ELSEVIER

Discrete Applied Mathematics 108 (2001) 85–103

DISCRETE APPLIED MATHEMATICS

Tree spanners in planar graphs ☆

Sándor P. Fekete[a, *], Jana Kremer[b]

[a] *Department of Mathematics, Technical University Berlin, Str. des 17 Juni 15, D-10623 Berlin, Germany*

[b] *Lehrstuhl für Volkswirtschaftslehre, Otto-Friedrich Universität Bamberg, D-96045 Bamberg, Germany*

Received 19 October 1998; revised 13 January 2000; accepted 28 April 2000

Abstract

A tree t-spanner of a graph G is a spanning subtree T of G in which the distance between every pair of vertices is at most t times their distance in G. Spanner problems have received some attention, mostly in the context of communication networks. It is known that for general unweighted graphs, the problem of deciding the existence of a tree t-spanner can be solved in polynomial time for $t = 2$, while it is NP-hard for any $t \geqslant 4$; the case $t = 3$ is open, but has been conjectured to be hard. In this paper, we consider tree spanners in planar graphs. We show that even for planar unweighted graphs, it is NP-hard to determine the minimum t for which a tree t-spanner exists. On the other hand, we give a polynomial algorithm for any fixed t that decides for planar unweighted graphs with bounded face length whether there is a tree t-spanner. Furthermore, we prove that it can be decided in polynomial time whether a planar unweighted graph has a tree t-spanner for $t = 3$.

Keywords: Graph spanners; Planar graphs; Distance in graphs

1. Introduction

A t-spanner of a graph G is a spanning subgraph H of G in which the distance between every pair of vertices is at most t times their distance in G. We can think of the "stretch factor" t as the relative price increase that has to be incurred for individual connections after replacing the network G by a cheaper subnetwork H. Spanners were first considered in the context of practical motivations from communication networks (see [20], who introduced spanners to synchronize asynchronous

☆ An extended abstract version [13] of this paper appears in the *Proceedings of the 24th International Annual Workshop on Graph–Theoretic Concepts in Computer Science* (WG 98). This work originates from the second author's thesis [16].

* Corresponding author.

E-mail addresses: fekete@math.tu-berlin.de (S.P. Fekete), jana.kremer@sowi.uni-bamberg.de (J. Kremer).

0166-218X/01/$ - see front matter
PII: S0166-218X(00)00226-2

networks). They have also been used for simplifying geometric data structures (see [9,11,2]). Surveys of results on the existence and efficient constructibility can be found in [19,22].

Depending on the objective for choosing a subnetwork, various kinds of spanners have been considered (see [1,3,15,18,20,21] for a selection of variants). Since the main motivation is to obtain a network of small total weight, particular attention has focused on *tree spanners*, where the subnetwork H is minimal with respect to edge removal. As Cai [6,7], and Cai and Corneil [8] showed, the problem of deciding the existence of a tree t-spanner in an unweighted graph G can be solved in polynomial time for $t = 2$; on the other hand, the problem is NP-complete for any $t \geqslant 4$. The case $t = 3$ is still open, but it was conjectured in [8] to be NP-complete.

As noted above, spanners have been considered in the context of geometric distance queries (see [9,11,2]). Since planar graphs form a particularly well-understood class of sparse graphs with a number of structural and algorithmic properties that make them interesting as spanners, the focus of those works has been on *planar spanners*, where the spanning graph H is required to be planar. Also, see [5] for a proof that it is NP-hard to determine a minimum weight planar t-spanner in a graph. They also showed that determining a minimum weight t-spanner in a planar graph is an NP-hard problem. Kortsarz [14] has obtained hardness results on the approximation of a related optimization problem: He shows that finding a t-spanner with a small number of edges is at least as hard as approximating the set cover problem, implying a lower bound of $\Omega(\log n)$ for an approximation factor, unless $\mathrm{P} = \mathrm{NP}$. Dodis and Khanna [12] improved this result by showing that under the same assumption, there is a lower bound of $\Omega(2^{\log^{\varepsilon} n})$ on the approximation ratio, for fixed $t \geqslant 5$.

Between considering tree spanners in general graphs and planar spanners in general graphs, it is natural to consider tree spanners in planar graphs. Not only does this allow a better understanding of the properties of graph spanners, but results on the stretch factors of tree spanners in planar graphs combine with bounds on the stretch factors of planar spanners in general graphs to yield estimates on tree spanners in general graphs.

In this paper, we show that deciding the existence of a tree t-spanner in a graph G is NP-complete, if t is part of the input, even when restricted to the situation where G is planar and unweighted. On the other hand, we prove that this problem can be solved in polynomial time for planar unweighted graphs with bounded face length and fixed t.

For some purposes and some graphs, not all pairs of connections have the same importance. This motivates the concept of (s,t)-spanners: For a partition of $E(G)$ into two given sets of edges E_1 and E_2, a tree (s,t)-spanner consists of edges in E_1, and it replaces any edge $(v_1, v_2) \in E_1$ by a path of at most s times its length, and any edge $(v_1, v_2) \in E_2$ by a path of at most t times its length. We show that for fixed s and t, the existence of a tree (s,t)-spanner in planar unweighted graphs with bounded face length can be checked in polynomial time. By a detailed analysis of the neighborhood structures of planar graphs with tree 3-spanners, we are able to show that a planar graph has a tree 3-spanner, iff it is a subgraph of a planar graph with bounded face

length that has a tree $(3,8)$-spanner. This implies a polynomial algorithm for deciding whether a planar graph G has a tree 3-spanner.

The rest of this paper is organized as follows: In Section 2, we introduce some basic concepts. Section 3 contains the NP-completeness of deciding the existence of a tree t-spanner in a planar graph. In Section 4, we describe the polynomial algorithm for deciding whether a planar graph with bounded face length has a tree (s,t)-spanner. Section 5 gives an overview of the polynomial algorithm for deciding whether a planar graph has a tree 3-spanner. In Section 6 we conclude with some open problems.

2. Preliminaries

Throughout this paper, we use the terminology of Bondy and Murty [4]. A graph G has edge set $E(G)$ and vertex set $V(G)$; we may simply write E and V when the meaning is clear. If H is a subgraph of G, then $G-H$ denotes the graph obtained by deleting from G all edges of H. If G and H are graphs on the vertex set V, then $G+H$ is the graph obtained by adding the edges of H to G. For a pair of vertices v_1 and v_2 in a connected graph G, we denote the length of a shortest path from v_1 to v_2 by $d_G(v_1,v_2)$. We will concentrate on the case of unweighted graphs without loops, so for any edge $(v_1,v_2)\in E(G)$, we have $d_G(v_1,v_2)=1$. For a planar graph G, we write G^* for the dual graph. For $S\subset V$, the number of the edges leaving S in the graph G is denoted by $\delta_G(S)$. For $S\subset V$, we denote by $N(S)$ the set of neighbors of S, i.e., the set of vertices $v\in V\setminus S$ with a $w\in S$, such that $(v,w)\in E$. For a set of vertices $S\subseteq V$, the subgraph induced by S is denoted by $G[S]$.

For a real number $t\geqslant 1$, a subgraph H of a connected graph G is a *t-spanner* if $d_H(v_1,v_2)\leqslant t\cdot d_G(v_1,v_2)$ for all $v_1,v_2\in E(G)$. A *tree t-spanner* is a t-spanner that is a tree. The parameter t is called the *stretch factor*; the smallest value t for which a graph G has a tree t-spanner is called the *tree stretch index* of G, denoted by $\sigma_T(G)$. It was shown in [8] that the following holds:

Lemma 1. *A subgraph H of a connected graph G is a t-spanner, iff for all edges $(v_1,v_2)\in E(G)-E(H)$, we have $d_H(v_1,v_2)\leqslant t$.*

This allows us to consider only integer stretch factors for unweighted graphs. If the condition $d_H(v_1,v_2)\leqslant t$ is satisfied for a particular edge $e=(v_1,v_2)\in E(G)-E(H)$, we say that e has a *short detour* in H; for the case of tree spanners T, there is a unique corresponding shortest path, denoted by $p_T(e)$.

3. An NP-completeness result

It was shown in [8] that it is NP-complete to decide whether $\sigma_T(G)\leqslant t$ for a general unweighted graph, as long as $t\geqslant 4$. In this section, we describe our proof that it is

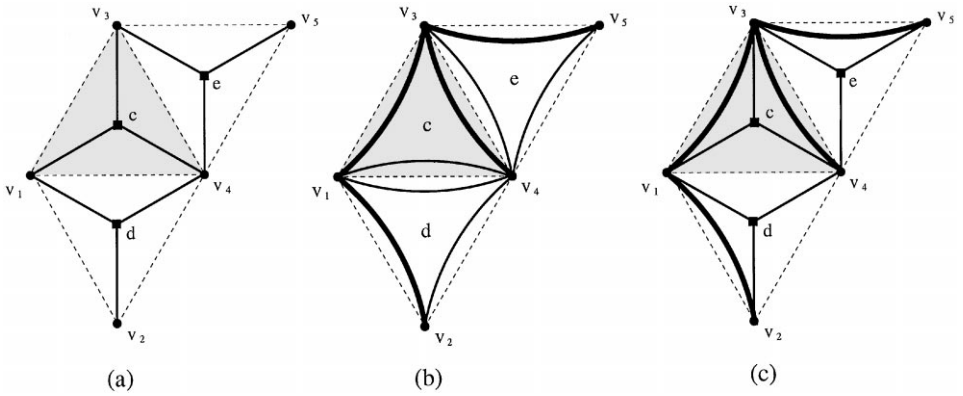

Fig. 1. (a) The graph G_I representing the PLANAR 3SAT instance $(x_1 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_4) \wedge (\bar{x}_3 \vee x_4 \vee \bar{x}_5)$; (b) the transformed graph G'_I and a spanning tree T' in G'_I (bold); (c) the graph $G_I + T'$.

NP-complete to decide $\sigma_T(G) \leqslant t$ for a planar unweighted graph, where t is part of the input. Our reduction is from a special subclass of 3-SAT instances, called PLANAR 3SAT, which was shown to be NP-complete by Lichtenstein [17].

A 3SAT instance I is said to be an instance of PLANAR 3SAT, if the following bipartite graph G_I is planar: Every variable and every clause in I is represented by a vertex in G_I; two vertices are connected, if and only if one of them represents a variable that appears in the clause that is represented by the other vertex (see Fig. 1(a) for an example).

In the following, we describe the necessary gadgets for our hardness proof.

3.1. The basic setup

As a first step, the graph G_I is transformed into a graph G'_I. As shown in Fig. 1, each set of three edges adjacent to the same clause vertex is replaced by a triangle – a so-called (Y, Δ)-transformation. From this graph G'_I, *any* spanning tree T' is chosen. This spanning tree has a certain stretch factor t', which is polynomially bounded by the size of I.

For the second step, we add the edges of T' to the graph G_I, as shown in Fig. 1. This yields the graph $G_I + T'$.

As a third step, the variable vertices v_r in $G'_I + T$ are replaced by variable gadgets G^r_{var} (described in the following subsection), each with two disjoint subsets of edges E^i_{true} and E^i_{false}; furthermore, edges adjacent to clause vertices are replaced by "path" gadgets, and edges of T' are replaced by suitable "edge" gadgets. This is done in a way that for the resulting graph G''_I, any spanning tree T is a tree t-spanner, iff

(1) certain edges are contained in T;
(2) for certain pairs of edges, precisely one edge is contained in T;
(3) for each variable v_r, either all edges of E^r_{true} are contained in T, or all edges in E^r_{false} are;

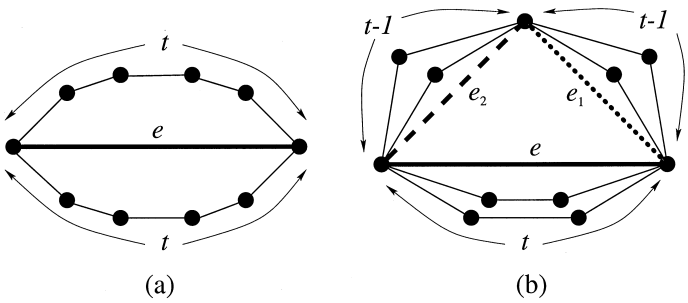

Fig. 2. (a) A forced edge; (b) a forced pair.

(4) for each clause c_j, precisely one of the three adjacent path gadgets provides a connection to an adjacent variable vertex v_r, denoted by v_j;
(5) for each clause c_j and the adjacent variable vertices v_i, the truth setting provided by one of the sets E^i_{true} or E^i_{false} satisfies c_j.

For the first property, we use the gadget shown in Fig. 2(a). It has been used extensively in the proofs of [8,5]. It is easy to see that any tree t-spanner of the graph G shown in the figure must contain the edge e. In the following, edges *forced* in this way are indicated by bold drawing.

Fig. 2(b) shows another gadget that can be used for forcing one out of two edges: For $t \geqslant 3$, any tree t-spanner must contain e and precisely one of the two edges e_1 and e_2.

The following subsections give a description of the remaining gadgets and their properties.

3.2. Gadgets for variables

Consider a variable vertex $v_r \in G_I$. For simplicity of notation, we omit superscripts r in this subsection.

Fig. 3 shows the gadget G_{var} for representing variables. It consists of a central "variable" vertex v, connected to "literal" vertices $v_1, \overline{v_1}, \ldots, v_s\ \overline{v_s}$ (with $s = \delta(v)$), by "true" edges $e_i = (v, v_i)$ and "false" edges $\overline{e_i} = (v, \overline{v_i})$. Any v_i and $\overline{v_i}$ are connected by an edge w_i that is forced by two paths of length $t \geqslant 3$ as shown in Fig. 2(a). $\overline{v_i}$ and v_{i+1} (indices modulo s) are connected by a path of length $t-2$, containing the vertices $\overline{v_i}, w_{(i,1)}, \ldots, w_{(i,t-3)}, v_{i+1}$. The edge $f_i = (\overline{v_i}, w_{(i,1)})$ is not forced, all other edges of the path are. Connections to the outside, i.e., to the rest of the graph, are at the literal vertices.

Furthermore, variable gadgets will be used in a way that outside connections are "long". More precisely, for any two literal vertices with different indices (i.e., v_i or $\overline{v_i}$, and v_j or $\overline{v_j}$ with $i \neq j$), a shortest path that does not use any edges in G_{var} contains at least 4 edges. If this condition is satisfied, we say that G_{var} has *long outside connections*.

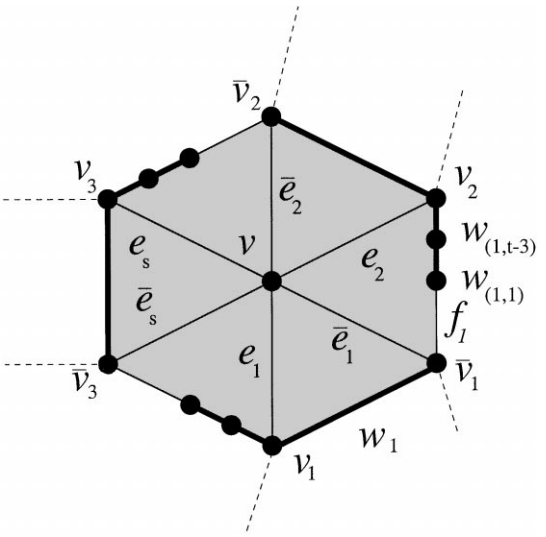

Fig. 3. Variable component G_{var}.

Now it is not hard to prove the following:

Lemma 2. *Suppose that G_{var} is contained in a graph G and has long outside connections. Then any tree t-spanner of G contains precisely one of the edges e_i, $\overline{e_i}$ for any $i = 1,\ldots,s$, and none of the edges f_i can be contained. Furthermore, the choice of e_i or $\overline{e_i}$ is the same for all $i = 1,\ldots,s$, i.e., if one edge e_i is contained then all edges e_i are contained.*

Proof. Consider a tree t-spanner T of G. Since v must be connected to the rest of the graph, for some i, one of the edges e_i, $\overline{e_i}$ must be contained in T. Since w_j must be contained in T, and T is cycle-free, for any j, not both e_j and $\overline{e_j}$ can be contained.

Without loss of generality, consider the case where e_1 is contained (the case of $\overline{e_1}$ will be treated later). If f_1 was contained in T, then neither $\overline{e_1}$, nor e_2, nor $\overline{e_2}$ could be contained in T. But then $p_T(\overline{e_2})$ has more than t edges, contradicting the fact that T is a t-spanner. Therefore, f_1 cannot be contained. Now consider $p_T(f_1)$. Since $p_T(f_1)$ has at most t edges, it must contain a path of length 3 from $\overline{v_1}$ to v_2. By assumption, there is no such path outside of G_{var}; inside of G_{var}, it is easy to see that such a path can only consist of the edges w_1, e_1, and e_2, showing that e_2 is contained in T. It follows by induction that no f_i is contained in T, but all e_i are.

The case where $\overline{e_1}$ is contained in T is treated analogously: first we conclude that f_s cannot be contained in T, then proceed to show that $\overline{e_S}$ must be contained, etc. □

In the following, containment of $E_{\text{true}} := \{e_i \mid i = 1,\ldots,s\}$ in a tree t-spanner will correspond to a "true" setting of the represented variable, while containment of $E_{\text{false}} := \{\overline{e_i} \mid i = 1,\ldots,s\}$ corresponds to a setting of "false".

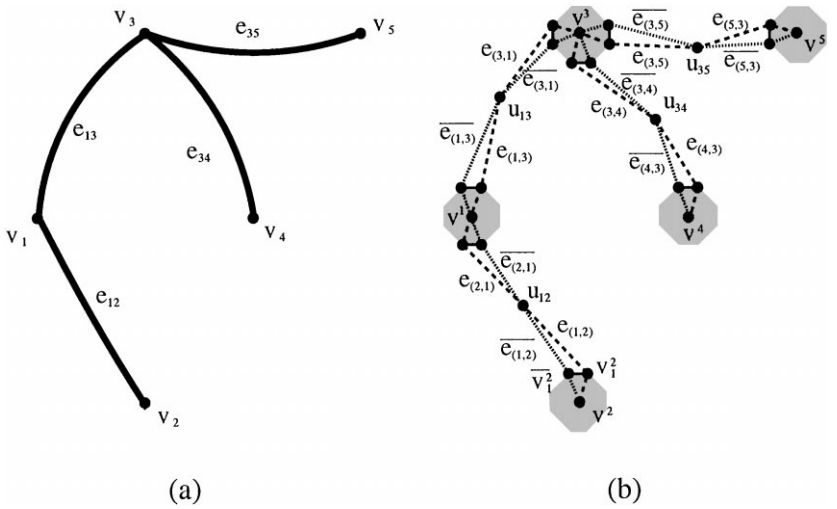

Fig. 4. (a) A spanning tree T'; (b) a representation of T' after introducing variable gadgets and edge gadgets.

3.3. Edge gadgets

After replacing variable vertices by variable gadgets as described in the previous section, any edge $e_{ij} \in T'$ between two variable vertices v_i and v_j is represented by the edge gadget in Fig. 4. It consists of three edges of each of the two variable gadgets, connected to a vertex u_{ij} by two pairs of edges $e_{(i,j)}, \overline{e_{(i,j)}}$, and $e_{(j,i)}, \overline{e_{(j,i)}}$. For any pair of edges $e_{(i,j)}, \overline{e_{(i,j)}}$, we use the gadget shown in Fig. 2(b) to make sure that precisely one of them will be contained in a tree t-spanner. Eventually, $e_{(i,j)}$ will be chosen if E^i_{true} is contained in T, and $\overline{e_{(i,j)}}$ will be chosen if E^i_{false} is contained in T.

It is straightforward to see that in the resulting graph $\tilde{G}_I$, all variable gadgets have long outside connections, so Lemma 2 applies. It follows that for any tree t-spanner T in $\tilde{G}_I$, any edge $e = (v_i, v_j)$ of T' is represented by a path of length 4 between v^i and v^j.

3.4. Clause gadgets

The basic idea is shown in Fig. 5.

Consider a clause vertex c in the graph G_I that is adjacent to the variable vertices v_1, v_2, v_3. In T', consider the three paths connecting v_1 with v_2, v_1 with v_3, and v_2 with v_3. These paths must induce a subtree $\tilde{T}$, consisting of three subpaths P_i, connecting v_i to a central vertex u. (Note that $u = v_i$ is possible.) Let $d_i = d_{T'}(v_i, u)$. Since T' has stretch factor t', we know that $d_i \leqslant t'$. Recall that each edge of T' is replaced by a set of four edges in T, as noted in Section 3.3, so $d_T(v^i, u) = 4d_i$.

For each variable i, connect c to a literal vertex of G^i_{var} by a path of length $4(t' - d_i) + 1$ as follows: If x_i appears in the clause, connect c to a literal vertex v^i_k

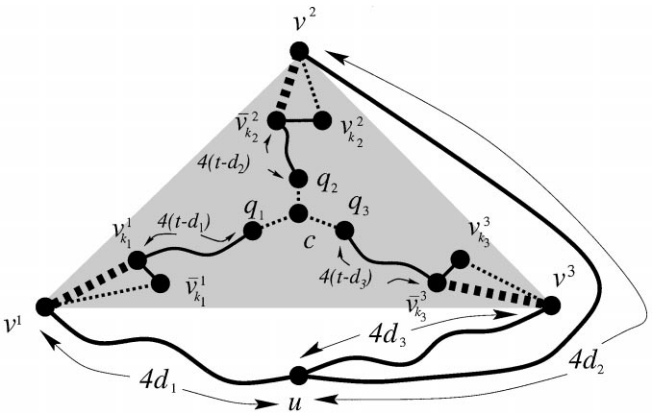

Fig. 5. Idea of the clause component, shown for $(x_1 \vee \overline{x_2} \vee \overline{x_3})$.

for a suitable $k_i \in \{1,\ldots,s\}$. If $\overline{x_i}$ appears in the clause, connect c to a literal vertex $\overline{v^i_k}$.

Of the $4(t'-d_i)+1$ edges in a path, all $4(t'-d_i)$ edges that are not adjacent to vertex c are forced by paths of length t. The remaining (unforced) edge is denoted by $g_i=(c,q_i)$.

3.5. The reduction

Using the above gadgets and properties, we claim:

Theorem 3. *It is NP-complete to decide $\sigma_T(G)\leqslant t$ for planar unweighted graphs G and integers t.*

Proof. Use the above construction, where $t=8t'+4$, and see Fig. 5. It is clear that the existence of a tree t-spanner T hinges on the appropriate choice of connections g_i at any clause vertex c to the adjacent vertices q_1, q_2, q_3, and thus to the vertices v^1, v^2, v^3. As seen above, in any tree t-spanner, v^i and v^j must be connected by a path that does not visit c and has length $4(d_i+d_j)$. Therefore, not both g_i and g_j can be contained in T, or there would be a cycle in T. Without loss of generality, assume that g_1 is contained in T. Suppose that the truth setting of variable 1, provided by one of the two edge sets E^1_{true}, E^1_{false}, does not satisfy clause vertex c. Then in T, c is connected to v^1 by a path of length $4(t'-d_1)+3$, which extends to a path of length $4(t'-d_1)+3+4(d_1+d_2)=4t'+4d_2+3$ from c to v^2. Since $p_T(g_1)$ has length at most $t=8t'+4$, there must be a path from v^2 to q_2 in T of length at most $t-4t'-4d_2-3=4t'-4d_2+1$. This means that the truth assignment of variable 2 must satisfy clause vertex c, showing that a tree t-spanner induces a satisfying truth assignment of the PLANAR 3SAT instance I.

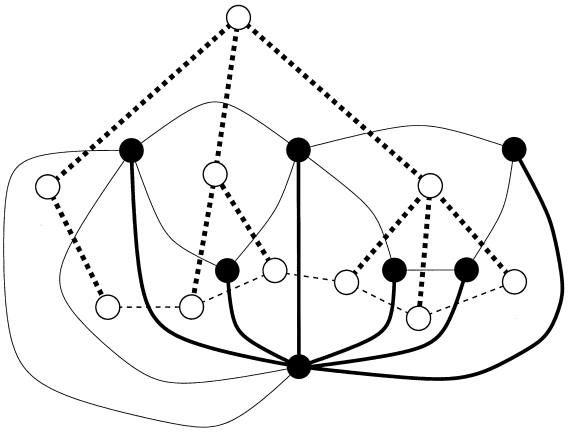

Fig. 6. A planar graph G (solid lines, black vertices) with a tree 2-spanner (bold edges), and the dual graph G^* (broken lines, white vertices) with a 3-cut tree (bold edges).

Conversely, it is straightforward to verify that a truth assignment of the PLANAR 3SAT instance I can be used to construct a tree t-spanner: Choose the corresponding edge sets E^i_{true} or E^i_{false}, $e_{(i,j)}$ or $\overline{e_{(i,j)}}$, and the connection g_i for a satisfying variable to be contained in T. □

4. Planar graphs with bounded face length

In this section we show that deciding the existence of a tree t-spanner in a planar graph with all faces of bounded length can be performed in polynomial time.

For this purpose, we introduce the notion of a *c-cut tree* in a graph:

Definition 4. Let T be a spanning tree in a graph G. Removing any edge $e \in T$ splits T into two connected components, inducing a partition of the vertex set into $P_T(e) = (V_T(e), V'_T(e))$. We say that T is a c-cut tree in G, if for all $e \in T$, $|\delta_G(V_T(e))| \leqslant c$.

It is straightforward to show that the following holds (see Fig. 6 for an example).

Lemma 5. *A planar graph G has a tree t-spanner, iff G^* has a $(t+1)$-cut tree.*

For a proof, note that there is a 1-to-1 correspondence between edges e_c in a $(t+1)$-cut tree and edges e in $G \setminus T$ for a tree t-spanner. The size of the cut induced by e_c is one less than the path in T that connects the vertices of e in T.

The set of cuts induced by the edges of the dual $(t+1)$-cut tree have a noncrossing, i.e., "nested" or "laminar" structure. (Nested families of objects play a role in many problems of combinatorics and optimization, see [10] for an application in the context of matchings.) For describing cut trees, we need additional information about

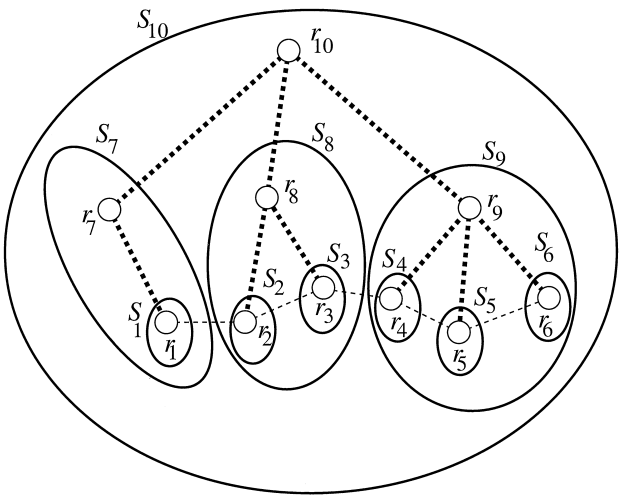

Fig. 7. A planar graph G^* with a 3-cut tree (bold edges), and the corresponding rooted nested family.

the relative structure of the nested sets. For this purpose, we introduce the notion of *rooted nested families* that can be used for establishing the following constructive characterization of c-cut trees:

Lemma 6. *A planar graph $G^*=(V^*,E^*)$ has a c-cut tree, iff there is a "rooted nested family" $F\subseteq 2^{V^*}\times V^*$ with the following properties*:
(1) $(V^*,r)\in F$ *for an* $r\in V^*$,
(2) $r\in S$ *for all* $(S,r)\in F$,
(3) $|\delta_G(S)|\leqslant c$ *for all* $(S,r)\in F$,
(4) *for all* (S_1,r_1), $(S_2,r_2)\in F$ *we have* $S_1\subseteq S_2$, *or* $S_1\subseteq V^*\setminus S_2$,
(5) *for all* $(S,r)\in F$ *with* $|S|>1$, *there are* $l\geqslant 1$ *and* $(S_i,r_i)\in F$, $1\leqslant i\leqslant l$, *with* $S\setminus\{r\}=\dot{\bigcup} S_i$ *and* $(r,r_i)\in E^*$.

Proof. See Fig. 7 for an example.

Suppose that we have a rooted nested family as described in the lemma. Consider a vertex $v=r$ with $(V,r)\in F$ as given by condition (1). By condition (5), the set $V\setminus\{r\}$ can be partitioned into disjoint sets S_i, with $(S_i,r_i)\in F$, and $(r,r_i)\in E^*$. While there is any unpartitioned S_i with $|S_i|>1$, we can continue to apply condition (5), and because of condition (2), this procedure must terminate. Clearly, the set of directed edges (r,r_i) encountered during the procedure forms a spanning arborescence A. (Because of condition (4), there cannot be any cycles.) The spanning tree C induced by A is a c-cut tree: By condition (3), any edge (r,r_i) induces a cut not larger than c.

To see the converse, consider a c-cut tree C. Now consider the following inductive procedure:

While there are unmarked edges in C, there must be a vertex v that has precisely one unmarked edge $e = (v, w)$ adjacent to it. Marking e leaves the set of unmarked edges connected. Removing e from C induces a partition $(S, \bar{S})$ of V. By assumption, $|\delta_{G^*}(S)| \leqslant c$. Without loss of generality, we may assume that $v \in S$, and the subgraph C_S of C induced by S does not contain any unmarked edges. If $|S| > 1$ then there must be marked edges in S that are adjacent to v. By construction, each of these edges e_i connects v to a vertex $u_i \in S$, and there is an $(S_i, u_i) \in F$ with $S_i \subset S$. Furthermore, for $u_i \neq u_j$, S_i and S_j are disjoint.

If e was the last unmarked edge, then for all vertices q_j adjacent to w, for some S_j, (S_j, q_j) has been added to F. Again, all these S_j are disjoint. This allows it to add (V, w) to F and satisfy condition (1). By construction, all other conditions are also satisfied.

This concludes the proof. □

Using the characterization from Lemma 6, we get the following result:

Theorem 7. *For fixed t, it can be decided in polynomial time for planar unweighted graphs G with bounded face length whether $\sigma_T(G) \leqslant t$.*

Proof. Consider the existence of a rooted nested family F of G^* as described in Lemma 6. Since t is fixed, there are only polynomially many possible cuts in G^* of size not larger than $t + 1$, implying we only have to consider polynomially many sets (S, r) that can be used for F. Since all faces in G have bounded length, the dual graph G^* has bounded degree, so there is a polynomial number of possible partitions for any (S, r). Using dynamic programming and proceeding by increasing size of S, we can decide in polynomial time whether there exists a rooted nested family as described in Lemma 6. □

As described in the introduction, the concept of tree t-spanners can be generalized:

Definition 8. Let G be a graph with $E(G) = E_1 \dot{\cup} E_2$. Then, a spanning tree T of G is a tree (s, t)-spanner for $G = (V, E_1 \dot{\cup} E_2)$, iff T is a subgraph of (V, E_1), and for all edges $(v_1, v_2) \in E_1 - T$, we have $d_T(v_1, v_2) \leqslant s$, and for all edges $(v_1, v_2) \in E_2 - T$, we have $d_T(v_1, v_2) \leqslant t$.

An analogous approach to the one for tree t-spanners can be used for obtaining similar results for tree (s, t)-spanners: Instead of $(t + 1)$-cut trees, consider $(t + 1, s + 1)$-cut trees, where a cut induced by an edge e is bounded by $s + 1$, if $e \in E_1$, and bounded by $t + 1$, if $e \in E_2$. Similarly, we adapt the definition of rooted nested families, and the resulting dynamic programming algorithm. Summarizing, we get:

Theorem 9. *For fixed s and t, it can be decided in polynomial time for planar unweighted graphs $G = (V, E_1 \dot{\cup} E_2)$ of bounded face length, possibly with multi-edges, whether there is a tree (s, t)-spanner.*

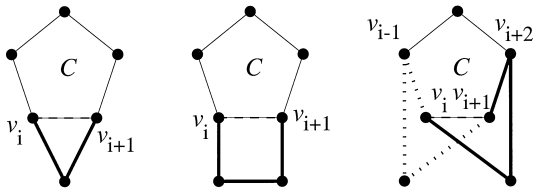

Fig. 8. Different possibilities for a short detour $p_T(v_i, v_{i+1})$ of an edge (v_i, v_{i+1}) in $C - T$.

5. Deciding the existence of tree 3-spanners in planar graphs

In this section, we describe the polynomial algorithm for deciding whether a planar unweighted graph G has a 3-spanner. The key idea is to add a set of edges E' to obtain a graph $G_{\leqslant 4}$ with face length bounded by 4, such that $G_{\leqslant 4} = (V, E \dot{\cup} E')$ has a tree $(3,8)$-spanner, iff G has a tree 3-spanner. The existence of a $(3,8)$-spanner in $G_{\leqslant 4}$ can be decided in polynomial time by the algorithm from the previous section.

The neighborhood structure of a face boundary determines which edges have to be inserted to subdivide the corresponding face. Therefore, we begin our analysis by exploring the neighborhood of a chordless cycle $C = v_0, \ldots, v_q, v_0$, $|C| \geqslant 5$ of a planar graph G with a tree 3-spanner T.

For any edge $e = (v_i, v_{i+1})$ in $C - T$, there must be a path in T that is not longer than 3 and not fully contained in C. The different possibilities for such a path are shown in Fig. 8; note that C is chordless.

Before we can derive Lemma 11, which is the first step in the description of the neighborhood of C, we need the following definitions. (Cf. Fig. 9 for an illustration.)

Definition 10. A path P is called *weakly dominated* by $U \subseteq N(P)$ if for any vertex $v \in P$, v or both its neighbors on P are adjacent to U, and if P is not a cycle, both of its endnodes are adjacent to U.

Now let $u \in N(C)$ be a vertex that does not weakly dominate the cycle C, and for an order induced by C, let $D_1, \ldots, D_r$ be the maximal paths of C that are weakly dominated by u. The first vertex of a path D_i according to the ordering is denoted by d_i^h, the last by d_i^t. By definition, there must be a path P_i between any two D_i and D_{i+1} that consists of at least two vertices that are non-adjacent to u. Thus, one endnode of P_i is adjacent to d_i^t and the other adjacent to d_{i+1}^h. $D_1, P_1, \ldots, D_r, P_r$ is called the *u-subdivision* of C. An example is shown in Fig. 9.

For any i, let P_i^1 be the "short" path between d_i^t and d_{i+1}^h, i.e., (d_i^t, P_i, d_{i+1}^h), while P_i^2 is the "long" path, i.e., $(D_{i+1}, P_{i+1}, \ldots, P_{i-1}, D_i)$.

A vertex $w \in N(C)$ is an *independent C-neighbor* of u, if it is adjacent to u in G and if there is an index $1 \leqslant i \leqslant r$ such that the following conditions hold:

(1) There is a path of at most two edges in G that connects w with a vertex of P_i, and

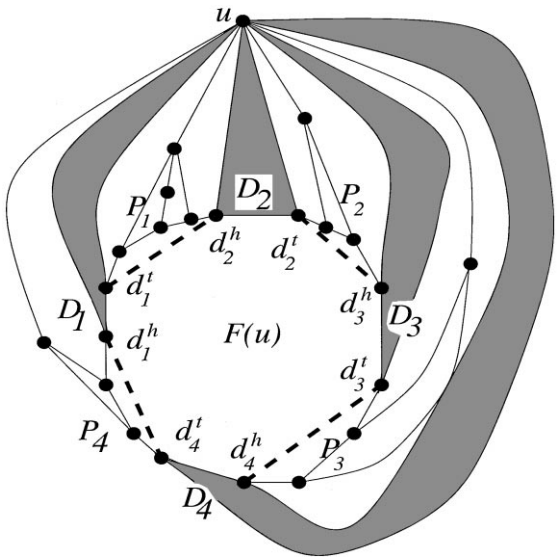

Fig. 9. A vertex u in a semi-dominating tree T_C, its u-partition, and the subface $F(u)$.

(2) there are vertices w_i^h, w_i^t in P_i^1 that are adjacent to w in G and vertices u_i^h, u_i^t in P_i^2 that are adjacent to u in G, such that w_i^h, w_i^t, u_i^h, and u_i^t are pairwise disjoint, and $u_i^h w_i^t$, $w_i^h u_i^t \in E(C)$ holds.

(Note that the path in 1 does not contain vertex u, since u is not adjacent to any vertex in P_i.)

The set of all independent C-neighbors is denoted by $N(C,u)$. A vertex $w \in N(C)$ is a C-successor of u, if there is a path $w_0, w_1, \ldots, w_k$ with $w_0 = u$, $w_k = w$, such that for any $1 \leqslant i \leqslant k$, the vertex w_i is an independent C-neighbor of w_{i-1}. The set of all C-successors is denoted by $D(C,u)$.

Lemma 11. *Let G be a planar graph with a tree 3-spanner T. If C is a chordless cycle in G, $|C| \geqslant 5$, then there is a subtree T_C of T, such that*

(1) *Any vertex $v_i \in C$ is adjacent to T_C, or both its neighbors v_{i-1} and v_{i+1} are adjacent to the same vertex of T_C. In particular, C is weakly dominated by $V(T_C)$.*

(2) *$w \in N(C,u)$ for every edge $(u,w) \in E(T_C)$.*

A tree with these properties is called a semi − dominatingtree *of C.*

Fig. 10 shows an example. Bold lines show the semi-dominating tree.

Proof. First we show that the subgraph $\tilde{T}_C$ of T formed by the edges of $G[N(C)]$ contained in a path $p_T(v_j, v_{j+1})$, $(v_j, v_{j+1}) \in C - T$ satisfies the first condition and is connected.

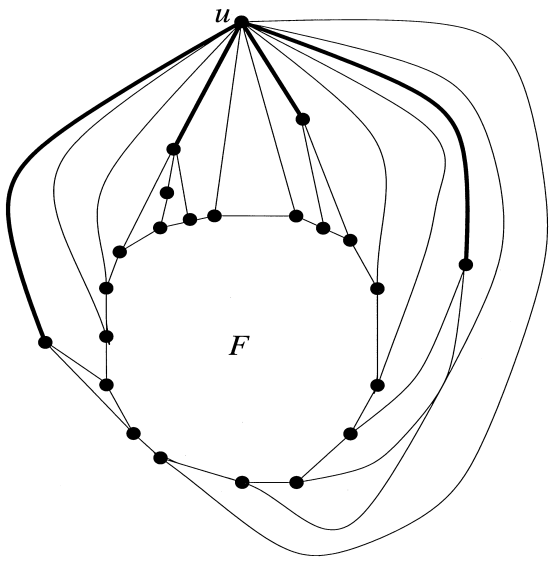

Fig. 10. A long chordless cycle C in G and a semi-dominating tree T_C (bold).

Consider the edges in a path $p_T(v_j, v_{j+1}) - C$, $(v_j, v_{j+1}) \in C - T$. Each of these edges must be contained in $p_T(v_k, v_{k+1})$ for an edge $(v_k, v_{k+1}) \in ((C - T) - (v_j, v_{j+1}))$, since T cannot contain a cycle. In particular, this is true for the edges of $p_T(v_j, v_{j+1}) - C$ incident to C. Because $p_T(v_k, v_{k+1})$ contains at most three edges, in this case (v_k, v_{k+1}) is adjacent to (v_j, v_{j+1}), or between (v_j, v_{j+1}) and (v_k, v_{k+1}) is a path of C with at most two edges. The latter holds iff $p_T(v_j, v_{j+1})$ and $p_T(v_k, v_{k+1})$ follow the course shown in the right part of Fig. 8 and this is the only possibility for two adjacent edges of C to be in T. Thus, condition (1) holds for $\tilde{T}_C$.

Clearly, two vertices of the same path $p_T(v_j, v_{j+1})$ are connected in $\tilde{T}_C$. Furthermore, the above considerations show that the vertices of $p_T(v_j, v_{j+1})$ are connected to the vertices of $p_T(v_k, v_{k+1})$ if (v_k, v_{k+1}) is the next edge of C that is not contained in T. Since C is connected, it follows by induction that $\tilde{T}_C$ is connected.

Now define T_C to be a minimal subtree of $\tilde{T}_C$ that satisfies condition (1), and let $(u, w) \in E(T_C)$. The edge (u, w) is contained in two paths $p_T(v_j, v_{j+1})$ and $p_T(v_k, v_{k+1})$ as stated above. Without loss of generality, let u (w) be adjacent to v_j (v_{j+1}), and v_{k+1} (v_k) (cf. Fig. 8). Since G is planar, u is not adjacent to an inner vertex of the (v_{j+1}, v_k)-path P of C that does not contain v_j. Minimality of T_C with respect to condition 1 implies that there are at least two vertices of P not adjacent to u, i.e., a path P_i of the u-subdivision of C contains all inner vertices of P.

Therefore, condition (2) for an independent C-neighbor is satisfied with $v_j = u_i^h$, $v_{j+1} = w_i^h$, $v_{k+1} = u_i^t$, and $v_k = w_i^t$. If w is adjacent to a vertex of P_i, it follows immediately that $w \in N(C, u)$. Otherwise, the second case of Fig. 8 must apply for w and a neighbor $x \neq u$ of w in T_C that is adjacent to a vertex of P_i, since $V(T_C)$ weakly dominates C and P_i has at least two nodes. □

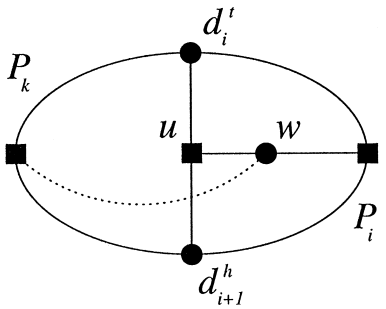

Fig. 11. C-neighbors are unique.

In the first step of our algorithm for determining tree 3-spanners, we compute a semi-dominating tree for every face boundary with more than four edges. Lemma 11 shows us that this is possible if G has a tree 3-spanner. It is shown in Lemma 13 how to find a semi-dominating tree in polynomial time. This step hinges on the following relationship between independent C-neighbors and semi-dominating trees.

Lemma 12. *For any vertex u of a semi-dominating tree, the set of its neighbors in this tree is the set of its independent C-neighbors.*

Proof. By definition, the set of the neighbors of u is a subset of its independent C-neighbors.

It follows from condition (1) in Lemma 11 that there must be a vertex $\tilde{w} \in N(P_i)$ of T_C for any path P_i of the u-subdivision of C. By definition, the neighbor w of u on the $(u,\tilde{w})$-path of T_C is an independent C-neighbor of u. It follows from planarity of G that w is an independent C-neighbor of u with respect to i, i.e., the conditions of the definition of the independent neighbor are fulfilled for i. Furthermore, it is the only independent C-neighbor of u with respect to i (see Fig. 11). □

The following lemma holds for all cycles C of G, chordless or not.

Lemma 13. *Let C be a cycle in a planar graph G, and $u \in N(C)$. If C has a semi-dominating tree T_C containing u, then*

$$T_C = G[D(C,u)] - \{(v,w)\colon w \notin N(C,v)\}.$$

Proof. It follows from Lemma 12 by an easy induction that the vertex sets of the graphs are the same.

Condition (2) of Lemma 11 means that every edge of T_C is an edge of the graph on the right-hand side of the claimed equation. Lemma 12 shows that the edges between u and all its C-neighbors are edges of T_C. It follows by induction that the edges of the right graph form a subset of $E(T_C)$. □

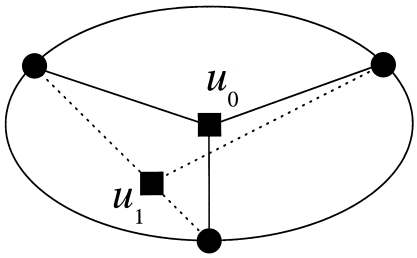

Fig. 12. Uniqueness of semi-dominating trees.

The characterization of Lemma 13 implies the uniqueness of the semi-dominating tree. This is one reason for the "shortness" of paths replacing the inserted edges in a tree 3-spanner:

Lemma 14. *For a given cycle C, $|C| \geqslant 5$, that bounds a face of G, the semi-dominating tree is uniquely determined, if it exists. Moreover, it is minimal with respect to condition* (1) *of Lemma* 11.

Proof. Let T_0 and T_1 be semi-dominating trees of C. From the planarity of G and condition (1) of Lemma 11, it follows that T_0 and T_1 contain a common vertex, since an appropriate contraction leads to the graph shown in Fig. 12. Applying Lemma 13, we get uniqueness.

The second part follows from uniqueness and definition of T_C in Lemma 11. □

Now suppose we have a face F bounded by a chordless cycle C, $|C| \geqslant 5$. If G has a tree 3-spanner then C must have a semi-dominating tree T_C of C which can be found by virtue of Lemma 13. In the next step, we describe how to subdivide F into smaller faces, so that the dynamic programming algorithm from the preceding section can be applied.

For every vertex $u \in V(T_C)$ and a u-subdivision of C, say, $D_1, P_1, \ldots, D_r, P_r$, we insert a set $E'(u)$ as follows. For any i, insert the edge (d_i^t, d_{i+1}^h) – shown by broken lines in Fig. 9. This yields a face $F(u)$ that is dominated by u. This face is triangulated by further new edges. Clearly, this procedure is polynomial.

The desired properties for the edge set $E'(C) = \bigcup_{u \in V(T_C)} E'(u)$ are formulated in the two following lemmas.

Lemma 15. *The graph $(V, E \cup E'(C))$ is planar and emerges from G by subdividing the face F into faces with boundary length at most* 4.

Proof. We show that for any edge $(u, w) \in T_C$, the following properties hold:

- If $D_1, P_1, \ldots, D_r, P_r$ is a u-subdivision, and $D_1^w, P_1^w, \ldots, D_l^w, P_l^w$ is a w-sub-division, then there is an integer $i \in \{1, \ldots, r\}$ such that $D_1^w, P_1^w, \ldots, D_l^w$ are subpaths of $P_i^1 = (d_i^t, P_i, d_{i+1}^h)$, assuming we choose the numbering of $D_1^w, P_1^w, \ldots, D_l^w, P_l^w$ appropriately.

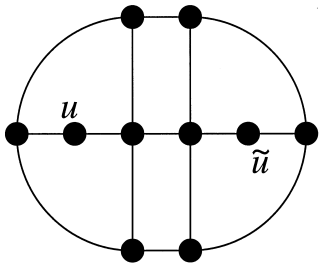

Fig. 13. "Shortness" of paths replacing inserted edges.

- Moreover, the edges of $E'(C)$ connecting the first vertex of D_1^w with the last vertex of D_l^w and the first vertex of D_{i+1}^u, d_{i+1}^h, with the last vertex of D_i^u, d_i^t, respectively, combined with edges of C, bound a face with at most four edges.

By definition, w is an independent C-neighbor of u; suppose it satisfies the conditions for an integer i. T_C is minimal with respect to condition 1 of Lemma 11, as has been shown in Lemma 14, i.e., u cannot be deleted without losing this property. Therefore, w does not weakly dominate $P_i^2 = (D_{i+1}, P_{i+1}, \ldots, P_{i-1}, D_i)$, in other words, this path contains P_l^w if we choose an appropriate numbering. On the other hand, w is not adjacent to any inner vertex of P_i^2, since G is planar. This implies the first property.

Since w is an independent C-neighbor of u with respect to i, i.e., condition (2) of the definition of C-neighbors holds, the second property applies as well.

Now the lemma follows easily by induction (cf. Fig. 9). □

Lemma 16. *For any tree* 3*-spanner* T *of* G *and every edge* $(v,w) \in E'(C)$, *we have* $d_T(v,w) \leqslant 8$.

Proof. Recall that T_C is a subtree of $\tilde{T}_C$ as defined in the proof of Lemma 11 and that every vertex of C lies on one of the paths $p_T(v_j, v_{j+1})$, $(v_j, v_{j+1}) \in C - T$. For $(v,w) \in E'(C)$, there is a vertex $u \in V(T_C)$ with $(v,w) \in E'(u)$. If we can show that v is connected to u by a path with at most four edges, the lemma follows by symmetry.

Since there is a path $p_T(v_j, v_{j+1})$, $(v_j, v_{j+1}) \in C - T$ that contains v, there is a vertex $\tilde{u}$ of $\tilde{T}_C$ with $d_T(v,\tilde{u}) \leqslant 2$ (see Fig. 8).

In the following we only need to consider the case where v is adjacent to $\tilde{u}$: if there is a second edge, we can contract it. Now consider the $(u,\tilde{u})$-path of T. Suppose there are at least three edges in this path. Each of these edges belongs to two paths of the form shown in the center of Fig. 8, i.e., at least six edges of $E(C) \setminus E(T)$ are involved.

Contract the subpaths of C between these edges and delete all vertices of $\tilde{T}_C$ not on the $(u,\tilde{u})$-path of T. Applying planarity of G, we conclude that the resulting graph must contain the graph shown in Fig. 13. It follows from planarity that we cannot have started with a graph where v is adjacent to both u and $\tilde{u}$. Since this is a contradiction, we conclude that the $(u,\tilde{u})$-path of T has at most two edges, and we are done. □

After inserting the edge sets $E'(C)$, we can apply the algorithm of the previous section to compute a tree 3-spanner. This yields a polynomial algorithm:

Theorem 17. *We can decide in polynomial time whether a planar unweighted graph G has a tree 3-spanner.*

Proof. In a first step, we subdivide any face into chordless faces by adding a copy of any of its chords. Next, for every face that is bounded by a chordless cycle C with more than four edges as described above, we insert the edge set $E'(C)$. Note that in the process of introducing new edges, we may create multi-edges. The resulting graph is called $G_{\leqslant 4}$.

As was shown in Lemma 15, the polynomial algorithm for determining the existence of a tree $(3,8)$-spanner of the preceding section can be applied. Since every tree $(3,8)$-spanner of $G_{\leqslant 4}$ is a tree 3-spanner of G, Lemma 16 implies that the algorithm finds a tree 3-spanner of G whenever there is one. □

6. Conclusion

In this paper we have shown that for planar graphs, it is possible to decide the existence of a tree 3-spanner in polynomial time. Our method makes strong use of planarity, yet the resulting algorithm is rather complicated. It has been conjectured that deciding the existence of a tree 3-spanner is an NP-complete problem, and our experience with planar graphs seems to support this belief.

On the other hand, we could prove that deciding the existence of a tree t-spanner is NP-complete, as long as t is part of the input. The complexity for fixed t is unclear, but there may be a polynomial method of deciding the question, possibly using a combination of dynamic programming and an analysis of neighborhood structures, as we did for the case $t=3$. Unfortunately, this analysis appears to become rather tedious even for $t=4$.

Acknowledgements

We would like to thank Dorothea Wagner, Ulrik Brandes, and Dagmar Handke for helpful discussions, and the anonymous referees for comments that helped to improve the presentation of this paper.

References

[1] I. Althöfer, G. Das, D. Dobkin, D. Joseph, J. Soares, On sparse spanners of weighted graphs, Discrete Comput. Geom. 9 (1993) 81–100.

[2] S. Arikati, D.Z. Chen, L.P. Chew, G. Das, M. Smid, D. Zaroliagis, Planar spanners and approximate shortest path queries among obstacles in the plane, in: J. Diaz (Ed.), Algorithms – ESA '96, Springer Lecture Notes in Computer Science, Vol. 1136, Springer, Berlin, 1996, pp. 514–528.
[3] B. Awerbuch, A. Baratz, D. Peleg, Efficient broadcast and light-weight spanners, Manuscript, 1992.
[4] J.A. Bondy, U.S.R. Murty, Graph Theory with Applications, North-Holland, New York, 1976.
[5] U. Brandes, D. Handke, NP-completeness results for minimum planar spanners, Proceedings of the 23th Workshop on Graph–Theoretic Concepts in Computer Science (WG '97), Springer Lecture Notes in Computer Science, Vol. 1335, 1997, pp. 85–99.
[6] L. Cai, Tree spanners: spanning trees that approximate distances, Ph.D. Thesis, University of Toronto, Toronto, Canada, 1992. Available as Technical Report 260/92, Department of Computer Science, University of Toronto.
[7] L. Cai, NP-completeness of minimum spanner problems, Discrete Appl. Math. 48 (1994) 187–194.
[8] L. Cai, D.G. Corneil, Tree spanners, SIAM J. Discrete Math. 8 (1995) 359–387.
[9] L.P. Chew, There is a planar graph almost as good as the complete graph, Proceedings of the Second ACM Symposium on Computational Geometry, SoCG, 1986, pp. 169–177.
[10] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank, A. Schrijver, Combinatorial Optimization, Wiley Interscience, New York, 1998.
[11] D.P. Dobkin, S.J. Friedman, K.J. Supowit, Delaunay graphs are almost as good as complete graphs, Discrete Comput. Geom. 5 (1990) 399–407.
[12] Y. Dodis, S. Khanna, Designing networks with bounded pairwise distance, Proceedings of the 30th ACM Symposium on the Theory of Computing, STOC, 1999, pp. 750–759.
[13] S.P. Fekete, J. Kremer, Tree spanners in planar graphs, Proceedings of the 24th International Annual Workshop on Graph–Theoretic Concepts in Computer Science, WG 98, 1998.
[14] G. Kortsarz, On the hardness of approximating spanners, Approximation Algorithms for Combinatorial Optimization (APPROX 98), Springer Lecture Notes in Computer Science, Vol. 1444, Springer, Berlin, 1998, pp. 135–146.
[15] G. Kortsarz, D. Peleg, Generating sparse 2-spanners, Proceedings of the Third Scandinavian Workshop on Algorithm Theory, SWAT, 1992.
[16] J. Kremer, Baumspanner in planaren Graphen, Diploma Thesis, Mathematisches Institut, Universität zu Köln, 1997.
[17] D. Lichtenstein, Planar formulae and their uses, SIAM J. Comput. 11 (1982) 329–343.
[18] A.L. Liestman, T. Shermer, Grid spanners, Networks 23 (1993) 123–133.
[19] D. Peleg, A.A. Schäffer, Graph spanners, J. Graph Theory 13 (1989) 99–116.
[20] D. Peleg, J.D. Ullman, An optimal synchronizer for the hypercube, Proceedings of the Sixth ACM Symposium on Principles of Distributed Computing, 1987, pp. 77–85.
[21] D. Richards, A.L. Liestman, Degree-constrained pyramid spanners, Parallel Distributed Comput. 25 (1995) 1–6.
[22] J. Soares, Graph spanners: a survey, Congr. Numer. 89 (1992) 225–238.