrained subproblems on these components. This would introduce no new error terms, since to find a (constrained) optimal solution in a connected graph H, it suffices to find solutions in each 2-connected component, and then greedily splice these together.

2.4 The Base Case

For the base case problem, we have a connected planar graph H of size at most $S = \Theta((\log n)^2/\epsilon^2)$, with at most $5f = \Theta((\log n)/\epsilon)$ constraint points. For every subset X of the constraint points, we want to compute an (H, X)-solution with cost within $\epsilon/4$ of the minimum $c^*(H, X)$. Assuming $\log n \geq 1/\epsilon$, we will do this in $n^{O(1/\epsilon)}$ time.

Our strategy is very similar to the above algorithm (decomposition followed by dynamic programming), with the following modifications:

- In the decomposition we stop at leaves of size less than $S' = (c' \log n)/\epsilon = \Theta(f)$, where c' is some large enough constant, and solve those problems exactly.
- All vertices (original vertices and constraint points) have weight one.
- Instead of using a fixed parameter (f in the main algorithm) in Theorem 2.1 for every subgraph K, we simply look for a cycle separator in K with at most $\sqrt{|K|}$ face-edges and $O(\sqrt{|K|})$ vertices.
- By the previous choice of cycle size, the contraction error from splitting K will be within a constant factor of the patching error bound (the

constant hidden in Theorem 2.1). Hence for our analysis, it will suffice to bound the patching error.

We could even use the original Lipton-Tarjan planar separator theorem [6, 7], except that it is convenient for us to keep the subgraphs connected by Lemma 2.2.

Fix H, and build the decomposition tree $\mathcal{T}$ of H as before. Let graph K be any node in $\mathcal{T}$, and let K_i be a child. Since we are using an unweighted separator, $|K_i| \leq 3/4|K|$. We use $3/4$ rather than $2/3$ here to allow some slack for the vertices that K_i gets from the separator. Thus the size of graphs decreases geometrically with depth in $\mathcal{T}$, and so does the number of new cut points at each split. Hence for any K in $\mathcal{T}$, the total number of constraint points (cut points from all ancestors) in K is $O(f)$. Suppose we have chosen $S' = \Theta(f)$ large enough so that at least half the vertices in a leaf graph are still ordinary vertices.

We wish to bound the overall number of constraint vertices created at all levels of $\mathcal{T}$, since this will bound the patching error, and hence also the contraction error. If an ordinary vertex v survives (without ever being contracted) to a leaf graph, call it a *survivor*. Consider a non-leaf graph K in $\mathcal{T}$; it contains at least $|K|/2$ survivors. To account for the $2\sqrt{|K|}$ new constraint points created by splitting K, we will charge $4/\sqrt{|K|}$ to each survivor in K. If we trace a single survivor v down through $\mathcal{T}$, we see that these charges are geometrically increasing, so that the total charge given v is $O(1/\sqrt{S'})$ (within a constant of its last charge). Summing over all survivors, the total number of constraint points created in $\mathcal{T}$ is $O(|H|/\sqrt{S'})$, which is $O(\epsilon|H|)/\sqrt{c'}$ (recall that here we assume that ϵ is at least $1/\log n$).

Since the total patching error, and thus the total error, is in the worst case proportional to the total number of constraint points, we can choose large enough c' to make the total relative error at most $\epsilon/4$.

Finally, we need a way to find an *optimal* (K, X)-solution when K is a connected planar graph of size $S' = O(\log n/\epsilon)$, and X is arbitrary (perhaps $|X|$ is half of $|K|$). Note that to find such a solution $\mathcal{P}$, it suffices to guess its multi-set of edges. That is, if $\mathcal{E}$ is a multi-set of edges in K such that its mod-2 boundary is X, $\mathcal{E}$ covers the vertices of K, and every edge in $\mathcal{E}$ is connected (through $\mathcal{E}$) to some element of X (or if X is empty, $\mathcal{E}$ spans K), then by Eulerian tour techniques we can recover a solution $\mathcal{P}$ using exactly the edges of $\mathcal{E}$. Notice that each edge appears at most twice in the optimal solution. Since K is planar, it has at most $O(|K|)$ edges and we simply need to test all $3^{O(|K|)} = n^{O(1/\epsilon)}$ such multi-sets of edges.

Remark. The preceding analysis assumed $\log n \geq 1/\epsilon$. For smaller n, we should choose $S' = c'(1/\epsilon)^2$, giving total time $2^{O(1/\epsilon^2)}$. In other words, when we do not treat ϵ as a constant, the general running time for the base case (and indeed for the entire algorithm) is $\left(n + 2^{1/\epsilon}\right)^{O(1/\epsilon)}$.

3 Yet Another Planar Separator Theorem

In the following, let G be an embedded 2-connected planar graph with n vertices. Suppose C is a Jordan curve that passes through some of the vertices, and the arcs of C (its arc components between the vertices) are either edges of G or *face-edges*: arcs that pass through the interior of some face of G. We call such a curve a *V-cycle* (vertex cycle), and its *size* is the number of vertices it passes through. The faces, edges, and vertices of G not traversed by C fall into either its interior or its exterior. For example, the V-cycle in the G of Figure 1 has size twenty, four face-edges, and five faces of G in its interior.

Suppose the vertices, faces, and edges of G are weighted, so that all weights are nonnegative, at most $2/3$, and sum to one. If the interior and exterior of V-cycle C both have weight at most $2/3$, we call C a *V-cycle separator.*

Theorem 3.1 *Let G be an 2-connected planar graph with weights as above, and $1 \leq f \leq \sqrt{n}$. Then there exists a V-cycle separator C with size $O(n/f)$ and $O(f)$ face-edges, and such a C may be found in polynomial time.*

This theorem applied to the dual graph allows us to extend the claim to larger values of f: we can always find a "cycle" of $O(f)$ faces and $O(n/f)$ vertices whose removal separates G. The theorem and its proof are closely related to the following [9] (also [4]):

Theorem 3.2 (Miller) *Let G be a weighted 2-connected planar graph with maximum face size d. Then G has a simple cycle separator C of size $O(\sqrt{nd})$, and such a C may be found in polynomial time.*

Theorem 3.1 implies part of Theorem 3.2: set $f = \sqrt{n/d}$, and replace the face-edges of C with paths around the boundaries of the crossed faces. The resulting cycle has size $O(\sqrt{nd})$, but further work is required to make it simple. Also, Theorem 3.2 implies a weaker version of Theorem 3.1: for each face of size greater than $d = n/f^2$, choose a bounding vertex and triangulate the face with "pseudo-edges" from that vertex. Now find a simple cycle separator of size $O(\sqrt{nd}) = O(n/f)$; the pseudo-edges become face-edges in the original graph, at most two in each of the $O(f^2)$ large faces.

Proof: Given a 2-connected planar graph G, we give it a *vertex-face labeling* as follows. Pick an arbitrary root face f_0, and give it label $l(f_0) = 0$. Next, give all vertices v around f_0 the label $l(v) = 1$. Next, give each unlabeled face f around a level 1 vertex the label $l(f) = 2$. Repeat this until all vertices and faces are labeled. This is simply a breadth first search (BFS) in the vertex-face graph of G: the vertex-face graph is a bipartite planar graph whose points are the vertices and faces of G, with the obvious adjacencies (the four-sided faces of the vertex-face graph correspond to edges of G). If we only consider the faces (the even levels), the labeling mimics a BFS from f_0 where faces are "adjacent" iff they share a vertex, as done in [9].

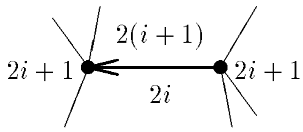

Figure 3: An F-boundary edge of G at level $2i+1$.

Consider an edge e in G. It has two vertices v_1, v_2 and two faces f_1, f_2. By the properties of BFS, either $l(v_1) \neq l(v_2)$ or $l(f_1) \neq l(f_2)$ may occur, but not both. If $l(f_1) \neq l(f_2)$, then we call e an *F-boundary edge* at level $l(e) = l(v_1) = l(v_2) = 2i+1$, since it separates faces at levels $2i$ and $2(i+1)$. Furthermore such edges have an orientation, say with the smaller face label "to the left" (Figure 3). A local argument shows that the F-boundary edges around a vertex must alternate in and out. Thus the F-boundary edges at level $2i+1$ decompose uniquely into a collection of non-nesting edge-disjoint simple oriented cycles (F-cycles).

Define the *interior* of a simple cycle as the side containing f_0. Then F-cycles at level $2i+1$ all have f_0 in their interiors, and have disjoint exteriors. Furthermore, F-cycles at different levels are vertex disjoint, and an F-cycle at level $2i+3$ lies in the exterior of a unique F-cycle at level $2i+1$. Let $\mathcal{T}$ be the partition tree whose nodes are the F-cycles ordered by their exteriors, and rooted at F_0, the level 1 boundary edges of f_0.

We next identify a subtree $\mathcal{T}'$ of $\mathcal{T}$, rooted at an F-cycle F and with leaf F-cycles L_1, ..., L_r, such that:

- each of these bounding F-cycles have size $O(n/f)$;
- $\mathcal{T}'$ has diameter $O(f)$;
- F has interior weight at most $2/3$;
- each L_j has exterior weight at most $2/3$.

This tree pruning is done as shown by Miller [9]. If any of these F-cycles is a separator, then we are done. So for the remainder of the proof, assume that F has exterior weight greater than $2/3$, and each L_j has interior weight greater than $2/3$.

Prune G as follows: replace the interior of F and the exterior of each L_j with a new *pseudo-face* of the same weight; call these new faces $f(F)$, $f(L_1), \ldots, f(L_r)$. Give these faces labels $l(f(F)) = l(F) - 1$ and $l(f(L_j)) = l(L_j) + 1$. The resulting graph G' now has a vertex-face labeling rooted at the pseudo-face $f(F)$, and its F-cycle tree is exactly $\mathcal{T}'$.

We further modify G': for each original face f (that is, not a pseudo-face) choose a *parent vertex* $v(f)$ with label $l(f)-1$, and add internal "face-edges" triangulating f from $v(f)$. Call the result G''. Divide the weight of each subdivided face f among its triangles, and give the triangles the same label $l(f)$. This labeling is a vertex-face labeling of G'' rooted at $f(F)$, and furthermore the vertex labels are now consistent with an ordinary BFS in G'' starting from the vertex set of F.

We construct a spanning tree T in G'' as follows:

- For each pseudo-face, add all its edges but one.
- For each vertex v not on a pseudo-face, choose a *parent face* $f(v)$ with label $l(f) = l(v) - 1$, and connect v to its grandparent $v(f(v))$, which has label $l(v) - 2$.
- For each leaf pseudo-face $f(L_j)$, connect one vertex to a grandparent, chosen as above.

Unlike $\mathcal{T}'$, tree T does not have $O(f)$ diameter, because of its long paths around the pseudo-faces. Any simple path in T involves at most $O(f+n/f) = O(n/f)$ vertices and $\text{diam}(\mathcal{T}') = O(f)$ face-edges of G (edges added in the construction of G'').

Any edge of G'' induces a simple cycle in T; say its "interior" is the side containing the root pseudo-face $f(F_0)$. If the cycle bounds a single face of G'', we call it a *leaf face*. Note that we constructed T so that each pseudo-face is a leaf face. We summarize some conditions on T and G'':

- T is a spanning tree in planar graph G''.
- Every non-leaf face of G'' is a triangle.
- T has an induced cycle (F) with exterior weight greater than $2/3$.
- T has an induced cycle (any L_j) with interior weight greater than $2/3$.

Under these conditions, a simplified version of Theorem 5 from [9] shows that some induced cycle C of T is a simple cycle separator for G'', and hence (by the choice of weights) also a V-cycle separator of G. The bounds on simple paths in T above give us the claimed bounds of $O(n/f)$ size and $O(f)$ face-edges; all other edges of C are ordinary edges in G. □

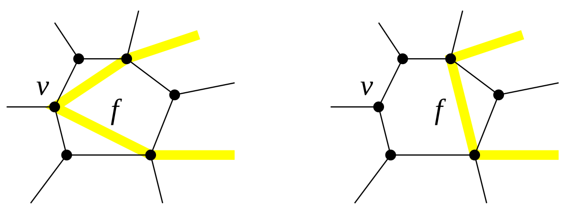

Figure 4: Splicing two consecutive face-edges.

Remarks. A review of the above proof shows that there is at most one face f of G with two face-edges in C, and those two edges (if they exist) are consecutive through the "parent vertex" v of the face. All other faces of G have at most one face-edge of C. This observation implies that when our algorithm splits G into G_1 and G_2 (and throws out the one isolated parent vertex if it is isolated in G_1 or G_2), it is effectively treating the two consecutive face-edges as the one spliced face edge in Figure 4. Hence the effective split of G uses at most one face-edge per face of G. To make G_1 or G_2 disconnected, the cycle would have to use two face-edges in some face. Hence the two contracted subgraphs G_1 and G_2 will remain connected.

To prove Theorem 2.1, we must consider the case when the given weighted graph H is connected but not necessarily 2-connected. In this case we consider the bipartite tree whose vertices are the 2-connected components and articulation points of G. By a standard tree-separator argument, we find a 2/3-separating vertex in this tree. If separating vertex is an articulation point v of H, let C be a cycle through v and the exterior face. Otherwise some 2-connected component G of H is a good tree separator, and we apply Theorem 3.1 to this component, with all weights outside G pushed to the articulation points on G. This tree-argument preserves the connectivity remarks of the previous paragraph, so we also have Lemma 2.2.

4 Discussion

The main open problem suggested by our work is whether there is a PTAS for the Euclidean TSP. *We conjecture that there is.* A first step in this direction would be a PTAS for the *weighted* planar graph TSP. To handle this extension, our ideas must be enhanced with a more sophisticated balancing scheme, as well as with clever rounding-off methods. It could be that such an algorithm, applied to the Delaunay triangulation, already improves on Christofides' algorithm. Finally, to approach the Euclidean TSP, we may need planar separator theorems for *planar point sets*.

It is also not clear whether Euclidean TSP is MAXSNP-hard in three or higher dimensions, since the hard examples of [11] are expanding graphs, and any set of well-spaced points in Euclidean space will instead have sublinear separators, when defined appropriately [8, 3].

References

[1] B. S. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153, 1994. See also 24th

FOCS, 1983.

[2] N. Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. In J. F. Traub, editor, *Sympos. on New Directions and Recent Results in Algorithms and Complexity*, page 441, New York, NY, 1976. Academic Press.

[3] D. Eppstein, G. L. Miller, and S.-H. Teng. A deterministic linear time algorithm for geometric separators and its applications. In *Proc. 9th Annual ACM Symposium on Computational Geometry*, pages 99–108, 1993.

[4] Hillel Gazit and Gary L. Miller. Planar separators and the euclidean norm. In *1st International Symposium on Algorithms*, pages 338–347, 1990.

[5] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* Wiley, New York, 1985/1992.

[6] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM Journal on Applied Mathematics*, 36:177–189, 1979.

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

[8] G. L. Miller, S.-H. Teng, and S. A. Vavasis. A unified geometric approach to graph separators. In *Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 538–547, 1991.

[9] Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *Journal of Computer and System Sciences*, 32:265–279, 1986.

[10] Papadimitriou and Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43, 1991. Also in STOC'88.

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