A linear-time approximation scheme for TSP in undirected planar graphs with edge-weights[*]

Philip N. Klein
Brown University

July 9, 2009

Abstract

We give an algorithm requiring $O(c^{1/\epsilon^2}n)$ time to find an ϵ-optimal traveling salesman tour in the shortest-path metric defined by an undirected planar graph with nonnegative edge-lengths. For the case of all lengths equal to 1, the time required is $O(c^{1/\epsilon}n)$.

1 Introduction

The traveling salesman problem is often the first problem researchers use to test a new optimization technique [32]. In a metric space, a *tour* is a cycle $(v_0\ v_2\ \ldots\ v_{n-1})$ of the points of the metric space, and the weight of the tour is the sum $\sum_{i=0}^{n} \text{dist}(v_i, v_{(i+1) \bmod n})$, where $\text{dist}(u, v)$ is the distance between u and v. The goal is to find the minimum-weight tour. The problem is MAXSNP-hard [36, 37] in arbitrary metric spaces, and the best approximation ratio known, that proved by Christofides[14], is 1.5. For the shortest-path metric of an unweighted planar graph (one in which every edge has weight one), Grigni, Koutsoupias, and Papadimitriou [23] gave an algorithm that requires $n^{O(1/\epsilon)}$ to find a $1+\epsilon$-optimal tour. Thus for fixed ϵ, the algorithm runs in polynomial time. Such a family of polynomial-time algorithms is called an *approximation scheme*.

Arora, Grigni, Karger, Klein, and Woloszyn [5] subsequently gave a polynomial-time approximation scheme for the more general problem in which the planar graph's edges have arbitrary

[*]A preliminary version was published in *Proceedings of the IEEE Symposium on Foundations of Computer Science* (2005), pp. 647–656.

nonnegative weights. Their algorithm requires $n^{O(\epsilon^{-2})}$ time. Both algorithms are somewhat complicated, and involve a recursive decomposition using new planar-separator lemmata. The latter paper introduced the idea of using a *spanner* result to handle edge-weights.

Arora [3] and Mitchell [34] had shown that a PTAS exists for *Euclidean TSP* (i.e., the subcase in which the points lie in $\Re^2$ and distance is measured using the Euclidean metric). This PTAS finds an ϵ-optimal tour in $n^{O(1/\epsilon)}$ time. Arora [4, 2] improved the running time of his algorithm to $O(n \cdot (\log n)^{O(1/\epsilon)})$, using randomization. Finally, Rao and Smith [38] gave a PTAS for the two-dimensional Euclidean case that takes time $O(\epsilon^{-O(\epsilon)} n + n \log n)$. (Their algorithm also used a spanner result.) The latter two approximation schemes are said to be *efficient* polynomial-time approximation schemes (EPTAS) because the time can be bounded by a function of ϵ times a polynomial function of n. Thus for an EPTAS, the degree of the polynomial does not grow with $1/\epsilon$.

In view of the fact that an ϵ-optimal tour can be found in the Euclidean case in time that is polynomial with a fixed degree, independent of ϵ, it seems natural to ask whether the same holds true for the planar case. In this paper, we answer this question.

Theorem 1 *There is an algorithm that, for any $\epsilon > 0$ and any planar graph G with nonnegative edge-weights, finds a $1+\epsilon$-optimal tour. The running time is $O(c^{1/\epsilon^2} n)$ where c is a constant. For the special case where all weights are 1, a similar algorithm requires $O(c^{1/\epsilon} n)$ time.*

Marx [33] subsequently showed that the running time for the unit-weight case is essentially optimal under a widely held complexity assumption.

1.1 Other related work

In a seminal paper, Baker [6] gives a method for obtaining PTASs for a variety of optimization problems in planar graphs, e.g. maximum-weight independent set and minimum-weight vertex cover. The resulting algorithms are linear time (for fixed ϵ). The key idea (interpreted in modern parlance) is to turn a problem in a planar graph to a problem in a graph with bounded treewidth.

Grigni and Sissokho ([24], building on [25]) have given a quasipolynomial approximation scheme for weighted TSP in minor-excluded graphs. This paper proved a spanner result for minor-excluded graphs. Berger, Czumaj, Grigni, and Zhao ([7], building on [16]) give a PTAS for the problem of finding a minimum-weight 2-edge-connected spanning multi-subgraph[1] of an edge-weighted planar graph, and a quasipolynomial approximation scheme for finding a

[1]Duplicate edges of the input graph are allowed in the solution.

minimum-weight 2-edge-connected or biconnected spanning subgraph[2] of an edge-weighted planar graph. This paper introduced a new spanner construction.

Demaine and Hajiaghayi [17] describe a framework for PTASs that is based on the notion of *bidimensionality*. They derive approximation schemes for subclasses of minor-excluded graphs that involve turning the input graph into a low-treewidth graph. Their results apply to graphs that are not planar. Their framework can be viewed as a way to generalize Baker's approach so as to derive algorithms for nonlocal problems, such as feedback vertex set and connected dominating set. For planar graphs in particular, they derive EPTASs for several unit-weight problems. In relation to their framework, our result is an example of how one can more thoroughly exploit planarity to derive a fast and simple EPTAS.

For a positive number s, an s-spanner of a graph G is a subgraph of G that approximately preserves the node-to-node distances of G: for any pair u, v of nodes of G, the distance in the subgraph must be at most s times the distance in G. There is a vast literature on spanner constructions. In this paper, we require a construction for $1+\epsilon$-spanners of planar graphs. Henceforth, for brevity we use the term *spanner* and omit mention of the parameter $1+\epsilon$.

1.2 The approach

The TSP approximation scheme consists of the following steps.

Spanner step: Delete some edges of the input graph while approximately preserving the optimal value.[3]

Slicing step Using breadth-first search in the planar dual together with a shifting argument, identify subgraphs (called *slices*). The weight of edges belonging to more than one slice is at most $1/k$ times the weight of the graph, and each connected component of each slice has a spanning tree of depth at most $k+1$, where k is a parameter.

Dynamic-programming step: Use dynamic programming to find an optimal solution in each connected component of each slice.

Combining step: The union of the tours found in the previous step comprises a tour for the original graph.

The time required by the dynamic-programming step is exponential in k. We show that there is a choice of k that depends only on ϵ for which the resulting tour is nearly optimal.

[2]No duplicates are allowed.

[3]This was the also the first step in [5] and subsequently in, e.g., [24] and one of the algorithms of [7].

In the preliminary version of this paper [30], a slightly different algorithm was described. In the second step, a procedure called *thinning* was applied to the planar dual of the graph. Thinning involves deleting edges; thinning the planar dual corresponds to *contracting* edges in the primal. Thinning in either the primal or the dual results in a graph with small branch-width. The method of thinning in the dual graph is novel, though quite simple. One nice way to formulate the result is as follows:

> For any positive integer k, there is a partition of the edges of a planar graph into k sets such that contracting the edges in any one of the sets yields a graph with bounded treewidth (where the bound depends on k).

This formulation of the result is due to Demaine, Hajiaghayi, and Mohar [19], who learned of this result from the preliminary version of this paper and subsequently generalized the result to apply to graphs of any bounded genus.

Because of the potential applicability of the planar result and of its role in subsequent developments, we provide a proof in Section 7

The approach used in this version of the paper to formulate the TSP approximation scheme, which we call *slicing*, emerged from joint work with Borradaile and Mathieu [10, 11]. This formulation does not require the algorithm to perform any contractions, which leads to a simpler algorithm.

The general approach used for TSP has proved useful in obtaining approximation schemes for other problems in planar graphs, including minimum-weight two-edge-connected spanning multi-subgraph,[4] TSP on a subset of the nodes [31], minimum-weight two-edge-connected spanning subgraph [8], and Steiner tree [10]. As mentioned above, the basic technique has been generalized [19] to apply to bounded-genus graphs, giving rise to new approximation schemes for such graphs.

1.3 Spanner step

The spanner step requires an algorithm that, given a n-node planar graph G_0 with edge-weights and given a parameter ϵ, deletes edges so as to obtain a graph G such that

S1: weight$(G) \leq \rho_\epsilon \cdot$ OPT(G_0)

S2: OPT$(G) \leq (1+\epsilon)$OPT(G_0), and

where OPT(G) is the value of the optimum for input graph G, and weight(G) is the sum of weights of edges in G.

[4]An $O(c^{1/\epsilon}n)$ algorithm for this problem can be obtained from the TSP algorithm by modifying the dynamic program.

We refer to the first step as *spanner step* because of the connection to s-spanners. An s-*spanner* of a graph G_0 is a subgraph G of G_0 with the same set of nodes, such that, for any pair u, v of nodes, the u-to-v distance in G is at most s times the u-to-v distance in G_0. As discussed in Lemma 9, to achieve Property S2 in the case of TSP, it suffices that G be a $1 + \epsilon$-spanner of G_0. In Section 3, we discuss a spanner construction that also achieves Property S1.

An n-node planar graph G_0 with no parallel edges or self-loops has at most $3n$ edges. For unit-weight edges, OPT(G_0) is at least n, so weight$(G_0) \leq \rho_\epsilon$OPT(G_0) holds for ρ_ϵ=3. In this sense, a trivial spanner result suffices for the unit-weight case.

We remark that Properties S1 and S2 can be considered for optimization problems other than TSP, and indeed for problems where a traditional s-spanner would not suffice. We propose use of the term *spanner result* to refer more generally to a construction achieving Properties S1 and S2. We have obtained such constructions for two other problems in planar graphs, leading to approximation schemes for these problems. The first problem [31] is a generalization of the problem studied here; the tour must visit a specified subset of nodes of the input graph (not necessarily all the nodes). The second problem [10] is Steiner tree, in which one seeks a minimum-weight tree spanning a specified subset of nodes.

2 Preliminaries

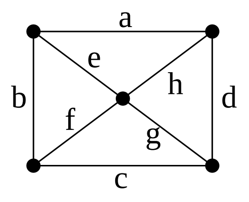
In this section, we describe the basic definitions and results on planar embeddings and planar duals. Most of the material is standard in concept, but the notation may be unfamiliar, and we also introduce a variant of contraction that we call *compression*, and state some related results. In Subsection 2.5, we give some definitions and results that help us reformulate the TSP.

For a rooted tree T and a node v that is not the root of T, the *parent edge* of v is the edge of T that connects v to its parent.

2.1 Combinatorial embeddings

The traditional geometric definition of planar embeddings involves drawings of a graph on the plane. Proofs and algorithms become simpler when one uses an alternative definition of embedded planar graphs, a combinatorial definition. See [35].

The idea of a combinatorial embedding was implicit in the work of Heffter [27]. Edmonds [21] first made the idea explicit, and Youngs [46] formalized the idea. A combinatorial embedding is sometimes called a *rotation system*. The idea is to represent at each node the arrangement of edges around that node, as illustrated in Figure 1.



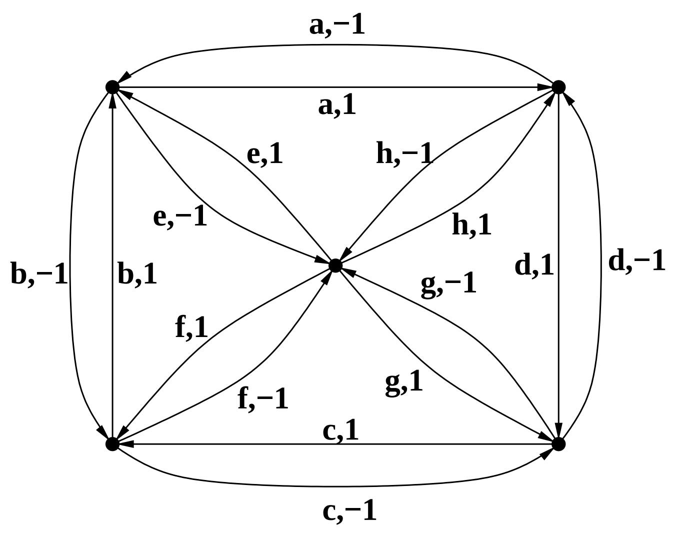


Figure 1: The first figure shows an undirected planar graph embedded in the plane, with its edges labeled. The *rotation* corresponding to the top-left node is the permutation cycle $(a\ b\ e)$, indicating that the edges a, b, and e are incident to that node, and are arranged counterclockwise around that node in the order a, b, e. Similarly, the rotation in the middle node is $(e\ f\ g\ h)$.The second figure shows the same undirected graph but with darts instead of edges. There are two oppositely directed darts for each (undirected) edge. The darts corresponding to edge e are $\langle e, 1\rangle$ and $\langle e, -1\rangle$. The rotation corresponding to a node consists of the darts that point away from the node. Thus the rotation corresponding to the top-left node is $(\langle a, 1\rangle\ \langle b, -1\rangle\ \langle e, -1\rangle)$. The rotation corresponding to the middle node is $(\langle e, 1\rangle\ \langle f, 1\rangle\ \langle g, 1\rangle\ \langle h, 1\rangle)$.

However, it is convenient to represent at each node not just which edges are incident to the node and in what order, but more specifically which ends of which edges are incident to the node. For example, if e is a self-loop (an edge whose endpoints are the same), the edge e would appear twice in the arrangement of incident edges, and it is helpful to be able to distinguish these two occurrences. We will refer to the ends of an edge as its *darts*, as we explain next.

For any given finite set E, we can interpret E as a set of edges, and we define $E \times \{\pm 1\}$ to be the corresponding set of *darts*. For each edge e, the darts of e, namely $\langle e, 1\rangle$ and $\langle e, -1\rangle$, represent the two opposite orientations of e. The edge of $\langle e, i\rangle$ is e. We define $\text{rev}(\cdot)$ (*rev* is short for *reverse*) to be the function that takes each dart to the corresponding dart in the opposite direction: $\text{rev}(\langle e, i\rangle) = \langle e, -i\rangle$.

We define an embedded graph on E to be a pair $G = \langle \pi, E\rangle$ where π is a permutation of the darts of E. The permutation cycles of π are called the *nodes* of G. Note that nodes are defined in terms of edges, rather than the other way round. This definition precludes isolated nodes. Each node v is a permutation cycle $(d_1\ d_2\ \ldots\ d_k)$.

For a graph G, we use $V(G)$, $E(G)$, and $D(G)$ to denote the node-set, the edge-set, and the dart-set of G. We use the same notation for subgraphs of G.

For a dart d of G, we define the tail of d in G, denoted $\text{tail}_G(d)$, to be the permutation cycle of π containing d. (We may omit the subscript when doing so creates no ambiguity.) We define $\text{head}_G(d) = \text{tail}_G(\text{rev}(d))$. The tail and head of a dart d are called the *endpoints* of d, and also the endpoints of the edge of d.

A *walk* of darts in G is a sequence $d_1 \ldots d_k$ of darts such that, for $i = 2, \ldots, k$, $\text{head}_G(d_{i-1}) = \text{tail}(d_i)$.[5] The *start* of the walk is $\text{tail}_G(d_1)$ and the *end* is $\text{head}_G(d_k)$. It is a *closed* walk if in addition $\text{head}_G(d_k) = \text{tail}_G(d_1)$. It is a simple path/cycle (cycle if closed, path if not) if no node occurs twice as the head of a dart. The walk, path, or cycle is said to contain an edge e if it contains a dart of e. It is said to contain a node v if v is the head or tail of some dart in the sequence. We define $\text{rev}(d_1 \ldots d_k) = \text{rev}(d_k) \ldots \text{rev}(d_1)$. A walk/path whose start is u and whose end is v is called a u*-to-*v walk/path.

To define the faces of the embedded graph, we define another permutation π^* of the set of darts by composing π with rev: $\pi^* = \pi \circ \text{rev}$. Then the *faces* of the embedded graph $\langle \pi, E\rangle$ are defined to be the permutation cycles of π^*. (See Figure 2.) Note that a face of G can be interpreted as a closed walk in G.

Note that this definition diverges from the traditional geometric definition of faces in the case of a disconnected graph. In that case, according to the definition considered here, for each

[5]Note that, even though we are concerned with undirected graphs, we use (directed) darts in our definition of walks because they provide more information about the structure of the walks.

connected component there will be a different external face. (In fact, this is necessary if one wishes to preserve the desirable property that the dual of the dual is the primal.)

2.2 Planarity

We say that an embedding π of a graph G is *planar* if it satisfies Euler's formula: $n-m+\phi = 2\kappa$, where n=number of nodes, m=number of edges, ϕ=number of faces, and κ=number of connected components. In this case, we say $G = \langle \pi, E\rangle$ is a *planar embedded graph*. We say a graph is a *planar graph* if there is a planar embedding for it.[6] Finding a planar embedding for a planar graph is a well-studied problem, and linear-time algorithms are known.[7], so we assume throughout this paper that every planar graph comes equipped with an embedding. It follows from Euler's formula that an n-node planar graph with no parallel edges has $O(n)$ edges.

2.3 Duality

The *dual* of a connected embedded graph $G = \langle \pi, E\rangle$ is defined to be the embedded graph $G^* = \langle \pi^*, E\rangle$. The permutation cycles of π^* are the faces of G. (See Figure 3.) According to this definition, the edge set of the dual is identical to the edge set of the original graph (called the *primal*). This identification of primal edges and dual edges is mathematically and notationally convenient (albeit sometimes confusing).

Since rev $\circ$ rev is the identity, $(\pi^*)^* = \pi$, we obtain the following.

Proposition 1 $G^{**} = G$.

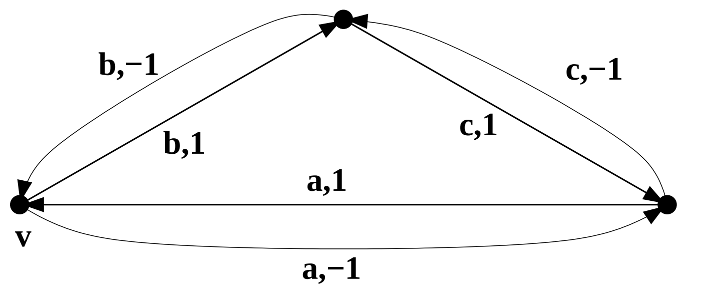
It can be shown that the dual of a connected graph is connected. It follows that the connected components of G^* correspond one-to-one with the connected components of G. Hence if G satisfies Euler's formula then so does G^*. Thus the dual of a planar embedded graph is a planar embedded graph.[8]

Let T be a spanning tree of G. For an edge $e \notin T$, there is a unique simple cycle consisting of e and the unique path in T between the endpoints of e. This cycle is called the *elementary cycle* of e with respect to T in G.

[6]For the purpose of the current result, all we need is that every graph embeddable on an orientable surface of genus zero has a combinatorial embedding that satisfies Euler's formula. However, it is known (see, .e.g, [35]) more generally that for any graph embedded on a closed, orientable surface, the corresponding combinatorial embedding determines the geometric embedding up to homeomorphism.

[7]The first was due to Hopcroft and Tarjan [29]. See [12] for a discussion of later work.

[8]For disconnected graphs, this definition of dual diverges from the geometric definition in that it assigns multiple dual nodes to a single region of the sphere/plane. According to the geometric definition, the dual of a graph is always connected. However, choosing that definition means giving up, for example, the nice property that $G^{**} = G$.



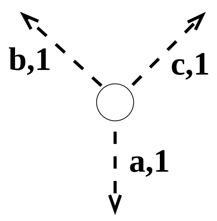


Figure 2: The figure on the left shows the dart representation of part of a graph. We can trace out the face containing the dart $\langle a,1\rangle$ as follows. First apply rev, obtaining the dart $\langle a,-1\rangle$ emanating from the node v. Next, apply π, obtaining $\langle b,1\rangle$, the dart after $\langle a,-1\rangle$ in the permutation cycle corresponding to v. This shows that $\langle b,1\rangle$ is the successor to $\langle a,1\rangle$ in the face. We apply the same process to $\langle b,1\rangle$, obtaining its successor $\langle c,1\rangle$ and that dart's successor in turn, $\langle a,1\rangle$. The face (permutation cycle of $\pi \circ$ rev) is thus $(\langle a,1\rangle\ \langle b,1\rangle\ \langle c,1\rangle)$. The figure on the right shows the corresponding fragment of the dual graph. There is a dual node corresponding to the face discussed above. The permutation cycle corresponding to this dual node is exactly the permutation cycle comprising the face: $(\langle a,1\rangle\ \langle b,1\rangle\ \langle c,1\rangle)$. However, we follow the convention of drawing the dual in such a way that the permutation cycle gives the *clockwise* order of darts. This convention helps when drawing the dual superimposed on the primal, for it enables us to draw primal and dual edges at approximately right angles to one another, as shown in Figure 3. The convention does not affect the dual graph as a mathematical object, only its depiction.

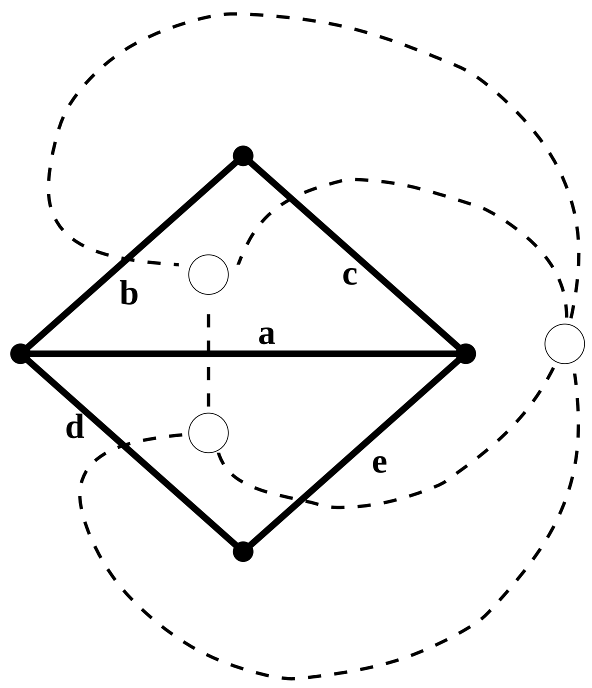


Figure 3: The first figure shows a graph (the solid nodes and edges) and, superimposed, its planar dual (the open nodes and dashed edges).

For a spanning tree T of G, we denote by T^* the set of edges of G that are not in T. The following is a classical result.

Proposition 2 *If G is a planar embedded graph and T is a spanning tree of G, then T^* is a spanning tree of G^*.*

We refer to T^* as the tree dual to T.

If $S \subseteq V(G)$, we use $\Gamma_G(S)$ to denote the set of edges e such that in G the edge e has one endpoint in S and one endpoint not in S. A set of this form is called a *cut* of G. Note that $\Gamma_G(S) = \Gamma_G(V(G) - S)$.

If S is connected in G and $V(G) - S$ is connected in G, we call $\Gamma_G(S)$ a *bond*.

Proposition 3 *If G is a planar embedded graph, the edges of a bond in G form a simple cycle in G^* and vice versa.*

It follows from Proposition 3 that every simple cycle C in G defines a bipartition of the faces of G; namely the bipartition $(S, V(G) - S)$ where $E(C) = \Gamma_{G^*}(S)$.

Let f_∞ be a face of G. We call f_∞ the *infinite face* by analogy to geometric embeddings. For combinatorial embeddings, the choice is arbitrary.[9]

We say the simple cycle C *encloses* a face f with respect to f_∞ if f belongs to the set S such that $E(C) = \Gamma_{G^*}(S)$ and $f_\infty \notin S$. We say that C encloses an edge with respect to f_∞ if the edge belongs to a face enclosed by C, and that it strictly encloses the edge if in addition the edge does not belong to C.

Lemma 4 *Let G be a connected planar embedded graph, let T be a rooted spanning tree of G, let v be a nonroot node of T, and let e be the parent edge of v. Then the elementary cycle of e in G^* with respect to T^* consists of the edges of $\Gamma_G($descendents of v in $T)$.*

Proof: Removing e from T breaks T into two connected components, one containing the descendents of v in T, and one containing the non-descendents. It follows that the cut Γ_G(descendents of v in T) is a bond, and therefore, by Proposition 3, the edges in that cut form a simple cycle C in G^*. The only edge of $E(T)$ belonging to $E(C)$ is e, so C consists of e together with a simple path of edges not in $E(T)$ connecting the endpoints of e in G^*. The edges not in $E(T)$ are in $E(T^*)$, so the simple path is a simple path in T^*. This proves the lemma. $\square$

2.4 Deletion and compression

We discuss two ways of removing edges from an embedded graph, deleting and compressing, both of which preserve the embedding (and preserve planarity). Compressing an edge is very similar to the operation of contracting the edge (the difference arises when the edge is a self-loop).

Deleting an edge e of an embedded graph $G = \langle \pi, E\rangle$ is an operation that produces the graph $G' = \langle \pi', E'\rangle$ where $E' = E - \{e\}$ and, for each dart of E',

$$\pi'[d] = \begin{cases} \pi[\pi[\pi[d]]] & \text{if } \pi[d] \text{ and } \pi[\pi[d]] \text{ are the darts of } e \\ \pi[\pi[d]] & \text{if } \pi[d] \text{ is a dart of } e \text{ but } \pi[\pi[d]] \text{ is not} \\ \pi[d] & \text{otherwise} \end{cases}$$

For a set S of edges, we denote by $G - S$ the embedded graph obtained by deleting the edges of S. The order of deletion does not affect the final embedded graph. It is easy to see that deletion preserves planarity.

Proposition 5 *An edge e is a self-loop of G iff it is a cut-edge of G^*.*

[9]For geometric intuition, consider that a planar graph can be embedded on the surface of a sphere. According to this embedding, every face is finite.

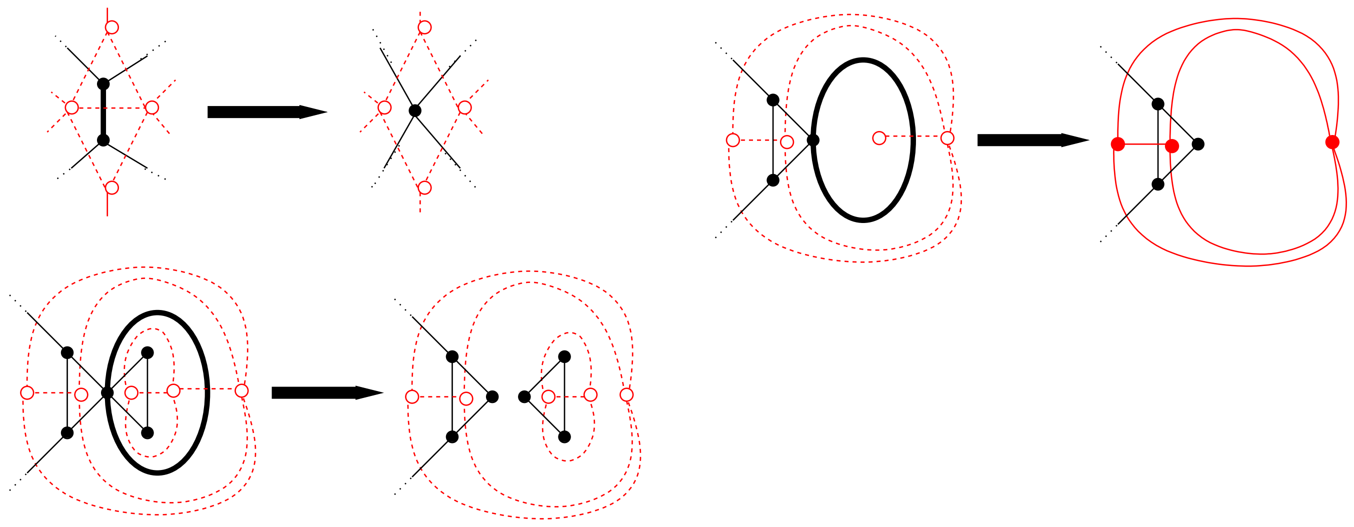


Figure 4: Three examples of compression. The graph with solid edges and solid nodes is the primal. The graph with dashed edges and open nodes is the dual. The edge being compressed is signified by a heavy line.

We define edge *compression* to be deletion in the dual. That is, compressing an edge e of G is an operation that produces the graph $(G^* - \{e\})^*$. We denote the result as $G/\{e\}$. Since deletion preserves planarity and the dual of a planar embedded graph is a plane graph, compression preserves planarity. The operations of deletion and compression commute.

Figure 4 illustrates the effect of edge compression on the underlying graph in three examples. If e is not a self-loop in G then the effect of compressing e in G is to contract e as shown in the top left diagram. The thick line represents the edge to compress. If e is a self-loop in G, so a cut-edge in G^*, and is not the only edge incident to either of its endpoints in G^* then the effect is to duplicate v, as shown in the bottom left diagram; one copy has as its incident edges those edges that in G are incident to v and strictly enclosed by e (with respect to some designated face f_∞) and the other copy has as its incident edges those edges that in G are incident to v and not enclosed by e (and not equal to e). If e is a self-loop in G and is the only edge incident to one of its endpoints in G^*, the effect is to delete e.

2.5 Preliminaries related to TSP

For an assignment weight$(\cdot)$ of nonnegative weights to the edges of G and a set S of edges, define weight$(S) = \sum\{\text{weight}(e) \;:\; e \in S\}$. For a subgraph H, define weight$(H) = $ weight$(E(H))$.

For the metric space of shortest paths in a graph, a tour corresponds to a closed walk in the graph that visits every node. The weight of the tour is the sum of weights of the edges comprising

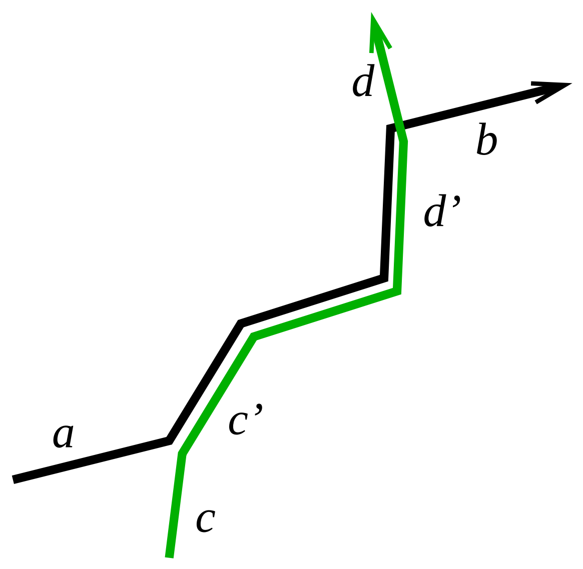


Figure 5: The light walk forms a crossing configuration with the bold walk.

the walk, counting multiplicities. For a connected graph G, let OPT$(G,$ weight$)$ be the minimum weight of such a tour. (We omit the second argument when doing so creates no ambiguity.)

Lemma 6 *For any walk W in a graph, there is a walk W' that visits the same nodes as W, such that every edge used by W' is used by W, and occurs at most twice in W'.*

Proof: Let W be a closed walk in G, and suppose some dart d occurs at least twice in W. Write $W = W_1\ d\ W_2\ d$. Then W_1 rev(W_2) is a closed walk of G that visits the same nodes as W but uses dart d fewer times. Repeating this step yields the lemma. $\square$

Lemma 6 shows that, in seeking the minimum-weight walk visiting a given set of nodes, we can restrict ourselves to considering walks in which each edge occurs at most twice.

Let W be a walk, and let $P = a\ W\ b$ and $Q = c\ W\ d$ be walks that are identical except for their first and last darts. Let c' be the successor of c in Q and let d' be the predecessor of d in Q. We say Q forms a *crossing configuration* with P (see Figure 5) if the permutation cycle at head(c) induces the cycle $(c\ c'$ rev$(a))$ and the permutation cycle at tail(d) induces the cycle $($rev$(d')\ b\ d)$.

We say a walk P *crosses* a walk Q if a subwalk of P and a subwalk of Q form a crossing configuration. The following folklore result was used by Arora et al. in [5].

Proposition 7 *For any tour in a planar graph, there exists a tour that visits the same nodes and comprises the same darts in the same multiplicities, and does not cross itself.*

Proof: Suppose $\hat{W} = W_1 \; a \; W \; b \; W_2 \; c \; W \; d$ is a closed walk where $c \; W \; d$ forms a crossing configuration with $a \; W \; b$. Then $d \, W_1 \; a$ rev$(c \; W_2 \; b)$ rev$(W) \; W$ is a closed walk visiting the same nodes and comprising the same darts in the same multiplicities, and with one fewer crossing configurations. □

Proposition 7 shows that we can restrict our attention to non-self-crossing walks.

An *Eulerian graph* is a graph G with the following properties.

- G is connected, and
- every node of G has even degree.

Perhaps the best-known result in graph theory is the following:

Proposition 8 *A graph G is Eulerian iff there is a walk in G in which each edge occurs exactly once.*

Such a walk is called an Eulerian cycle. There is a linear-time algorithm that, given an Eulerian graph, finds an Eulerian cycle.

A graph H is a *multi-subgraph* of G if H can be obtained from a subgraph of G by duplicating some edges. We call it a *bi-subgraph* if the maximum multiplicity of any edge is at most two.

It follows from the Eulerian characterization that finding a minimum-weight tour in a graph G is equivalent to finding a minimum-weight Eulerian multi-subgraph of G that includes every node of G. Lemma 6 shows that furthermore it suffices to find a minimum-weight Eulerian bi-subgraph that includes every node.

We slightly generalize the notion of Eulerian multi-subgraph to handle disconnected graphs. For a possibly disconnected graph G, we say H is a *multi-Eulerian multi-subgraph* of G if for each connected component K of G there is a connected component of H that is an Eulerian multi-subgraph of K. For a disconnected graph, define OPT(G, weight) to be the sum over connected components K of OPT(K, weight). Then OPT(G, weight) is the minimum weight of a multi-Eulerian multi-subgraph of G.

3 Spanner

Althöffer, Das, Dobkin, Joseph, and Soares [1] considered a simple and general procedure for producing a spanner in a (not necessarily planar) graph G_0: start with an empty graph G, consider the edges of G_0 in increasing order of weight, and add an edge to G if the edge's weight was much smaller than the minimum-weight path in G_0 between its endpoints. They did not address the exact running time of the procedure, but it clearly consists of $O(n)$ iterations, each involving a shortest-path computation. For planar graphs, therefore, it runs in $O(n^2)$ time [28]. They proved several results about the size and weight of the resulting spanner, including the following result that is specific to planar graphs.

Theorem 2 (Althöffer et al.) *For any planar graph G_0 with edge-weights and any $\epsilon > 0$, there is an edge subgraph G such that*

A1: *weight$(G) \leq (1 + 2\epsilon^{-1})MST(G_0)$, where $MST(G_0)$ is the weight of the minimum spanning tree of G_0, and*

A2: *for every pair of nodes u and v,*

$$\begin{aligned} & \textit{minimum weight of a } u\textit{-to-}v \textit{ path in } G \\ \leq \quad & (1+\epsilon) \cdot \textit{minimum weight of a } u\textit{-to-}v \textit{ path in } G_0 \end{aligned} \tag{1}$$

Lemma 9 *Properties A1 and A2 imply Properties S1 and S2 of Section 1.3 with $\rho_\epsilon = 1 + 2\epsilon^{-1}$.*

Proof: Because a tour includes a spanning tree, $MST(G) \leq \text{OPT}(G)$. Hence Property A1 implies that Property S1 of Section 1.3 is achieved with $\rho_\epsilon = 1 + 2\epsilon^{-1}$.

Now we show that Property A2 implies Property S2, i.e. that $\text{OPT}(G) \leq (1 + \epsilon_0)\text{OPT}(G_0)$. (This argument was used in [5].) Let T_0 be an optimal tour of G_0. For each edge uv of T_0 that is not in G, there is a u-to-v path in G of weight at most $(1 + \epsilon)$weight(uv); replace uv in T_0 with that path. The result of all the replacements is a tour T_1 whose weight is at most $1 + \epsilon$ times that of T_0. This shows $\text{OPT}(G) \leq (1 + \epsilon)\ \text{OPT}(G_0)$. □

By exploiting planarity, we can give an algorithm that runs in linear time but that can be shown (using the same analysis technique used by Althöffer et al.) to achieve the same properties.

Theorem 3 *There is a linear-time algorithm that, given a planar graph G_0 with edge-weights and any $\epsilon > 0$, outputs an edge subgraph G with Properties A1 and A2.*

The algorithm is as follows.

```
define SPANNER($G_0, \epsilon$):
    let $x[\cdot]$ be an array of numbers, indexed by edges
    find a minimum spanning tree $T$ of $G_0$
    assign $x[e] :=$ weight$(e)$ for each edge $e$ of $T$
    initialize $S := \{$edges of $T\}$
```

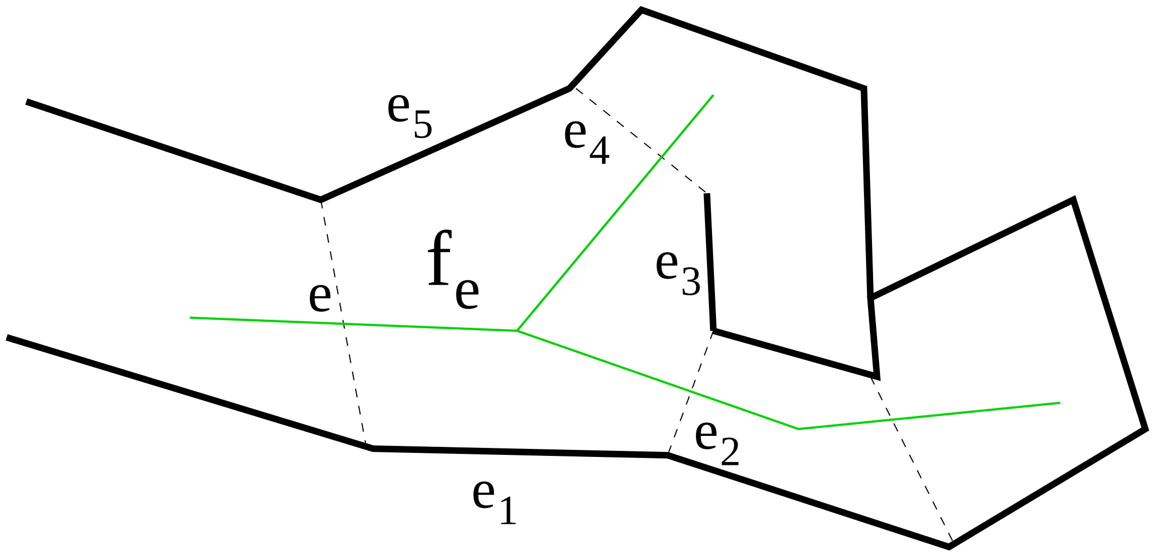


Figure 6: Diagram showing part of dual tree (in light edges) and primal tree (in dark edges) and primal nontree edges (dashed): e_2 and e_4 are child edges of e in the dual tree. The face f_e is indicated.

```
let $T^*$ be the dual tree, rooted at the infinite face
for each edge $e$ of $T^*$, in order from leaves to root
  let $f_e$ be the face of $G_0$ whose parent edge in $T^*$ is $e$
  let $e$=$e_0, e_1, \ldots, e_s$ be the sequence of edges comprising $f_e$
  $x_{\text{omit}} := \sum_{i=1}^{s} x[e_i]$
  if $x_{\text{omit}} > (1+\epsilon)\text{weight}(e)$
    then add $e$ to $S$ and assign $x[e] := \text{weight}(e)$
    else assign $x[e] := x_{\text{omit}}$
return $S$
```

The minimum spanning tree of G_0 can be found in linear time using the algorithm of Cheriton and Tarjan [13].

Now we address correctness of the procedure. Say an edge e is *accepted* when e is assigned to S, and *rejected* if e is considered but not assigned to S.

Lemma 10 *In the for-loop iteration in which e is considered, for every other edge e_i of f_e, $x[e_i]$ has been assigned a number.*

Proof: The face f_e has only one parent edge in T^*, and it is e. For every other edge e_i of f_e, either e_i belongs to T or e_i is a child edge of f_e in T^*. □

For any edge e of G_0 not in T,

- let $\hat{G}_e$ denote the subgraph of G_0 consisting of accepted edges together with e,
- let $\hat{f}_e$ denote the face of $\hat{G}_e$ that contains e and encloses f_e,
- let $\hat{W}_e$ denote the walk formed by the sequence of edges comprising $\hat{f}_e$ not including e itself, and
- let $P_e = \begin{cases} e & \text{if } e \text{ is accepted} \\ \hat{W}_e & \text{otherwise} \end{cases}$

Note that each of $\hat{W}_e$ and P_e has the same endpoints as e. For an edge e of T, define $P_e = e$. The basic argument of the following lemma comes from [1].

Lemma 11 *For any edge e of G_0, not in T,*

1. *every edge of $\hat{f}_e$ is either in T or is a descendent of e in T^*, and*
2. *$\hat{W}_e = P_{e_1} \cdots P_{e_s}$, where $e_1 \ldots e_s$ is the walk consisting of the edges comprising f_e other than e.*

Proof: by induction. Consider the case in which e is a leaf-edge of T^*. Let f be the corresponding leaf-node in G_0^*. Because f is a leaf, the only incident edge that is in T^* is e itself, so $e_1, \ldots, e_s$ belong to T. All these edges are accepted, proving Part 1. To prove Part 2, note that $W_e = e_1 \cdots e_s$ and that $P_{e_i} = e_i$ for $i = 1, \ldots, s$. Thus the lemma holds for e.

Consider the case where e is not a leaf. Let $\hat{G}_{e+}$ be the subgraph of G_0 consisting of accepted edges together with $e, e_1, \ldots, e_s$. For each e_i, recall that $\hat{f}_{e_i}$ is the face of $\hat{G}_{e_i}$ that contains e_i and encloses f_{e_i}. We claim that $\hat{f}_{e_i}$ is also a face of $\hat{G}_{e+}$. To prove the claim, note that $\hat{G}_{e_i}$ can be obtained from $\hat{G}_{e+}$ by deleting a subset of $\{e, e_1, \ldots, e_s\} - \{e_i\}$. None of these edges are edges of T or descendents in T^* of e_i, so, by Part 1 of the inductive hypothesis, none belongs to $\hat{f}_{e_i}$.

Note that $\hat{G}_e$ can be obtained from $\hat{G}_{e+}$ by deleting those edges among $e_1, \ldots, e_s$ that are rejected. By the claim, each such deletion replaces a rejected edge e_i in f_e with the walk $\hat{W}_{e_i}$. This together with the definition of P_{e_i} proves Part 2. By Part 1 of the inductive hypothesis, every edge in each $\hat{W}_{e_i}$ is an edge of T or a descendent of e_i in T and hence a descendent of e as well. This proves Part 1. $\square$

Lemma 12 *In the for-loop iteration that considers e,*

- *the value assigned to x_{omit} is weight$(\hat{W}_e)$, and*

- *the value assigned to $x[e]$ is weight(P_e).*

Proof: The proof is by induction. By Lemma 10, the edges $e_1, \ldots, e_s$ are considered before e. By the inductive hypothesis, $x[e_i] = \text{weight}(P_e)$. By Lemma 11, weight$(\hat{W}_e) = \sum_{i=1}^{s} x[e_i]$, which proves the first statement. The second statement follows by definition of P_e. □

Corollary 13 *For each edge e, weight$(P_e) \leq (1+\epsilon)$weight(e).*

Proof: If e is accepted, $P_e = e$ so the statement holds trivially. Suppose e is rejected. By the conditional in the algorithm, in the iteration considering e, the value assigned to x_{omit} was at most $(1+\epsilon)$weight(e). By the first part of Lemma 12, weight$(\hat{W}_e)$ and therefore weight(P_e) are at most $(1+\epsilon)$weight(e). □

Corollary 14 *The graph of accepted edges satisfies Property A2.*

Proof: For any pair of nodes u and v, let P be the shortest u-to-v path in G_0. For each edge e of P,there is a walk P_e consisting of accepted edges between the endpoints of e. By Corollary 13, weight$(P_e) \leq (1+\epsilon)$weight(e). Replacing each edge e of P with P_e therefore yields a walk of weight at most $\sum_{e \in P}(1+\epsilon)$weight$(e)$, which is at most $(1+\epsilon)$weight(P). □

Lemma 15 *At any time during the algorithm's execution, the weight of the infinite face in the graph consisting of accepted edges is at most*

$$2 \cdot MST(G_0) - \epsilon \cdot \textit{weight}(\textit{accepted edges not in } T)$$

Proof: The proof is by induction. Before the for-loop commences, the graph of accepted edges is T, the minimum spanning tree of G_0. Hence the weight of the infinite face is exactly $2 \cdot MST(G_0)$, so the lemma's statement holds for this time. Consider a for-loop iteration, and let e be the edge being considered. If e is not accepted, there is no change to the set of accepted edges, so the lemma's statement continues to hold.

Suppose e is accepted. Let G_{after} be the subgraph consisting of edges accepted so far, and let $G_{\text{before}} = G_{\text{after}} - \{e\}$. Note that G_{after} can be obtained from $\hat{G}_e$ by deleting edges that will be accepted in the future. By the leaves-to-root ordering, none of the deleted edges are descendents of e in T^*. By Part 1 of Lemma 11, therefore, $\hat{f}_e$ is a face of G_{after}. Let g be the other face of G_{after} that contains e.

We claim that g is the infinite face of G_{after}. To prove the claim, note that G_{after} can be obtained from G_0 by deleting edges that have already been rejected and edges not yet considered. By the leaves-to-root ordering, e's proper ancestors in T^* have not yet been considered, so they are among the edges deleted. These deletions are contractions in the dual. The root of T^* is the infinite face, so the contractions result in g being the infinite face.

Note that G_{before} can be obtained from G_{after} by deleting e. This deletion replaces e in the face g with $\hat{W}_e$. This shows that

$$\begin{aligned}
&\text{weight of infinite face in } G_{\text{before}} - \text{weight of infinite face in } G_{\text{after}} \\
&= \text{weight}(\hat{W}_e) - \text{weight}(e) \\
&> (1+\epsilon)\text{weight}(e) - \text{weight}(e) \text{ because } e \text{ was accepted} \\
&= \epsilon \cdot \text{weight}(e)
\end{aligned}$$

which shows that the lemma's statement continues to hold. □

Corollary 16 *The graph G of accepted edges satisfies Property A1.*

Proof: By Lemma 15, the weight of the infinite face in the graph consisting of all accepted edges is at most

$$2 \cdot MST(G_0) - \epsilon \cdot \text{weight}(\text{accepted edges not in } T)$$

so weight(accepted edges not in T) $\leq 2\epsilon^{-1} \cdot MST(G_0)$. Since weight$(T) = MST(G_0)$, it follows that the weight of all accepted edges is at most $(1 + 2\epsilon^{-1})MST(G_0)$. □

This completes the proof of Theorem 3.

4 Slices

Let G be a connected planar embedded graph and let weight$(\cdot)$ be an edge-weight assignment. Let k be a parameter. Recall that G^* denotes the planar dual of G. Let f_∞ be the infinite face of G, which is a vertex of G^*. Define the *level* of a node v of G^* to be its breadth-first-search distance in G^* from f_∞, i.e. the minimum number of edges in an f_∞-to-v path in G^*. Define the level of an edge e to be ℓ if one endpoint has level ℓ and the other endpoint has level $\ell + 1$.

For $j = 0, 1, \ldots, k-1$, let S_j denote the set of edges e whose levels are congruent to j mod k. Let $t = \text{minarg}_j \text{weight}(S_j)$, and let $S = S_t$. We obtain the following bound.

$$\text{weight}(S) \leq (1/k)\,\text{weight}(G) \tag{2}$$

For $i = 0, 1, 2, \ldots$, let E_i be the set of edges e having at least one endpoint with level in the range $(t + (i-1)k, t + ik]$. We define *slice* i of G to be the subgraph of G (the primal graph) consisting of the edges E_i. Note that an edge of G belongs to two distinct slices only if the edge belongs to S.

The theorem below shows that the total weight of optimal tours in the slices exceeds the weight of the optimal tour of G by at most twice the weight of S.

Theorem 4 $\sum_i \mathit{OPT}(\mathit{slice}\ i) \leq 2\,\mathit{weight}(S) + \mathit{OPT}(G)$.

The next theorem states that the planar dual of each slice has a low-depth spanning tree.

Theorem 5 *For $i = 0, 1, 2, \ldots,$ each connected component of the planar dual of slice i has a rooted spanning tree of depth at most $k + 1$.*

In the rest of this section, we prove Theorems 4 and 5.

The following lemma is illustrated in Figure 7.

Lemma 17 *For $i = 1, 2, 3, \ldots,$ the edges of level $t + (i-1)k$ form a set A_i of simple cycles in G with the following properties:*

1. *The cycles are edge-disjoint.*

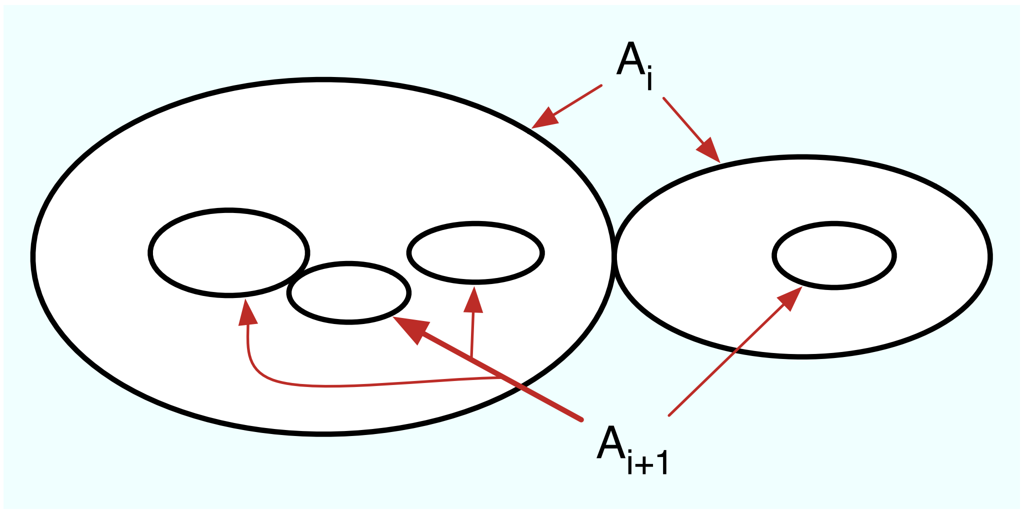
2. *Every face is enclosed by at most one of the cycles.*

3. *A face u of G is enclosed by one of the cycles iff in the dual graph G^* the node u has level greater than $t + (i-1)k$.*

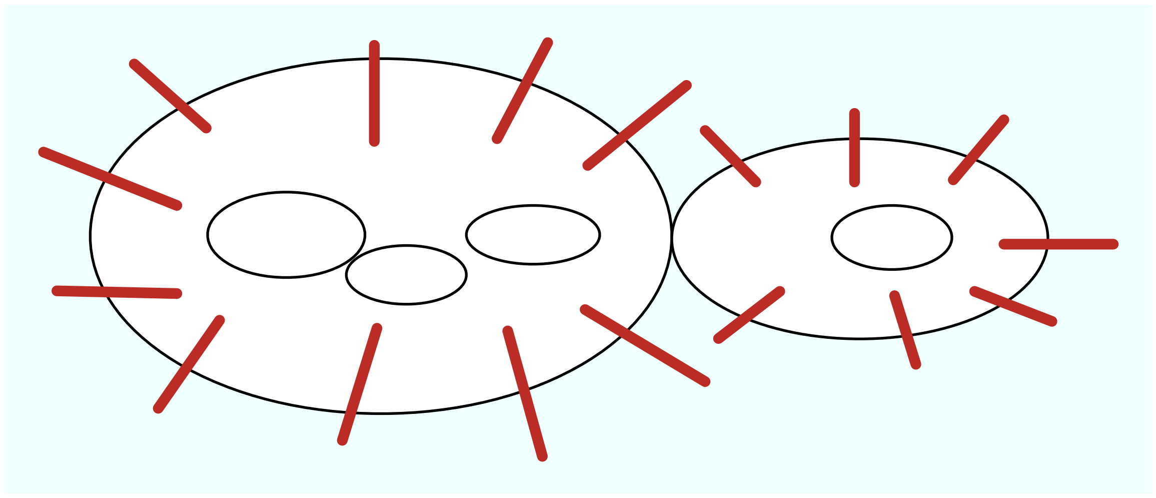
Proof: Let T be a breadth-first-search tree of G^* rooted at f_∞. For $i = 1, 2, \ldots,$ let $\mathcal{K}_i$ be the set of connected components of the subgraph of G^* consisting of nodes whose levels exceed $t + (i-1)k$. Let $A_i = \{\Gamma_{G^*}(V(K)) \ : \ K \in \mathcal{K}_i\}$.

Let K be a connected component in $\mathcal{K}_i$. For any node v not in K, if v is not f_∞ then v has a parent p whose level is one less than that of v. If p's level is at most $t + (i-1)k$ then p is not in K; if p's level is greater than $t + (i-1)k$ then so is v's, so if p were in K then v would also be in K, a contradiction. Thus p is not in K. By induction, G^* contains a v-to-f_∞ path that avoids K, proving that the nodes of G^* not in K are connected, so $\Gamma_{G^*}(V(K))$ is a bond. By Proposition 3, the edges of $\Gamma_{G^*}(V(K))$ form a simple cycle C_K in G. The faces enclosed by C_K are the nodes of K.

Consider two components $K_1, K_2 \in \mathcal{K}_i$. Since $V(K_1)$ and $V(K_2)$ are disjoint, the faces enclosed by C_{K_1} are disjoint from the faces enclosed by C_{K_2}. Furthermore, for $j = 1, 2$, an edge belongs to C_{K_j} if in G^* the edge connects a node of K_j to a node at level $t + (i-1)k$, which shows that C_{K_1} and C_{K_2} are edge-disjoint. $\square$



(a)



(b)

Figure 7: Cycles in A_i and A_{i+1} are shown. Note that the cycles of A_{i+1} are enclosed within cycles of A_i. In the figure on the bottom, the dual edges corresponding to cycles in A_i are indicated by thick lines.

Lemma 18 *Let $A_1, \ldots$ be the set of cycles from Lemma 17. For $i \geq 1$, an edge e of G belongs to slice i iff e is enclosed by some cycle in A_i and not strictly enclosed by any cycle in A_{i+1}. An edge e belongs to slice 0 iff e is not strictly enclosed by any cycle in A_1.*

Proof: By definitions of slice and dual, for $i \geq 1$, an edge e belongs to slice i iff e belongs to a face f whose level in G^* is in $(t + (i-1)k, t + ik]$. By Lemma 17, the level of f is in $(t + (i-1)k, t + ik]$ iff f is enclosed by a cycle in A_i and not by a cycle in A_{i+1}. The lemma follows by the definition of a cycle enclosing an edge. The case of slice 0 is similar. □

We say a subgraph is *even* if every node has even degree.

Lemma 19 *Let R be an Eulerian multi-subgraph of G, let C be a simple cycle of G, and let X be the set of nodes enclosed by C. There is a subset $\widehat{C}$ of the edges of C such that*

1. *weight$(\widehat{C}) \leq \frac{1}{2}$weight$(C)$, and*
2. *each connected component of $R - \Gamma_G(X) \cup \widehat{C}$ is even.*

For a graph G, a node v, and a set A of edges, we define $\deg_G(v, A)$ to be the number of edges in A that in G are incident to v,

Proof: Because R is Eulerian, $\deg(v, R)$ is even for every node v, so $\sum\{\deg(v, R) \ : v \in V(C)\}$ is even. The closed walk (Eulerian cycle) corresponding to R enters X the same number of times as it leaves, so $|R \cap \Gamma_G(X)|$ is even. It follows that $\sum\{\deg(v, R - \Gamma_G(X)) \ : \ v \in V(C)\}$ is even. Hence the set $Y = \{v \in V(C) \ : \ \deg(v, R - \Gamma_G(X)) \text{ is odd}\}$ has even cardinality.

Write $C = P_1 \ \ldots \ P_{|Y|}$ where each P_i is a path whose endpoints belong to Y and whose internal nodes do not. Let $\widehat{C}$ denote the set of edges in $P_1, P_3, P_5, \ldots, P_{|Y|-1}$ or the set of edges in $P_2, P_4, P_6, \ldots, P_{|Y|}$, whichever has less weight. This choice ensures Property 1 in the lemma's statement. Also, for each vertex $v \in V(C)$, $\deg(v, \widehat{C})$ is odd iff $v \in Y$, which proves Property 2.

□

Lemma 20 *For some $i \geq 1$, let W denote the set of nodes on cycles $C \in A_i$. Two nodes of W are connected via a path in slice i iff they are connected via a path consists only of edges belonging to cycles C in A_i*

Proof: For two nodes $x, y \in W$, let P be the x-to-y path in slice i that uses the fewest edges not belonging to cycles $C \in A_i$. Assume for a contradiction that P contains some edge e that does not belong to a cycle $C \in A_i$. Let $\widehat{P}$ be the maximal subpath of P that contains e but whose internal nodes do not belong to W.

By Lemma 18, e is strictly enclosed by some cycle $C \in A_i$. Since no internal node of $\widehat{P}$ belongs to W, every edge of $\widehat{P}$ must be enclosed by the same cycle C. But then the endpoints of $\widehat{P}$ must belong to that same cycle C. Consequently, $\widehat{P}$ can be replaced by a subpath of C, contradicting the choice of P. $\square$

Now we can prove Theorems 4 and 5.

Proof of Theorem 4: Let M be the multiset of edges comprising the optimal tour of G. Then M is an Eulerian bisubgraph of G. Let M_i denote the submultiset of M consisting of edges in slice i. To prove Theorem 4, we will show that, for each slice i, there is a multiset D_i of edges of slice i such that $M_i \cup D_i$ is an Eulerian multi-subgraph of slice i, i.e.

1. every node of slice i has even degree with respect to $M_i \cup D_i$, and
2. for every connected component K of slice i, there is a corresponding connected component of $M_i \cup D_i$ that visits all nodes of K.

We ensure that $\sum_i \text{weight}(D_i) \leq 2\,\text{weight}(S)$.

We build D_i in three steps. The first two steps address achieving Property 1. By Lemma 18, slice i consists of the edges enclosed by cycles of A_i and not strictly enclosed by cycles of A_{i+1}. If v belongs to no cycle in either A_i or A_{i+1}, then every edge of M incident to v belongs to M_i, so $\deg(v, M_i)$ is already even.

In step one, we address the case of nodes v belonging to cycles in A_i. For each cycle $C \in A_i$, we apply Lemma 19 to M and C, obtaining an edge-subset $\widehat{C} \subseteq E(C)$, and we include $\widehat{C}$ in D_i. This change affects the degree of a node v only if v belongs to some cycle $C \in A_i$. Such a node v has even degree with respect to those edges of $M \cup \widehat{C}$ that are enclosed in C. Summing over all cycles $C \in A_i$, the node has even degree with respect to those edges of $M \cup \bigcup\{\widehat{C} \ : \ C \in A_i\}$ that belong to slice i.

In step two, we address the case of nodes v belonging to cycles in A_{i+1}. Because the infinite face of a planar embedded graph can be chosen arbitrarily (and by definition of *enclosed*), we can apply Lemma 19 to each cycle $C \in A_{i+1}$ and to the set X of nodes *not* strictly enclosed by C, obtaining a set $\widehat{C}$ of edges such that $\deg(v, R - \Gamma_G(X) \cup \widehat{C})$ is even for each node v of C. We include each set $\widehat{C}$ in D_i. As before, this change affects the degree of a node v only if v belongs to some cycle $C \in A_{i+1}$, and ensures that such a node v has even degree with respect to those edges of $M \cup \bigcup\{\widehat{C} \ : \ C \in A_{i+1}\}$ that belong to slice i.

In step three, we address Property 2. For each $C \in A_i$, we add $E(C)$ to D_i. This does not change the parity of any node's degree. Let v_1 and v_2 be two nodes in slice i. For $j = 1, 2$, it follow from Lemma 18 that v_j is enclosed by some cycle $C_j \in A_i$. Let w be a node on the boundary of the infinite face. Let P_j be a v_j-to-w path using edges of M, and let $\widehat{P_j}$ be a maximal prefix of P_j consisting of edges enclosed by C_j. Since w is not strictly enclosed by C_j, $\widehat{P_j}$ must end at a node w_j of C_j.

Suppose v_1 and v_2 belong to the same connected component K of slice i. Then w_1 and w_2 also belong to K. By Lemma 20, there is a w_1-to-w_2 path using only edges of $\{E(C) \,:\, C \in A_i\}$, and hence using only edges of D_i. Combining this path with $\widehat{P_1}$ and $\widehat{P_2}$, we obtain a path using only edges of $M \cup D_i$ that belong to slice i. This proves Property 2.

Finally, we bound $\sum_i \text{weight}(D_i)$. The cycles $C \in A_i$ consist of edges having level $t+(i-1)k$, so

$$\sum_i \sum \{\text{weight}(C) \,:\, C \in A_i\} = \text{weight}(S)$$

The weight added to $\bigcup D_i$ in each of steps one and two is at most $\sum_i \frac{1}{2} \sum \{\text{weight}(C) \,:\, C \in A_i\}$. The weight added in step three is $\sum_i \sum \{\text{weight}(C) \,:\, C \in A_i\}$. The total is at most $2\,\text{weight}(S)$. This completes the proof of Theorem 4. $\square$

Proof of Theorem 5: Let K be a connected component of slice i where $i \geq 1$. By Lemmas 18, slice i can be obtained from G by (i) deleting edges properly enclosed by cycles of A_{i+1} and (ii) deleting edges not enclosed by cycles of A_i. For each cycle $C \in A_{i+1}$, deleting the edges properly enclosed by C merges the faces enclosed by C into a single face. Let D be the set of edges not enclosed by cycles of A_i. Deleting the edges in D corresponds in the planar dual G^* to compressing edges both of whose endpoints have levels at most $t + (i-1)k$. Let T be the breadth-first-search tree of G^*, and consider the effect of these operations on T. Recall that compressing a non-self-loop edge is equivalent to contracting. First, in the planar dual, compress all the edges of T that are in D. Because these edges form a subtree of T, none is a self-loop, so these compressions are contractions. For each (dual) node v having level at most $t + (i-1)k$, there is a path in T consisting of edges of D from v to the root, so the contractions merge all these nodes into a single node $\hat{r}$. Let $\widehat{T}$ be the set of edges of T that remain.

Each face of slice i that was a face of G had distance at most $t + ik$ from the root in T, and hence has distance at most $(t + ik) - (t + (i-1)k)$ from the root $\hat{r}$ in $\widehat{T}$. Each face arising from deleting edges properly enclosed by cycles of A_{i+1} is adjacent in the dual to some node that had been at level $t + ik$, and hence has distance at most $k + 1$ from $\hat{r}$.

Each of the remaining edges of D is now a self-loop with common endpoint $\hat{r}$. Compressing these edges in the dual might in general split $\hat{r}$ into multiple nodes corresponding to multiple connected components in the primal graph. However, each connected component retains its own low-depth spanning tree. This completes the proof of Theorem 5. $\square$

5 TSP algorithm

Now we describe the TSP algorithm.

Let G_0 be the input planar embedded graph, and let weight$(\cdot)$ be the input edge-weight assignment.

Step 1 (Spanner Step): Let ϵ_0 be the desired accuracy. Define $\epsilon = \epsilon_0/2$. Obtain a subgraph G of G_0 that has Properties S1 and S2 of Section 1.3.

Step 2 (Slicing Step): Use breadth-first search in the planar dual to find the slices as described in Section 4, with $k = 2\epsilon^{-1}\rho_\epsilon$, where ρ_ϵ is the multiplier in Property S1. For each slice, for each connected component of that slice, the planar dual has a spanning tree of depth at most $k + 1$.

Step 3 (Dynamic programming): For each slice, for each connected component of that slice, find a minimum-weight Eulerian multi-subgraph of that component

Step 4: Combine the multi-subgraphs to obtain an Eulerian multi-subgraph of G, then turn it into a tour of G.

5.1 Running time

Let n be the number of nodes in the input graph G_0. Assume G_0 has no parallel edges, so it has $O(n)$ edges. For unit-weight graphs, Step 1 is trivial: $G = G_0$ and ρ_ϵ is a constant. For arbitrary weights, Theorem 3 gives an $O(n)$ algorithm achieving $\rho_\epsilon = 1 + 2\epsilon^{-1}$. Steps 2 and 4 take $O(n)$ time.

As for Step 3, Cook and Seymour observe [15] that TSP can be solved in a graph of bounded branchwidth. In Section 7, we state a theorem, due to Tamaki [45], that shows that each slice has branch-width at most $2k + 3$.

Because Cook and Seymour do not formally describe or analyze their dynamic program, in Section 6 we describe a dynamic program that can be used in Step 3. This dynamic program exploits planarity to get a running time of $O(c^k n')$ for a graph of size n' (where c is a constant). Summing over all connected components of all slices, the running time for Step 3 is $O(c^k n)$. The choice of k yields a running time of $O(d^{1/\epsilon} n)$ for unit-weight graphs and $O(d^{1/\epsilon^2} n)$ for arbitrary weights, where d is a constant.

5.2 Correctness

Theorem 6 *The algorithm finds a tour of weight at most* $(1+\epsilon_0)OPT(G_0)$.

Proof: The tours found in Step 3 are connected and jointly visit all nodes, so their union is connected and spans all nodes. Every node v has even degree with respect to every tour that contains it, so v has even degree with respect to the multiset union of these tours. Thus the multiset union is Eulerian. The Eulerian characterization (Proposition 8) implies that the union can be transformed into a tour.

The weight of the tour is $\sum_i \text{OPT}(\text{slice } i)$, which by Theorem 4 is at most $\text{OPT}(G)+2\,weight(S)$. To complete the proof of Theorem 6, we bound these two terms. Property S2 states that $\text{OPT}(G) \leq (1+\epsilon)\text{OPT}(G_0)$. Observe that

$$\begin{aligned}
2\,\text{weight}(S) &\leq (2/k)\,\text{weight}(G) && \text{by (2)} \\
&\leq \epsilon\rho_\epsilon^{-1}\text{weight}(G) && \text{by choice of } k \\
&\leq \epsilon \cdot \text{OPT}(G_0) && \text{by Property S1}
\end{aligned}$$

Since $\epsilon_0 = 2\epsilon$, this completes the proof of Theorem 6. □

6 Solving TSP in a planar embedded graph with bounded dual radius

In this section we describe an algorithm that, given an edge-weighted planar embedded graph H, a low-depth spanning tree of H^*, and a set R of nodes, finds an minimum-weight walk W such that $R \subseteq V(W)$. To find an optimal tour, R is set to $V(H)$. Rather than describe the algorithm for this special case, we describe the algorithm for the more general case because doing so requires very little change.

Theorem 7 *There is an algorithm that, given a planar embedded graph* H *without parallel edges, an edge-weight assignment for* H*, a subset* R *of nodes of* H*, and a spanning tree* T^* *of* H^* *in which every simple path has length at most* ℓ*, finds a minimum-weight connected even multi-subgraph of* H *that visits all nodes in* R*. The algorithm takes time* $O(c^\ell|V(H)|)$ *for some constant* c*.*

First we show how to reduce the problem to the case in which the degree of the input graph is bounded by three. Then we show how to solve this case using dynamic programming.

6.1 Reduction to degree three

Lemma 21 *Let H be a graph, let W be an even connected multi-subgraph of H, and let e be an edge of H. Then $W - \{e\}$ is an even connected multi-subgraph of $H/\{e\}$.*

Proof: Since the edges of W are connected in H, the edges of $W - \{e\}$ are connected in $H/\{e\}$. For every node v that is not an endpoint of e in H, the degree of v in $H/\{e\}$ with respect to $W - \{e\}$ equals the degree of v in H with respect to W. Let u_1 and u_2 be the endpoints of e in H. These nodes are coalesced in $H/\{e\}$ to form a single node whose degree with respect to $W - \{e\}$ is

$$\sum_{i=1}^{2}(\deg_H(u_i, W) - |W \cap \{e\}|) = \deg_H(u_1, W) + \deg_H(u_2, W) - 2|W \cap \{e\}|$$

Since each of the terms on the right-hand side is even, the sum is even. □

Lemma 22 *Let H be a graph, let e be an edge of H, and let W be an even connected multi-subgraph of $H/\{e\}$. Then one of W, $W \cup \{e\}$, $W \cup \{e\} \cup \{e\}$ is an even connected multi-subgraph of H.*

Proof: Trivial if e is a self-loop. Otherwise, let u_1 and u_2 be the endpoints of e in H. These nodes are coalesced in $H/\{e\}$ to form a node v. Since $\deg_{H/\{e\}}(v, W)$ is even, $\sum_{i=1}^{2} \deg_H(u_i, W)$ is even.

Case 1: $\deg_H(u_1, W)$ is odd. Then $\deg_H(u_2, W)$ is also odd, and there are edges in W incident to u_1. Hence $\deg_H(u_i, W \cup \{e\})$ is even for $i = 1$ and 2, and $W \cup \{e\}$ is connected.

Case 2: $\deg_H(u_1, W)$ is even but at least one edge of W is incident to u_1 or u_2. Then $\deg_H(u_2, W)$ is also even, so $\deg_H(u_i, W \cup \{e\} \cup \{e\})$ is even for $i = 1$ and 2, and $W \cup \{e\} \cup \{e\}$ is connected.

Case 3: No edge of W is incident to u_1 or u_2. Then in $H/\{e\}$ no path in W passes through v, so the fact that W is connected in $H/\{e\}$ implies that W is connected in H. Clearly $\deg_H(u_i, W)$ is even for $i = 1$ and 2. □

Now we give the reduction to the degree-three case.

Step 1: Triangulate the faces of H^* by adding zero-weight artificial edges until every face has size at most three. Let A be the set of artificial edges added. Let $\hat{H}^*$ be the resulting planar embedded graph.

Step 2: H can be obtained from $\hat{H}$ by contracting the artificial edges, which merges some nodes. Let $\hat{R} = \bigcup_{v \in R}\{\text{nodes of } \hat{H} \text{ merged to form } v\}$.

Step 3: Let W be a minimum-weight connected even multi-subgraph of $\hat{H}$ that visits all nodes of $\hat{R}$.

Step 4: Return $W - A$.

Lemma 23 *$W - A$ is a minimum-weight even multi-subgraph visiting all nodes of R.*

Proof: By repeated application of Lemma 22, using the fact that the artificial edges have zero weight. we infer OPT$(\hat{H}) \leq$ OPT(H). Therefore weight$(W) \leq$ OPT(H). By repeated application of Lemma 21, we infer that $W - A$ is an even connected multi-subgraph of H that visits all nodes of R. □

6.2 Overview of dynamic program

Now we describe how to find an optimal tour of $\hat{H}$ visiting all nodes of $\hat{R}$. The graph H^* has a rooted spanning tree T^* in which every simple path has at most ℓ edges, and $\hat{H}^*$ is obtained from H^* by adding edges, so T^* is also a spanning tree of $\hat{H}$. Because every face of $\hat{H}^*$ is a triangle, $\hat{H}$ has degree at most three. Let $\hat{T}$ be the set of edges of $\hat{H}$ not in T^*. Then $\hat{T}$ is a spanning tree of $\hat{H}$ and hence has degree at most three. Root $\hat{T}$ at a node r of degree 1 in $\hat{T}$. The dynamic program will work up $\hat{T}$ from the leaves to the root. For each edge of $\hat{T}$, the dynamic program will construct a table. The value of OPT$(\hat{H})$ will be be computed from the table associated with the edge connecting the root to its child. Once the value of OPT$(\hat{H})$ is known, the tour itself can be constructed in a post-processing phase by working down from the root to the leaves. (The post-processing is straightforward, and we do not describe it here.)

6.3 Terminology

Before giving a detailed description of the tables, we need to introduce some terminology.

Traversals Let $\Gamma_{\hat{H}}(S)$ be a cut. We say a nonempty, dart-disjoint set $\mathcal{P}$ of walks in $\hat{H}$ is a *traversal of S in $\hat{H}$* if

- the start node and end node of each path are not in S,
- the internal nodes of each path are in S.

It follows that the first and last darts of each path belong to $\Gamma_{\hat{H}}(S)$., i.e. that each is a dart of some edge in $\Gamma_{\hat{H}}(S)$.

Define

$$\begin{aligned}\text{weight}(\mathcal{P}) &= \sum\{\text{weight}(d)\ : d \in \mathcal{P}, d \text{ not a dart of } \Gamma_{\hat{H}}(S)\} \\ &\quad + \frac{1}{2}\sum\{\text{weight}(d)\ : d \in \mathcal{P}, d \text{ a dart of } \Gamma_{\hat{H}}(S)\}\end{aligned}$$

Configurations A *configuration* K of a cut $\Gamma_{\hat{H}}(S)$ is a nonempty set of ordered pairs $\langle, d_i, d_j\rangle$ of darts of $\Gamma_{\hat{H}}(S)$ such that the head of d_i and the tail of d_j belong to S, and such that each dart of $\Gamma_{\hat{H}}(S)$ occurs at most once in each position. The number of configurations is at most $(2\eta)!$, where $\eta = |\Gamma_{\hat{H}}(S)|$.

If S is connected in $\hat{H}$ then the embedding determines a cyclic ordering of the edges of $\Gamma_{\hat{H}}(S)$, say $(e_1\ \cdots\ e_\eta)$. In this case, we say that a configuration is *crossing* if it includes a dart pair corresponding to the pair $\langle e_p, e_q\rangle$ of edges and also a dart pair corresponding to the pair $\langle e_r, e_s\rangle$ of edges, where $p < r < q < s$. A Catalan bound shows that the number of noncrossing configurations is $2^{O(\eta)}$. This is where planarity is used in the dynamic program.[10]

For a configuration K, define weight(K) to be the sum of the weights of the darts in K.[11]

Let $C_1, \ldots, C_d$ be cuts in a graph, and let $K_1, \ldots, K_d$ be corresponding configurations. A *subtour* is a sequence $d_0, \ldots, d_{b-1}$ such that, for $j = 0, \ldots, b-1$, the pair $\langle d_j, d_{(j+1) \bmod b}\rangle$ belongs to some configuration. We say $K_1, \ldots, K_d$ are *consistent* if for each pair C_i, C_j of cuts, each dart represented in both C_i and C_j occurs in both K_i and K_j or occurs in neither, and if there is no subtour.

Define

$$\kappa(\mathcal{P}) = \{\langle \text{first dart of } P,\ \text{last dart of } P\rangle\ :\ P \in \mathcal{P}\}$$

6.4 Definition of the tables

In this subsection we describe the tables produced by the dynamic program. For each edge e of $\hat{T}$, let v_e denote the child endpoint of e, and let D_e denote the descendents of v_e. By Lemma 4, the edges comprising $\Gamma_{\hat{H}}(D_e)$ are exactly the edges comprising the elementary cycle of e in $\hat{H}^*$ with respect to T^*. We denote this cycle by C_e. (See Figure 8.) That elementary cycle consists of e together with a simple path in T^* between the endpoints of e. The cycle therefore contains at most $\ell + 1$ edges. This shows $|\Gamma_{\hat{H}}(D_e)| \leq \ell + 1$.

[10]For a discussion of Catalan numbers, see any text on combinatorics, e.g. [43]. Noncrossing configurations and a Catalan bound were used in a dynamic program for TSP by Arora et al. [5]. Concurrent with the appearance of a preliminary version of this paper, Dorn et al. [20] published an extended abstract discussing planar-graph algorithms that also exploited Catalan-type bounds and noncrossing matchings.

[11]The weight of a dart is the weight of the corresponding edge.

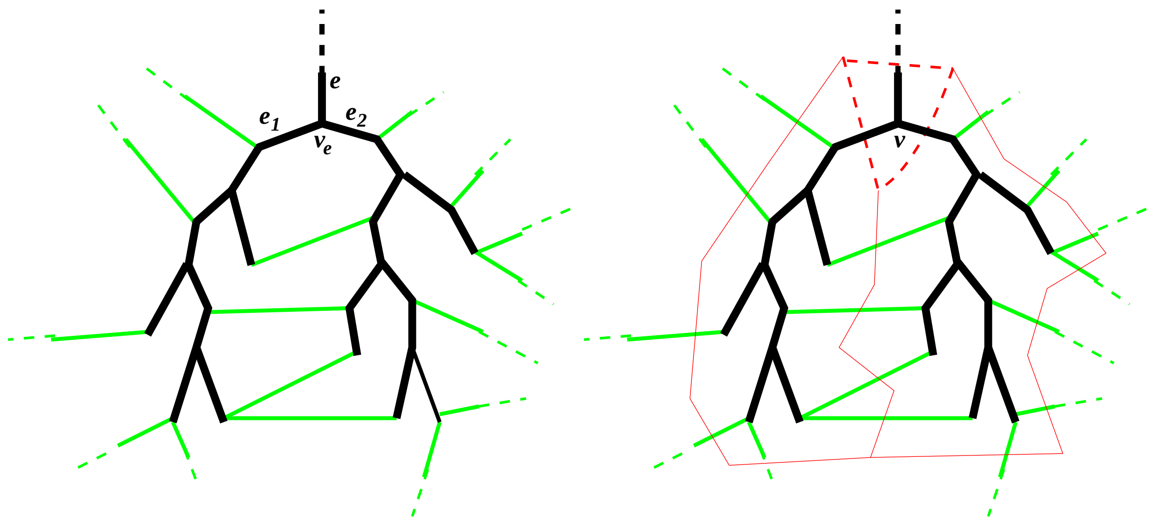


Figure 8: A subgraph arising in the dynamic program. The edge e, the child node v_e, and the child edges e_1 and e_2 are labeled. The dark edges are tree edges. On the right is the same subgraph with some edges of the dual graph also shown. Note that the edges of $\Gamma(\{\text{descendents of } v_e\})$ form an elementary cycle in the dual, as do the edges of $\Gamma(\{\text{descendents of } e_1\})$ and the edges of $\Gamma(\{\text{descendents of } e_2\})$.

For a cut $\Gamma_{\hat{H}}(S)$ of $\hat{H}$ where S is connected in $\hat{H}$, for a configuration K of $\Gamma_{\hat{H}}(S)$, define

$$\begin{aligned} M_S(K) = \min\{\text{weight}(\mathcal{P}) \ : \ & \kappa(\mathcal{P}) = K, \\ & \mathcal{P} \text{ is a traversal of } S, \text{ and} \\ & S \cap \hat{R} \subseteq V(\mathcal{P})\} \end{aligned}$$

We show in Corollary 27 that, for each edge e of T, the dynamic program will construct a table TAB_e, indexed by the noncrossing configurations K of $\Gamma_{\hat{H}}(D_e)$, such that $\text{TAB}_e[K] = M_{D_e}(K)$.

For the root edge $\hat{e}$ of T (the edge of T incident to r), each edge of $\Gamma_{\hat{H}}(D_{\hat{e}})$ is incident to the root r. It follows that every traversal of $D_{\hat{e}}$ defines a tour of $\hat{H}$ using each dart at most once, and vice versa. By Proposition 7, there is an optimal tour that is noncrossing. Hence

$$\begin{aligned} &\text{OPT}(\hat{H}) \\ &= \ \min\{M_{\hat{e}}(K) + \frac{1}{2}\text{weight}(K) \ : \ K \text{ a configuration of } C_{\hat{e}}\} \end{aligned}$$

because only half the weight of each edge of K is counted in $M_{\hat{e}}(K)$. Since r has degree at most three, $C_{\hat{e}}$ has $O(1)$ configurations. Thus $\text{OPT}(\hat{H})$ can be computed in $O(1)$ time from the table $\text{TAB}_{\hat{e}}$.

6.5 The recurrence relation

Let e be an edge of the tree $\hat{T}$, and let $e_1, \ldots, e_s$ be its child edges ($s \leq 2$). Let $D_0 = \{v_e\}$. For $i = 1, \ldots, s$, let $D_i = D_{e_i}$. Note that D_e is the disjoint union $\bigcup_{i=0}^{s} D_i$. For $i = 0, 1, \ldots, s$, let C_i denote $\Gamma_{\hat{H}}(D_i)$.

A traversal $\mathcal{P}$ of D_e *induces* a traversal $\mathcal{P}_i$ of D_i for $i = 0, 1 \ldots, s$ as follows: for each path $P \in \mathcal{P}$, break P into subpaths at the nodes of P that are not in D_i, and retain only those darts d such that at least one endpoint of d is in D_i. The remaining subpaths form a traversal of D_i. The following lemma is immediate.

Lemma 24 *Let $\mathcal{P}$ be a traversal of D_e, and let $\mathcal{P}_0, \ldots, \mathcal{P}_s$ be the traversals that $\mathcal{P}$ induces for $D_0, \ldots, D_s$. Then*

$$\textit{weight}(\mathcal{P}) = \sum_{i=0}^{s} \textit{weight}(\mathcal{P}_i)$$

and $\kappa(\mathcal{P}), \kappa(\mathcal{P}_0), \kappa(\mathcal{P}_1), \ldots, \kappa(\mathcal{P}_s)$ are consistent.

Lemma 25 *For traversals $\mathcal{P}_0, \ldots, \mathcal{P}_s$ of $D_0, \ldots, D_s$, if K is a configuration of C_e such that $K, \kappa(\mathcal{P}_0), \ldots, \kappa(\mathcal{P}_s)$ are consistent then there is a traversal $\mathcal{P}$ of C_e that induces $\mathcal{P}_0, \ldots, \mathcal{P}_s$ such that $\kappa(\mathcal{P}) = K$.*

Proof: By gluing together paths from different $\mathcal{P}_i$'s that have a common dart, one constructs paths whose start and end darts are in $\Gamma(D_e)$. The consistency condition ensures that the glueing can be completed, and that the start and end darts are represented in K. □

Corollary 26 *For any configuration K of C_e,*

$$M_{D_e}(K) = \min\{\sum_{i=0}^{s} M_{D_i}(K_i) \ : \ K, K_0, \ldots, K_s \textit{ are consistent}\}$$

Proof: To show that the left-hand side is at most the right-hand side, fix consistent configurations $K, K_0, \ldots, K_s$. For $i = 0, \ldots,$ let $\mathcal{P}_i$ be the traversal achieving the minimum in the definition of $M_{D_i}(K_i)$. (If there is no such traversal, the right-hand side is infinity.) By Lemma 25, there is a traversal $\mathcal{P}$ of D_e that induces $P_0, \ldots, P_s$. It follows that $D_e \cap \hat{R} \subseteq V(\mathcal{P})$.

By the first part of Lemma 24, weight$(\mathcal{P}) = \sum_{i=0}^{s}$ weight$(\mathcal{P}_i)$, so $M_{D_e}(K) \leq \sum_{i=0}^{s} M_{D_i}(K_i)$.

To show that the right-hand side is at most the left-hand side, let K be a configuration such that $M_{D_e}(K)$ is finite, and let $\mathcal{P}$ be the traversal achieving the minimum in the definition of $M_{D_e}(K)$. Let $\mathcal{P}_0, \ldots, \mathcal{P}_s$ be the traversals that $\mathcal{P}$ induces for $D_0, \ldots, D_s$. It follows from Lemma 24 that the right-hand side is at most weight$(\mathcal{P})$. □

6.6 The dynamic program

We now give a recursive algorithm TSP-DP(e) that for each edge e of T populates the table TAB$_e$.

```
define TSP-DP$(e)$:
1 let $e_1, \ldots, e_s$ be the child edges of $e$ ($s \leq 2$)
2 for $i = 1, \ldots, s$,
3     recursively call TSP-DP$(e_i)$.
4 initialize each entry of $\text{TAB}_e$ to $\infty$.
5 for each consistent tuple $(K, K_0, K_1, \ldots, K_s)$
          of configurations of $\Gamma(D_e), \Gamma(\{v_e\}), \Gamma(D_{e_1}), \ldots, \Gamma(D_{e_s})$
6     $\text{TAB}_e[K] := \min\{\text{TAB}_e[K]$ ,
                $M_{\{v_e\}}(K_0) + \sum_{i=1}^{k} \text{TAB}_{e_i}[K_i]\}$
```

Note that in Step 6, $M_{\{v_e\}}(K_0)$ can be computed directly in $O(1)$ time. The correctness of the algorithm follows from Corollary 26 by induction.

Corollary 27 (Correctness of TSP-DP**)** *For each edge e of T, for each noncrossing configuration K of C_e, $\text{TAB}_e[K] = M_{D_e}(K)$.*

6.7 Analysis of the dynamic program

In Step 5, each of the cuts $\Gamma(D_{e_1}), \ldots, \Gamma(D_{e_s})$ has size at most $\ell + 1$, so has $O(c^\ell)$ configurations for a constant c. The cut $\Gamma(\{v_e\})$ has size at most three, and $s \leq 2$, so the number of tuples in Step 5 is $O(d^\ell)$ for a constant d. Thus each invocation of TSP-DP requires $O(d^\ell)$ time. The number of invocations is $|V(\hat{H})| - 1$, so the entire dynamic program takes time $O(d^\ell |V(\hat{H})|)$. Combined with the reduction of Subsection 6.1, this completes the proof of Theorem 7.

7 Achieving low branch-width by contracting edges

Branch-width is a graph measure akin to (and within a constant factor of) tree-width. (We will review the definition presently.) Many computational graph problems that are NP-hard for general graphs can be solved for graphs with bounded branch-width. The approach used for TSP in this paper can be used for other problems as well. The purpose of this section is to present a result that facilitates broader application of the approach:

Theorem 8 *There is a linear-time algorithm that, for any planar graph G and integer k, finds a decomposition $S_0, \ldots, S_{k-1}$ of the edges of G such that, for $i = 0, 1, 2, \ldots, k-1$, the graph obtained from G by contracting the edges of S_i has branch-width at most $2(k+2)$.*

The analogous theorem with contraction replaced by deletion was implicit in the work of Baker [6] and made explicit by Demaine, Hajiaghayi, and Karabayashi [18], who proved a version for H-minor free graphs.

An easy but useful consequence of Theorem 8 is as follows.

Corollary 28 *There is a linear-time algorithm, that, for any planar graph G, edge-weight assignment weight$(\cdot)$, and integer k, finds a set S of edges of weight at most $(1/k)$weight(G) whose contraction yields a graph of branch-width at most $2(k+2)$.*

Before proving Theorem 8, we review the definition of branch-width given by Seymour and Thomas. For a graph G and a set X of edges, $\partial(X)$ denotes the set of nodes v of G such that at least one edge incident to v is in X and at least one is not. Two sets *cross* if neither contains the other and they are not disjoint.

For a finite set $\mathcal{X}$, a *carving* of $\mathcal{X}$ is a family $\mathcal{C}$ of subsets of $\mathcal{X}$ such that

1. $\emptyset, \mathcal{X} \notin \mathcal{C}$,

2. no two members of $\mathcal{C}$ cross, and

3. $\mathcal{C}$ is maximal subject to 1 and 2.

Let G be a graph. Let $\mathcal{C}$ be a carving of $E(G)$. The branch-width of $\mathcal{C}$ in G is $\max_{X \in \mathcal{C}} |\partial(X)|$. The *branch-width* of G is the minimum, over all carvings $\mathcal{C}$ of $E(G)$, of the width of $\mathcal{C}$.

The following lemma is implicit in Baker's approach [6], and the idea has been used several times since then (e.g. [9, 22, 26])

Lemma 29 (Thinning Algorithm) *There is a linear-time algorithm that, for any planar embedded graph G and integer k, finds a decomposition of the edges into subsets $S_0, \ldots, S_{k-1}$ such that, for $i = 0, 1, 2, \ldots, k-1$, there is a planar embedded graph H_i with the following properties:*

1. *H_i has the same edges as G.*

2. *Each connected component of H_i has a spanning tree of depth at most k.*

3. *$H_i - S_i = G - S_i$.*

Proof: Assume without loss of generality that G is connected. For some node r,define the *level* of a node v of G to be its breadth-first-search distance from r, i.e. the minimum number of edges in an r-to-v path in G. Define the *level* of an edge e to be i if one endpoint has distance i from r and the other endpoint has distance $i+1$.

For $i = 0, 1, \ldots, k-1$, let S_i denote the set of edges e whose levels are congruent to i mod k.

For $i = 0, 1, \ldots, k-1$ and for $j = 0, 1, 2, \ldots$, let $H_i^{(j)}$ be the graph obtained from G by deleting all nodes at distances greater than $jk + i$, and contracting every edge e of the breadth-first-search tree T whose level is less than $(j-1)k + i$. The contractions coalesce into a single root all nodes at distances less than or equal to $(j-1)k + i$. (For $j = 0$, there is already a single root, namely r.) Since every node of G that remains in $H_i^{(j)}$ had distance at most $jk + i$ in G, it follows that (A) in $H_i^{(j)}$ every node has distance at most k from the root. Moreover, (B) the graph $H_i^{(j)} - \{$edges at level $(j-1)k + i$ in $G\}$ is exactly the subgraph of G induced by nodes with levels in $((j-1)k + i, jk + i]$.

Let H_i be the disjoint union of $H_i^{(0)}, H_i^{(1)}, H_i^{(2)}, \ldots$. Property 2 follows from A. Property 3 follows from B. Property 1 follows from Property 3 and the definition of S_i. $\square$

We first give a technical lemma, then we state a result of Tamaki [45], and then prove Theorem 8.

Lemma 30 *Let G be a planar embedded graph and let e be a self-loop in G. Every biconnected component of G except $\{e\}$ is a biconnected component of $G/\{e\}$.*

Proof: Let v be the common endpoint of e. Any path in G that contains e must pass through e, so the only simple cycle in G that contains e is the cycle consisting solely of e. Any path in G from an edge enclosed by e to an edge not enclosed by e must pass through v. Hence a simple cycle C in G cannot include both an edge strictly enclosed by e and an edge not enclosed by e. Assume without loss of generality that C consists only of edges strictly enclosed by e. Any two such edges incident to a common node in G are also incident to a common node in $G/\{e\}$, which proves that C is a simple cycle in $G/\{e\}$. $\square$

For a planar embedded graph G, Tamaki [45] defined $VF(G)$ to be the node-face incidence graph, i.e. the planar embedded bipartite graph whose node set is the union of the node set of G and the face set of G, and where there is an edge between a node and a face if the node is on the boundary of the face. Tamaki proved the following theorem.

Theorem 9 (Tamaki) *There is a linear-time algorithm that, given a planar embedded graph G and a rooted spanning tree T of $VF(G)$, outputs a branch-decomposition of G whose width is at most the height of T.*

Proof of Theorem 8: Apply Lemma 29 to G^* to find a decomposition of the edges into subsets $L_0, \ldots, L_{k-1}$ such that, for each i, there is a planar embedded graph H_i^* such that $H_i^* - L_i = G^* - L_i$ and each connected component of H_i^* has a spanning tree of depth at most k. The nodes of H_i^* are faces of H_i, so $VF(H_i)$ has a spanning tree of depth at most $2k + 1$. By Tamaki's result, therefore, H_i has branch-width at most $2k + 1$. It is known [41] that contraction does not increase branch-width. Hence H_i/L_i has branch-width at most $2k+1$. Since $H_i^* - L_i = G^* - L_i$, we have $H_i/L_i = G/L_i$, so we have shown that G/L_i has branch-width at most $2k + 1$.

Consider the process of obtaining G/L_i by compressing the edges of L_i one by one in an order that postpones compressing self-loops until only self-loops remain to be compressed. (Recall that a self-loop corresponds in the dual to cut-edges, so this corresponds in the dual to first deleting only non-cut-edges.)

Let S_i be the set of edges e of L_i such that e was not a self-loop when it was compressed, and let L_i' be the remaining edges of L_i. Then G/S_i is obtained from G by *contracting* the edges of S_i, and G/L_i is obtained from G/S_i by compressing the edges of L_i'. By Lemma 30, every biconnected component of G/S_i is a biconnected component of G/L_i. Since the branch-width of a graph is at most 1 more than the maximum of the branch-widths of its biconnected components (assuming at least one such component has more than one edge), it follows that that branch-width of G/S_i is at most one plus the branch-width of G/L_i. $\square$

Acknowledgements

Claire Kenyon and Mikkel Thorup contributed greatly to the quality of the presentation through their criticisms and suggestions. Thanks also to Glencora Borradaile and Erik Demaine for helpful discussions and suggestions.

References

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, L. Soares, "On sparse spanners of weighted graphs," *Discrete and Computational Geometry* 9 (1993), pp. 81–100.

[2] S. Arora, "Polynomial-time approximation schemes for Euclidean TSP and other geometric problems," *Journal of the ACM* **45** (1998), pp. 753–782.

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

[4] S. Arora, “Nearly linear time approximation schemes for Euclidean TSP and other geometric problems,” *Proceedings of the 38th Annual IEEE Foundations of Computer Science* (1997), pp. 554–563.

[5] S. Arora, M. Grigni, D. R. Karger, P. N. Klein, A. Woloszyn, “ A polynomial-time approximation scheme for weighted planar graph TSP,” *Proceedings of the 9th Annual ACM-SIAM Symposium on Discrete Algorithms* (1998), pp. 33–41.

[6] B. Baker. “Approximation algorithms for NP-complete problems on planar graphs,” *Journal of the ACM* **41** (1994), pp. 153–180.

[7] A. Berger, A. Czumaj, M. Grigni, and H. Zhao. Approximate minimum 2-connected subgraphs in weighted planar graphs. *Proceedings of the 13th Annual European Symposium on Algorithms* (2005), pp. 472–483.

[8] A. Berger, M. Grigni, “Minimum weight 2-edge-connected spanning subgraphs in planar graphs,” *Proceedings of the 34th International Colloquium on Automata, Languages and Programming* (2007), pp. 90-101.

[9] H. L. Bodlaender, “Some classes of graph with bounded treewidth,” *Bulletin of the European Association for Theoretical Computer Science* **36**, (1988), pp. 116-126.

[10] G. Borradaile, C. Kenyon-Mathieu, and P. Klein, “A polynomial-time approximation scheme for Steiner tree in planar graphs,” *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), pp. 1285-1294.

[11] G. Borradaile, P. Klein, and C. Mathieu, “Steiner tree in planar graphs: An $O(nlogn)$ approximation scheme with singly exponential dependence on epsilon,”. *Proceedings of the 10th International Workshop on Algorithms and Data Structures* (2007), v. 4619 of *Lecture Notes in Computer Science*, pp. 275-286.

[12] J. M. Boyer and W. J. Myrvold, “Simplified planarity,” *Journal of Graph Algorithms and Applications* **8** (2004), pp. 241–273.

[13] D. R. Cheriton, R. E. Tarjan, “Finding minimum spanning trees,” *SIAM Journal on Computing* **5** (1976), pp. 724–742.

[14] 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* (1976), p. 441. Academic Press, New York.

[15] W. J. Cook and P. D. Seymour, “Tour Merging via Branch-decomposition,” *INFORMS Journal on Computing* **15** (2003), pp. 233–248.

[16] A. Czumaj, M. Grigni, P. Sissokho, and H. Zhao, “Approximation schemes for minimum 2-edge-connected and biconnected subgraphs in planar graphs,” *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms* (2004), pp. 489–498.

[17] E. D. Demain, M. Hajiaghayi, “Bidimensionality: new connections between FPT algorithms and PTASs,” *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms* (2005), pp. 590–601.

[18] E. D. Demaine, M. T. Hajiaghayi, and K. Karabayashi, “ Algorithmic graph minor theory: decomposition, approximation, and coloring,” *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (2005), pp. 637–646

[19] E. D. Demaine, M. T. Hajiaghayi, and B. Mohar, “Approximation algorithms via contraction decomposition,” *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms* (2007), pp. 278-287.

[20] F. Dorn, E. Penninkx, H. L. Bodlaender, F. V. Fomin,”Efficient exact algorithms on planar graphs: exploiting sphere-cut branch decompositions,” *Proceedings of the 13th Annual European Symposium on Algorithms* (2005), pp. 95–106.

[21] J R. Edmonds, “A combinatorial representation for polyhedral surfaces,” *Notices of the American Mathematical Society* **7** (1960), p. 646.

[22] D Eppstein, “ Subgraph isomorphism in planar graphs and related problems,” *Journal of Graph Algorithms and Applications* **3** (1999) pp. 1–27.

[23] M. Grigni, E. Koutsoupias, and C. H. Papadimitriou, “An approximation scheme for planar graph TSP,” *Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science* (1995), pp. 640–645.

[24] M. Grigni and P. Sissokho, “Light spanners and approximate TSP,” *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms* (2002), pp. 852–857.

[25] M. Grigni, “Approximate TSP in graphs with forbidden minors,” *Proceedings of the 27th International Colloquium on Automata, Languages, and Programming* (2000), pp. 869-877.

[26] M. Grohe, “Local tree-width, excluded minors, and approximation algorithms,” *Combinatorica* **23** (2003), pp. 613-632.

[27] L. Heffter, “Über das Problem der nachbargebiete,” *Mathematische Annalen* **38** (1891), pp. 477-508.

[28] M. R. Henzinger, P. N. Klein, S. Rao, and S. Subramanian, “Faster shortest-path algorithms for planar graphs,” *Journal of Computer and System Sciences* **55** (1997), pp. 3–23.

[29] J. Hopcroft and R. E. Tarjan, “Efficient planarity testing,” *Journal of the ACM* **21** (1974), pp. 549–568.

[30] P. N. Klein, “A linear-time approximation scheme for planar weighted TSP,” *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science* (2005), pp. 647-657.

[31] P. N. Klein, “A subset spanner for planar graphs, with application to subset TSP,” *Proceedings of the 38th Annual ACM Symposium on Theory of Computing* (2006), pp. 749–756.

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

[33] D. Marx, ”On the optimality of planar and geometric approximation schemes,” f*Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science* (2007), pp. 338-348.

[34] J. S. B. Mitchell, “Guillotine subdivisions approximate polygonal subdivisions: Part II– a simple PTAS for geometric k-MST, TSP, and related problems,” *SIAM Journal on Computing* **28** (1999), pp. 298–1309.

[35] B. Mohar and C. Thomassen, *Graphs on Surfaces* (2001), The Johns Hopkins University Press, Baltimore.

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

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

[38] S. B. Rao and W. D. Smith, “Approximating geometrical graphs via ’spanners’ and ‘banyans,”’ *Proceedings of the 30th Annual ACM Symposium on Theory of Computing* (1998), pp. 540–550.

[39] N. Robertson and P. D. Seymour, "Graph Minor III. Planar Tree-Width," *Journal of Combinatorial Theory, Series B* **36** (1984), pp. 49–63.

[40] N. Robertson and P. D. Seymour, "Graph minors IV: Tree-width and well-quasi-ordering," *Journal of Combinatorial Theory, Series B* **48** (1990), pp. 227–254.

[41] N. Robertson and P. D. Seymour, "Graph minors X. Obstructions to tree-decomposition," *Journal of Combinatorial Theory, Series B* **52** (1991), pp. 153–190.

[42] P. D. Seymour and R. Thomas, "Call routing and the ratcatcher," *Combinatorica* **14** (1994), pp. 217–241.

[43] R. P. Stanley, *Enumerative Combinatorics, Vol. 1* (2000), Cambridge University Press, Cambridge.

[44] K. Talwar, "Bypassing the embedding: algorithms for low-dimensional metrics," *Proceedings of the 36th Annual ACM Symposium on Theory of Computing* (2004), pp. 281–290.

[45] H. Tamaki, "A linear-time heuristic for the branch-decomposition of planar graphs," *Proceedings of the 11th Annual European Symposium on Algorithms* (2003), pp. 765-775.

[46] J.W.T. Youngs, "Minimal imbeddings and the genus of a graph," *J. Math. Mech.* **12** (1963), pp. 303–315.