Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

The Steiner Forest Problem revisited

Elisabeth Gassner [1]

Technische Universität Graz, Institut für Mathematik B, Steyrergasse 30, 8010 Graz, Austria

ARTICLE INFO

Article history:
Received 29 September 2008
Accepted 8 May 2009
Available online 21 May 2009

Keywords:
Steiner Forest
Steiner subgraph
Computational complexity analysis
Series-parallel graph
Outerplanar graph

ABSTRACT

The Steiner Forest Problem (SFP for short) is a natural generalization of the classical Steiner Tree Problem. Instead of only one terminal net there is given a set of terminal nets that have to be connected by choosing edges at minimum cost. Richey and Parker [M.B. Richey, R.G. Parker, On multiple Steiner subgraph problems, Networks 16 (4) (1986) 423–438] posed the question whether SFP is hard on series-parallel graphs. We partially answer this question by showing that SFP is strongly NP-hard on graphs with treewidth 3. On the other hand, a quadratic time algorithm for the special case on outerplanar graphs is suggested. Since series-parallel graphs have treewidth 2 and outerplanar graphs are series-parallel, we almost close the gap between polynomially solvable and hard cases.

1. Introduction

The Steiner Tree Problem belongs to the most well-studied problems in combinatorial optimization. Given an edge-weighted graph $G = (V, E, c)$ and a set of terminals $T \subseteq V$, the task of the Steiner Tree Problem is to find a subset of edges $F \subseteq E$ such that all terminals lie in one connected component in the graph induced by F, i.e., all terminals are connected with each other. A comprehensive survey of applications, complexity and algorithms can be found in [8].

A natural generalization of the Steiner Tree Problem is the Steiner Forest Problem (SFP for short) where the task is to connect several terminal nets, i.e., there is given a set $\mathcal{T} = \{T_i \subseteq V \mid i = 1, \dots, p\}$ of terminal nets and the task is to find a subset $F \subseteq E$ of edges such that all vertices of T_i (for $i = 1, \dots, p$) lie in the same connected component in the graph induced by F. SFP is a generalization of the Steiner Tree Problem and hence it is strongly NP-hard. There exists a 2-approximation algorithm that runs in $\mathcal{O}(m \log m)$ time (Agrawal, Klein and Ravi [1] and later Goemans and Williamson [5] in a more general framework).

Richey and Parker [9] studied several versions of Steiner Subgraph Problems. They call the Steiner Forest Problem *Steiner Subgraph Problem with edge sharing*. The Steiner Subgraph Problem without edge sharing forbids two nets to use a common edge, i.e., the task is to find pairwise disjoint subsets $F_i \subset E$ such that F_i induces a Steiner Tree for T_i. The Steiner Subgraph Problem without edge sharing is NP-hard, even in series-parallel graphs. In the case of the Steiner Subgraph Problem with edge sharing, i.e., the Steiner Forest Problem, the authors prove that a modification of the original problem is NP-hard on series-parallel graphs but leave the complexity status of the original problem as open question.

Organization of this paper. In this paper we give a partial answer to the open question posed in [9]: After a formal definition of SFP in Section 2, we show in Section 3 that SFP is NP-hard on graphs with treewidth three. The NP-hardness of SFP even on graphs with bounded treewidth indicates that SFP is indeed substantially harder than the classical Steiner Tree Problem for which there exists a linear time algorithm for the special case on graphs with bounded treewidth (e.g.,

E-mail address: gassner@opt.math.tugraz.at.

[1] This research has been supported by the Austrian Science Fund (FWF) Proj. P18918-N18.

1570-8667/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.jda.2009.05.002

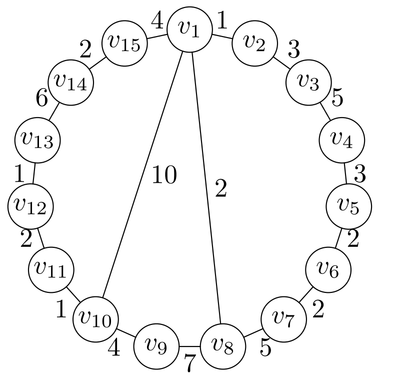


Fig. 1. An outerplanar graph with cycle $C = (v_1, v_2, \ldots, v_{15})$ and chords (v_1, v_8) and (v_1, v_{10}). Moreover, $P = (v_1, v_2, \ldots, v_8)$ is a subdivision path.

see [2,6]). Observe that the class of series-parallel graphs is equivalent to the class of graphs with treewidth at most two [3]. On the other hand, for the special case of outerplanar graphs (that are special series-parallel graphs) a polynomial time algorithm for SFP is suggested in Section 4. Hence, the gap between hard and easy classes is almost closed.

2. Problem formulation

In this section SFP is defined in a formal way: Let a graph $G = (V, E)$, cost coefficients $c(e) \in \mathbb{R}$ for $e \in E$ and a set of terminal nets $\mathcal{T} = \{T_i \subseteq V \mid i = 1, \ldots, p\}$ be given. The task of SFP is to find a subset $F \subseteq E$ of edges such that all vertices of each terminal net lie in the same common connected component in the graph induced by F and

$$\sum_{e \in F} c(e)$$

is minimized.

Note that any instance of SFP can be reduced to one in which either the terminal nets are pairwise disjoint or the terminal nets all have cardinality 2. It is easy to see that such transformations take linear time. The transformed set of terminal nets such that each net contains exactly two vertices is denoted by $K(\mathcal{T})$. A terminal net $\{v_a, v_b\} \in K(\mathcal{T})$ is called pair and v_a is called partner of v_b and vice versa. A vertex v_i that is not a terminal, i.e., there is no pair that contains v_i, is called partner-free vertex.

Notation. Throughout this paper we will use the following notation: Let $G = (V, E)$ be a graph. The number of vertices (edges) of G is denoted by n (m). The degree of a vertex in graph G is denoted by $\deg_G(v)$. For a subset $F \subseteq E$ of edges the graph induced by F is denoted by G_F. Let P be a path. The interior of path P is the subgraph of P that does not contain the two endpoints of P. Moreover, let s, t be two vertices that lie on P. Then we write $s, t \in P$ and the subpath of P connecting s and t is denoted by $SP_P(s, t)$. A subdivision path P of G is a connected subgraph such that there are two vertices $s, t \in P$ with $\deg_P(s) = \deg_P(t) = 1$ (the endvertices of P) and $\deg_G(v) = \deg_P(v) = 2$ holds for all vertices $v \neq s, t$ in P. Consider the graph of Fig. 1: $(v_1, v_2, \ldots, v_8)$ is a subdivision path while $P = (v_{15}, v_1, v_2)$ is not a subdivision path because $\deg_G(v_1) = 4 \neq 2 = \deg_P(v_1)$. Let P be a subdivision path of G. In the absence of ambiguity, we write P instead of $E(P)$ where $E(P)$ denotes the set of edges of P. Hence, if P' and P'' are subdivision paths of G then $P' - P'' = E(P') \setminus E(P'')$. For a subset $X \subseteq E$ of edges $c(X) = \sum_{e \in X} c(e)$ denotes the total cost of edges in X.

Finally, let $I = (G, c, \mathcal{T})$ be an instance of SFP then $\mathcal{F}(I)$ denotes the set of feasible solutions of I. An instance $I' = (G', c', \mathcal{T}')$ is a reduction of I if there exists a subset $Opt(I) \subseteq \mathcal{F}(I)$ that contains at least one optimal solution of I, a surjective function $f : \mathcal{F}(I') \to Opt(I)$ and a constant const. $\in \mathbb{R}$ such that $c(f(F')) = c'(F') + \text{const.}$ holds for every $F' \in \mathcal{F}(I')$. Obviously, every algorithm that solves I' can be used to solve I.

3. NP-hardness

In this section, we show that SFP is strongly NP-hard even for graphs with treewidth 3. The proof is done by a reduction from the Vertex Cover Problem which is known to be strongly NP-hard [4].

Let $\tilde{G} = (\tilde{V}, \tilde{E})$ be an instance of the Vertex Cover Problem with $|\tilde{V}| = n$ and $|\tilde{E}| = m$. We construct an instance of SFP that consists of vertex- and edge-gadgets:

For every $v \in \tilde{V}$ we have a vertex-gadget (V_v, E_v) with

$$V_v = \{s, t, v\},$$
$$E_v = \{(s, v), (v, t)\}.$$

For every $e = (v_i, v_j) \in \tilde{E}$ we construct an edge-gadget (V_e, E_e) with

$$V_e = \{s, t, z_e, z_{e,v_i}, z_{e,v_j}\},$$
$$E_e = \big\{(z_e, z_{e,v_i}), (z_e, z_{e,v_j}), (s, z_{e,v_i}), (s, z_{e,v_j}), (z_{e,v_i}, t), (z_{e,v_j}, t)\big\}.$$

The vertex- and an edge-gadgets are glued together by identifying vertex s and vertex t of every gadget. The edge cost are

$$c(i, j) = \begin{cases} n & (i, j) = (s, v) \text{ for } v \in \tilde{V}, \\ n + 1 & \text{otherwise.} \end{cases}$$

Finally, $\mathcal{T} = \mathcal{T}_E \cup \mathcal{T}_V$ with

$$\mathcal{T}_E = \big\{\{z_e, t\} \mid e \in \tilde{E}\big\},$$
$$\mathcal{T}_V = \big\{\{z_{e,v}, v\} \mid v \in \tilde{V} \text{ and } v \text{ is an endpoint of } e\big\}.$$

Observe that each terminal net contains exactly two vertices. See Fig. 2 for an illustration of the construction (ignore that some edges are bold).

We show that there exists a Vertex Cover of cardinality at most k if and only if there exists a feasible solution F of the constructed instance of SFP with cost at most $3m(n + 1) + n^2 + k$.

Let $C \subseteq \tilde{V}$ be a Vertex Cover of $\tilde{G}$ with $|C| \leqslant k$. For every edge $e \in \tilde{E}$ choose one endpoint of e which is in C, i.e., let $\phi(e) \in C$ such that $\phi(e)$ is an endpoint of e. Now we define a solution for SFP as follows: Choose all edges of the form $(z_e, z_{e,\phi(e)})$. Within the vertex-gadgets choose

$$(v, t) \quad \text{if } v \in C,$$
$$(s, v) \quad \text{if } v \notin C.$$

Within edge gadgets choose

$$(z_{e,v_i}, t) \quad \text{if } v_i \in C,$$
$$(s, z_{e,v_i}) \quad \text{if } v_i \notin C.$$

See Fig. 2 for an illustration of such a solution of SFP.

Let F be the set of edges obtained by the rules just described. Every vertex $v \in \tilde{V}$ is either connected via s or via t to all its partners in $\mathcal{T}_V$. By construction $\phi(e) \in C$ and hence $(z_e, z_{e,\phi(e)}) \in F$ and $(z_{e,\phi(e)}, t) \in F$. All pairs in $\mathcal{T}_E$ are connected. Therefore, F is feasible for SFP. Observe that F contains exactly three edges of every edge-gadget and one edge of every vertex-gadget. Therefore,

$$c(F) = 3m(n + 1) + k(n + 1) + (n - k)n = 3m(n + 1) + n^2 + k.$$

Now assume that F is a feasible solution of SFP with $c(F) \leqslant 3m(n + 1) + n^2 + k$. Due to the feasibility of F there are at least three edges of every edge-gadget and at least one edge of every vertex-gadget in F. Assume that there is a connection between s and t in G_F. Then there exists a vertex-gadget with two edges in F or an edge-gadget with four edges in F. Hence, we would have

$$c(F) \geqslant 3m(n + 1) + n^2 + n + 1 > 3m(n + 1) + n^2 + k$$

which contradicts the assumption $c(F) \leqslant 3m(n + 1) + n^3 + k$. Therefore, for every $v \in \tilde{V}$ either $(s, v) \in F$ or $(v, t) \in F$. Define

$$C = \big\{v \in \tilde{V} \mid (v, t) \in F\big\}.$$

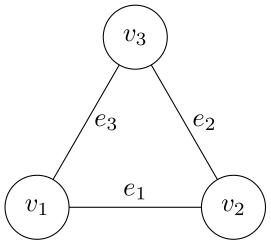
We have to show that C is a Vertex Cover of $\tilde{G}$. Assume that there exists an edge $e = (v_i, v_j) \in \tilde{E}$ such that $(s, v_i), (s, v_j) \in F$. Since there is no connection between s and t in G_F, we have $(s, z_{e,v_i}), (s, z_{e,v_j}) \in F$. But since F is feasible and there is a path from z_e to t in G_F there must be a path from s to t which leads to a contradiction. Observe that G_F is a forest without connection between s and t which implies that there are exactly 3 edges of every edge-gadget in F. Therefore, C is a Vertex Cover that satisfies

$$c(F) = 3m(n + 1) + n\big(n - |C|\big) + (n + 1)|C| = 3m(n + 1) + n^2 + |C| \leqslant 3m(n + 1) + n^2 + k$$

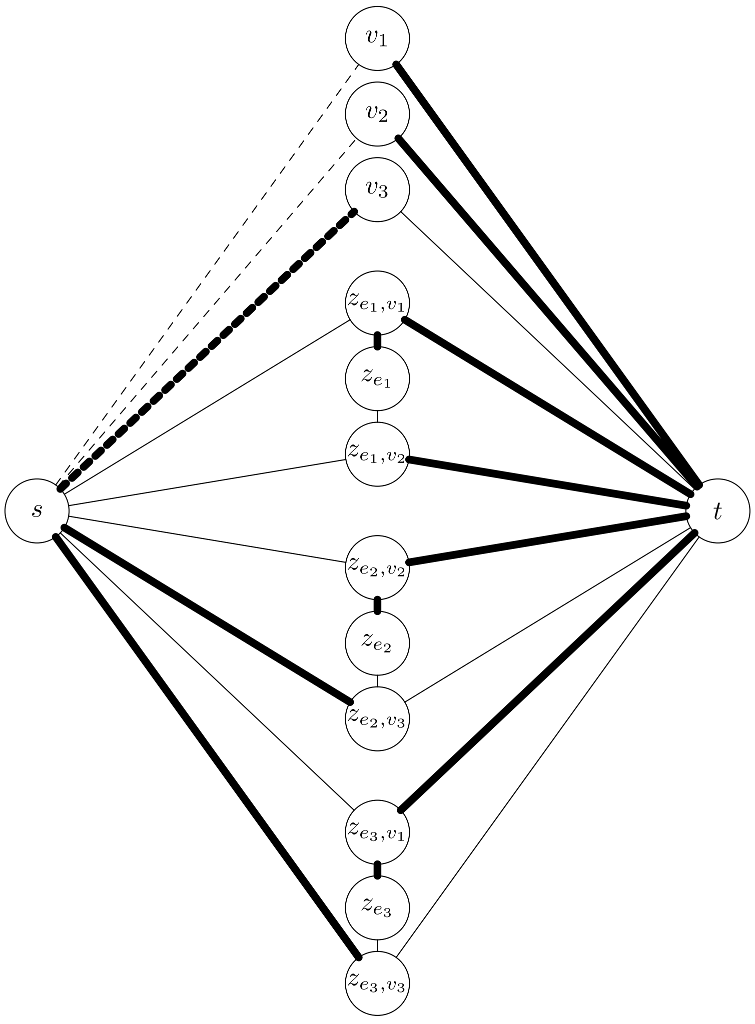
and hence $|C| \leqslant k$.

Finally, we show that the constructed graph has treewidth 3. Consider the following tree decomposition: There is a bag $\{s, t, v\}$ for each $v \in \tilde{V}$, bags $\{s, t, z_e, z_{e,v_i}\}$ and $\{s, t, z_e, z_{e,v_j}\}$ for each $e = (v_i, v_j) \in E$ and finally bag $\{s, t\}$. There is an edge between the bags $\{s, t\}$ and $\{s, t, v\}$ (for every $v \in \tilde{V}$). For each $e = (v_i, v_j) \in E$ there is an edge between $\{s, t\}$ and $\{s, t, z_e, z_{e,v_i}\}$ and between $\{s, t, z_e, z_{e,v_i}\}$ and $\{s, t, z_e, z_{e,v_j}\}$. It is straightforward to verify that this construction yields a tree decomposition of G with width 3. Moreover, the treewidth is not less than 3 because G contains a K_4 as minor.

Theorem 3.1. *The Steiner Forest Problem is strongly* NP*-hard even on graphs with treewidth three.*



(a) Instance $\tilde{G} = (\tilde{V}, \tilde{E})$ of the Vertex Cover problem.



(b) Constructed instance $G = (V, E)$ of SFP: Dashed lines have weight n and solid lines have weight $n+1$. The terminal nets are equal to $\mathcal{T}_E = \{\{z_{e_1}, t\}, \{z_{e_2}, t\}\{z_{e_3}, t\}\}$ and $\mathcal{T}_V = \{\{v_1, z_{e_1,v_1}\}, \{v_1, z_{e_3,v_1}\}, \{v_2, z_{e_1,v_2}\}, \{v_2, z_{e_2,v_2}\}, \{v_3, z_{e_1,v_3}\}, \{v_3, z_{e_3,v_3}\}\}$. Bold lines correspond to a solution of SFP that is associated with vertex cover $C = \{v_1, v_2\}$ of $\tilde{G}$.

Fig. 2. Construction for the $\mathcal{NP}$-hardness proof of SFP.

4. The special case of outerplanar graphs

In this section, we give a polynomial time algorithm for SFP in outerplanar graphs. A graph $G = (V, E)$ is outerplanar if it has a crossing-free embedding in the plane such that all vertices are on the outer face. There are several linear time recognition algorithms for outerplanar graphs (e.g., Mitchell [7] or Wiegers [10]). See Brandstädt et al. [3] for a survey on this graph class.

Let $G + K_1$ denote the graph that is obtained from G by adding a new vertex and joining it with all vertices of G. Wiegers [10] observed that G is outerplanar if and only if $G + K_1$ is planar. Hence, every planarity test can be used to test outerplanarity. Moreover, the planar embedding of $G + K_1$ induces an (not necessarily unique) ordering $(v_1, v_2, \ldots, v_n)$ of the vertices of G on the outer face. We will refer to this ordering throughout. Observe that the graph obtained after adding all missing edges of the form (v_i, v_{i+1}) for $i = 1, \ldots, n-1$ and (v_n, v_1) is again outerplanar (see Fig. 3). In order to make sure that no such auxiliary edge is in an optimal solution of SFP, we have to assign extremely high cost to these edges, e.g., $c(e') = \sum_{e \in E} c(e)$. Hence, from now on we assume that G consists of an outer cycle $C = (v_1, \ldots, v_n)$ and some chords (see Fig. 1 for an illustration). Note that it takes linear time to achieve an instance with this property.

The main idea of the algorithm is to successively reduce the underlying network of the instance until a path remains. An optimal solution of SFP on a path contains all edges with negative cost coefficients as well as all unique paths that

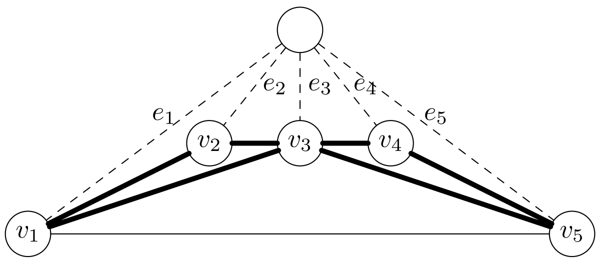


Fig. 3. Let G be the graph induced by the bold edges and K_1 is given by the dashed edges. The graph obtained by G after adding (v_1, v_5) is again outerplanar and consists of the cycle $C = (v_1, v_2, v_3, v_4, v_5)$ and the chords (v_1, v_3) and (v_3, v_5).

connect vertices of the same terminal net. In a reduction step the current instance $I = (G, c, \mathcal{T})$ is transformed to a reduced instance $I' = (G', c', \mathcal{T}')$. The surjective function of the reduction is given by a replacement rule of the following form: Given a feasible solution F' of I' then $f(F')$ is obtained from F' after applying the given replacement rule.

As soon as an optimal solution of the reduced instance is known the replacement rules are applied in reverse order until an optimal solution of the original instance is determined. The key reductions are of the following form: Let $P = (v_1, \ldots, v_k)$ be a subdivision path and $(v_1, v_k) \in E$. The edge (v_1, v_k) is called chord. Then we find a reduced instance that does not contain (v_1, v_k). After deleting all chords, we end up with an instance on a path where SFP is easy to be solved.

Before starting with the main part of our algorithm, the set of terminal nets is modified until it has a particular simple structure:

Step 1: Removing crossings

In the first step terminal nets that cross along C are considered:

Lemma 4.1. *Let $\{v_i, v_j\}, \{v_s, v_t\} \in K(\mathcal{T})$ such that $i < s < j < t$ holds, i.e., $\{v_i, v_j\}$ and $\{v_s, v_t\}$ cross each other. Then these four terminals lie in a common component in G_F for every feasible solution F of* SFP.

Proof. Assume that there exists a feasible solution F such that the four vertices do not lie in a common component in G_F. Then there exists a path P_1 from v_i to v_j and a path P_2 from v_s to v_t in G_F such that P_1 and P_2 are vertex-disjoint. However, the vertices lie in a particular order on the outer cycle which implies the existence of a K_4-minor in G. This contradicts the assumption of outerplanarity of G. □

Lemma 4.1 implies that all terminal nets that contain crossing pairs can be unioned. Consider the following auxiliary graph $H(\mathcal{T})$: $H(\mathcal{T})$ contains a vertex for each pair in $K(\mathcal{T})$. There are two types of edges: There is an edge between $\{v_i, v_j\}$ and $\{v_s, v_t\}$ whenever $\{v_i, v_j\} \cap \{v_s, v_t\} \neq \emptyset$ (intersection edge) and there is an edge between $\{v_i, v_j\}$ and $\{v_s, v_t\}$ whenever these two pairs are crossing (crossing edge).

The connected components of $H(\mathcal{T})$ imply a partition of $V = V_0 \cup V_1 \cup \cdots \cup V_r$ (cf. intersection edges) such that the vertices in V_0 are partner-free (these vertices do no appear in $H(\mathcal{T})$) and vertices in V_i $(i = 1, \ldots, r)$ have to be pairwise connected, i.e., we get a new representation of the set of terminal nets of the form $\{V_i \mid i = 1, \ldots, r\}$. Each terminal net V_i can be transformed to a set of terminal nets each of which has cardinality 2 and there are no crossings.

The transformation described above can be done in $\mathcal{O}(n^2)$ time because $H(\mathcal{T})$ has at most n vertices and n^2 edges.

In the following we assume that we are given an instance $(G, c, \mathcal{T})$ with $K = K(\mathcal{T})$ is an equivalent representation of the set of terminal nets $\mathcal{T}$ and K contains only pairs and has no crossings.

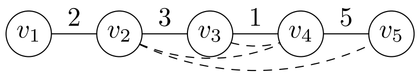
In the next two subsection we show how to obtain an instance without negative cost coefficients and partner-free vertices of degree 2.

Step 2: Removing negative cost coefficients and partner-free vertices of degree 2

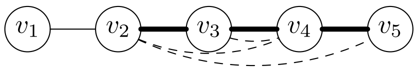
Recall that we allow negative cost coefficients. Obviously, an optimal solution will contain all edges with negative cost. Since the class of outerplanar graphs is closed under edge contractions, negative cost edges may be contracted and included in the final solution.

Given an instance $(G, c, \mathcal{T})$ of SFP with nonnegative cost coefficients and a vertex $v_i \in V$ with $\deg_G(v_i) = 2$ such that v_i has no partner in K. Then there exists an optimal solution that either contains both edges that are incident to v_i or none of them. Therefore, smooth (the inverse of subdivide) every maximal subdivision path P whose internal vertices are partner-free with a single edge e_P with cost $\sum_{e \in P} c(e)$. This operation can be performed in linear time by a depth-first search.

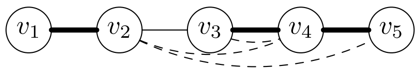
Note that formally both procedures, the deletion of negative cost edges and partner-free vertices of degree 2, are trivial forms of instance reductions.



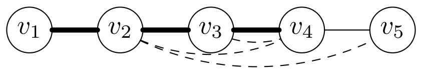
(a) An independent subdivion path $\bar{P} = (v_1, \ldots, v_5)$ (solid lines) and its pairs (dashed lines).



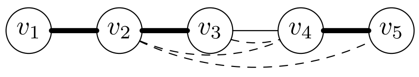
(b) All pairs are connected inside.



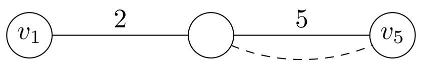
(c) Solution provided that $\{v_2, v_4\}$ is the minimum outside connected pair.



(d) Solution provided that $\{v_2, v_5\}$ is the minimum outside connected pair.



(e) Solution provided that $\{v_3, v_4\}$ is the minimum outside connected pair.



(f) Instance after fully reducing $\bar{P}$.

Fig. 4. Possible connections of pairs on an independent subdivision path.

Step 3: Fully reducing a subdivision path

Assume that we are given an instance $I = (G, c, \mathcal{T})$ without crossings, negative cost coefficients and partner-free vertices of degree 2. Let $P = (v_1, \ldots, v_k)$ be a subdivision path of G. Then we introduce the following definitions:

- $K_P = \{\{v_i, v_j\} \in K \mid v_i, v_j \in P\}$ is the set of pairs on P.
- A subdivision path P is called *independent* if there is no internal vertex v_i of P that has a partner outside of P.
- A subdivision path P is called *fully reduced* if it can be decomposed into several subdivision paths $P = (\bar{P}_1, \ldots, \bar{P}_r)$ such that for $i = 1, \ldots, r$ $\bar{P}_i$ is independent and contains at most two edges.

The task of the following subsection is to find a reduction $I' = (G', c', \mathcal{T}')$ such that the subdivision path P' in G' that is associated with the subdivision path P in G is fully reduced.

Consider subdivision path $P = (v_1, \ldots, v_k)$ of G. Every vertex $v_i \in P$ that has a partner that is not in P is called split-vertex of P. We say that the endpoints of P are split-vertices by definition. These split-vertices split P into several subdivision paths and hence yield a decomposition $P = (\bar{P}_1, \ldots, \bar{P}_r)$. Each such subdivision path $\bar{P}_i$ $(i = 1, \ldots, r)$ is independent because there are no crossings. However, some of these subdivision paths may contain more than two edges. The following describes a reduction I' such that the counterpart of $\bar{P}_i$ in G' has at most two edges. Consider the example illustrated in Fig. 4(a): The solid lines are edges of an independent subdivision path $\bar{P} = (v_1, \ldots, v_5)$ and the dashed lines describe the pairs on $\bar{P}$. We distinguish two types of connections between two partners: There is a connection between them by way of a path fully contained in $\bar{P}$ (inside connection) or the connection needs at least one edge not contained in $\bar{P}$ (outside connection). Either all pairs on $\bar{P}$ are connected inside (see Fig. 4(b)) or there exists at least one pair that is connected outside. Assume that $\{v_2, v_4\}$ is the pair whose subpath $SP_{\bar{P}}(v_2, v_4)$ is shortest among all pairs that are connected outside. This pair is called minimum outside connected pair. Then $\{v_2, v_5\}$ is automatically connected outside and $\{v_3, v_4\}$ is connected inside (due to the minimality property of $\{v_2, v_4\}$). The obtained solution is illustrated in Fig. 4(c). Analogous solutions are obtained provided that $\{v_2, v_5\}$ or $\{v_3, v_4\}$ is a minimum outside connected pair (see Fig. 4(d) and Fig. 4(e)).

Recall that $SP_{\bar{P}}(v_i, v_j)$ denotes the subpath on $\bar{P}$ between v_i and v_j. The set of edges on P that are not in $SP_{\bar{P}}(v_i, v_j)$ are denoted by $\overline{SP}_{\bar{P}}(v_i, v_j)$, i.e.,

$$\overline{SP}_{\bar{P}}(v_i, v_j) = P - SP_{\bar{P}}(v_i, v_j).$$

Lemma 4.2. *Let $(G, c, \mathcal{T})$ be an instance of* SFP *with nonnegative cost coefficients and let $\bar{P} = (v_1, \ldots, v_q)$ be an independent subdivision path of G. Then there exists an optimal solution F^* of* SFP *that satisfies one of the following conditions*:

1. $\bar{P} \cap F^* = \arg\min_{\{v_i, v_j\} \in K_{\bar{P}}} c(F_{\bar{P}}(v_i, v_j)) =: \mathcal{A}_1(\bar{P})$, *or*
2. $\bar{P} \cap F^* = \bigcup_{\{v_i, v_j\} \in K_{\bar{P}}} SP_{\bar{P}}(v_i, v_j) =: \mathcal{A}_2(\bar{P})$, *or*
3. $\bar{P} \cap F^* = \bar{P} := \mathcal{A}_3(\bar{P})$

with

$$F_{\bar{P}}(v_i, v_j) = \left(\bigcup_{\substack{\{v_s, v_t\} \in K_{\bar{P}} \\ i \leqslant s < t \leqslant j}} SP_{\bar{P}}(v_s, v_t) \right) \cup \left(\overline{SP}_{\bar{P}}(v_i, v_j) \right).$$

Proof. Let F^* be an optimal solution of I. F^* may contain all edges of $\bar{P}$ (cf. $\mathcal{A}_3(\bar{P})$). Assume that no optimal solution contains all edges of $\bar{P}$ then there is no connection between v_1 and v_q along $\bar{P}$ in G_F for every optimal F. The pairs outside of $\bar{P}$ are connected in $G_{F^*\setminus\bar{P}}$ (because not all edges of $\bar{P}$ lie in F^*) and hence every $\tilde{F} \subset E$ with

$$c(\tilde{F}) = \min_{\substack{F\setminus\bar{P}=F^*\setminus\bar{P} \\ \text{all pairs in } K_{\bar{P}} \\ \text{are connected in } G_F}} c(F)$$

is an optimal solution. The pairs in $K_{\bar{P}}$ can be connected inside or outside of $\bar{P}$. If they are all connected inside then an optimal solution satisfies $F^* \cap \bar{P} = \mathcal{A}_2(\bar{P})$. Provided that $\{v_i, v_j\} \in K_{\bar{P}}$ is a minimum outside connected pair, an optimal solution satisfied $F^* \cap \bar{P} = F_{\bar{P}}(v_i, v_j)$. Observe that whenever F^* with $F^* \cap \bar{P} = F_{\bar{P}}(v_i, v_j)$ is feasible then $\tilde{F}$ with $\tilde{F} = F^* \cup F_{\bar{P}}(v_s, v_t) \setminus F_{\bar{P}}(v_i, v_j)$ for some $\{v_s, v_t\} \in K_{\bar{P}}$ is also feasible. Hence, an optimal solution will choose the cheapest alternative among the sets $F_{\bar{P}}(v_i, v_j)$ for $\{v_i, v_j\} \in K_{\bar{P}}$ and hence we get $F^* \cap \bar{P} = \mathcal{A}_1(\bar{P})$ which concludes the proof. □

We are now able to find a reduction $I' = (G', c', \mathcal{T}')$ of an independent subdivision path such that the associated subdivision path in G' contains at most two edges:

Reduction *(Fully reducing an independent subdivision path).* Given an independent subdivision path $\bar{P}$ of instance $I = (G, c, \mathcal{T})$ an associated reduced instance $I' = (G', c', \mathcal{T}')$ is defined as follows: Replace $\bar{P} = (v_1, \ldots, v_q)$ by $\bar{P}' = (v_1, x, v_q)$ with

$$c(v_1, x) = c(P) - c\bigl(\mathcal{A}_2(P)\bigr),$$
$$c(x, v_q) = c(P) - c\bigl(\mathcal{A}_1(P)\bigr)$$

and $K' = K \cup \{\{x, v_q\}\} \setminus K_{\bar{P}}$. We use the following *Replacement rule*: $(v_1, x) \to \mathcal{A}_1(\bar{P})$ and $(x, v_q) \to \mathcal{A}_2(\bar{P})$.

Reconsider the example of Fig. 4(a): $F_{\bar{P}}(v_2, v_4) = 8$, $F_{\bar{P}}(v_2, v_5) = 6$, $F_{\bar{P}}(v_3, v_4) = 10$ and hence the cheapest solution among those where at least one pair is connected outside has cost $c(\mathcal{A}_1(\bar{P})) = 6$. If all pairs are connected inside, we get $c(\mathcal{A}_2(\bar{P})) = 9$ and $c(\bar{P}) = 11$. Path $\bar{P}$ is replaced by $\bar{P}' = (v_1, x, v_5)$ where (v_1, x) models $\mathcal{A}_1(\bar{P})$, edge (x, v_5) models $\mathcal{A}_2(\bar{P})$ and if both edges are chosen then the original instance has an optimal solution with $\mathcal{A}_3(\bar{P})$ (see Fig. 4(f)).

Observe that $\mathcal{A}_1(\bar{P}) \cup \mathcal{A}_2(\bar{P}) = \bar{P}$ and hence if $\{\{v_1, x\}, \{x, v_q\}\} \subseteq F' \in \mathcal{F}(I')$ (i.e., both edges are in F') then $f(F')$ contains all edges of $\bar{P}$ and hence $f(F') \cap \bar{P} = \mathcal{A}_3(\bar{P})$. Let

$$Opt(I) = \bigl\{F \in \mathcal{F}(I) \mid F \cap \bar{P} \in \bigl\{\mathcal{A}_1(\bar{P}), \mathcal{A}_2(\bar{P}), \mathcal{A}_3(\bar{P})\bigr\}\bigr\}$$

then the function f defined by the previous replacement rule is defined on $\mathcal{F}(I)$ and maps to $Opt(I)$. It is straightforward to show that f is surjective: Let $F' \in \mathcal{F}(I)$ then either $(v_1, x) \notin F'$ or $(x, v_q) \notin F'$ or both edges are in F'. Observe that there is no feasible solution that excludes both edges because otherwise the pair $\{x, v_q\}$ would not be connected. If $(v_1, x) \notin F'$ or $(x, v_q) \notin F'$ then all pairs in $K' \setminus \{x, v_q\}$ are connected in $G_{F'\setminus P'}$ and hence all pairs in $K \setminus K_{\bar{P}}$ are connected in $G_{f(F')}$. By definition if $(v_1, x) \notin F'$ then $(x, v_q) \in F'$ and hence $f(F') \cap \bar{P} = \mathcal{A}_2(\bar{P})$ and hence all pairs in $K_{\bar{P}}$ are connected in $G_{f(F')}$. If $(x, v_q) \notin F'$ then $f(F') \cap \bar{P} = \mathcal{A}_1(\bar{P})$ which implies the feasibility of $f(F')$. Finally, if $(v_1, x), (x, v_q) \in F'$ then $f(F') \cap \bar{P} = \mathcal{A}_3(\bar{P}) = \bar{P}$. It is easy to see that $f(F')$ is then feasible. Hence, $f(F')$ is a feasible solution with $f(F') \in Opt(I)$. If the sets $\mathcal{A}_k(\bar{P})$ for $k = 1, 2, 3$ are pairwise disjoint, then f is a bijective function. Observe that $\mathcal{A}_1(\bar{P}) \neq \mathcal{A}_k(\bar{P})$ for $k = 2, 3$. In case that $\mathcal{A}_2(\bar{P}) = \mathcal{A}_3(\bar{P})$ the function f is not injective but still surjective. Hence, f is a surjective function. Simple calculations yield

$$c'(F') = c\bigl(f(F)\bigr) - c(P) + c\bigl(\mathcal{A}_1(P)\bigr) + c\bigl(\mathcal{A}_2(P)\bigr).$$

Hence, I' is a reduction of I but the subdivision path associated with $\bar{P}$ contains only two edges.

Summarizing, the procedure of fully reducing a subdivision path works as follows: Given a subdivision path $P = (v_1, \ldots, v_k)$:

- Determine the split-vertices on P. Let $\bar{P}_i$ $(i = 1, \ldots, r)$ be the independent subdivision paths on P between the split-vertices.
- For every $i = 1, \ldots, r$ do:
  - Let $\bar{P}_i = (v_1, \ldots, v_q) = \bar{P}$.
  - Determine $F_{\bar{P}}(v_s, v_t)$ for all $\{v_s, v_t\} \in K_{\bar{P}}$.
  - Let $\mathcal{A}_1(\bar{P}) = \arg\min_{\{v_s, v_t\} \in K_{\bar{P}}} c(F_{\bar{P}}(v_s, v_t))$. Furthermore, determine $\mathcal{A}_2(\bar{P})$ and $c(\bar{P})$.
  - Replace $\bar{P}$ by (v_1, x, v_q) and update K by $K \cup \{\{x, v_q\}\} \setminus K_{\bar{P}}$.
  - Set $c(v_1, x) = c(\bar{P}) - \mathcal{A}_2(\bar{P})$ and $c(x, v_q) = c(\bar{P}) - \mathcal{A}_1(\bar{P})$.
  - *Replacement rule*: $(v_1, x) \to \mathcal{A}_1(\bar{P})$ and $(x, v_q) \to \mathcal{A}_2(\bar{P})$.

The procedure for fully reducing a subdivision path enables us to assume without loss of generality that there are at most two edges between two consecutive split-vertices. The last reduction step is to delete a chord. After deleting all chords, we end up with a path where SFP is easy to solve.

Step 4: Deleting chords

Let us start with a direct consequence of fully reduced subdivision paths:

Lemma 4.3. *Let $G = (V, E)$ be outerplanar with outer cycle $C = (v_1, v_2, \dots, v_n)$ and let $P = (v_1, \dots, v_k)$ be a subdivision path of G. Moreover, let P be fully reduced. Then every feasible solution F of* SFP *satisfies $|P \setminus F| \leqslant 1$, i.e., there is at most one edge of P missing in F.*

Proof. Assume that $(v_i, v_{i+1}), (v_j, v_{j+1}) \in P \setminus F$ with $i < j$. If there is a split-vertex on $SP_P(v_{i+1}, v_j)$, we immediately get a contradiction because there is no connection between the split-vertex and its partner outside of P. If there is no split-vertex on $SP_P(v_{i+1}, v_j)$ then the fact that P is fully reduced implies that v_i and v_{j+1} are split-vertices and $v_{i+1} = v_j$. But since v_j has degree 2 it has at least one partner. On the other hand, v_j is isolated and hence there is no connection from v_j to its partner. □

Lemma 4.3 implies that every feasible solution F of SFP either contains all edges of P or there is at most one missing edge. Now let $P = (v_1, \dots, v_k)$ be a fully reduced subdivision path and let $(v_1, v_k) \in E$. Edge (v_1, v_k) is called chord while the subdivision path does not contain a chord-edge. Then either $(v_1, v_k) \in F$ or $(v_1, v_k) \notin F$. Hence, we get four possible combinations:

- Type 1: One edge of P and (v_1, v_k) are missing in F.
- Type 2: One edge of P is missing in F and $(v_1, v_k) \in F$.
- Type 3: All edges of P are in F but (v_1, v_k) is missing in F.
- Type 4: All edges of P and (v_1, v_k) are in F.

Let F^* be an optimal solution of an instance with nonnegative cost coefficients. Due to the nonnegativity we may assume that there exists an optimal solution that does not contain any cycles, i.e., there exists an optimal solution that is not of Type 4.

Type 2 and Type 3 imply that all vertices of P lie in one connected component of G_{F^*}. Hence, whenever a solution of Type 2 is feasible one can exchange the missing edge of F^* by (v_1, v_k) to get a feasible solution of Type 3 and vice versa. Hence, every optimal solution of Type 2 or 3 will exclude a most expensive edge of $P \cup \{(v_1, v_k)\}$. Let $c_{\max}(P) = \max_{e \in P} c(e) = c(e_{\max})$ where $e_{\max}$ is chosen arbitrarily if there are several most expensive edges.

We consider the following two cases:

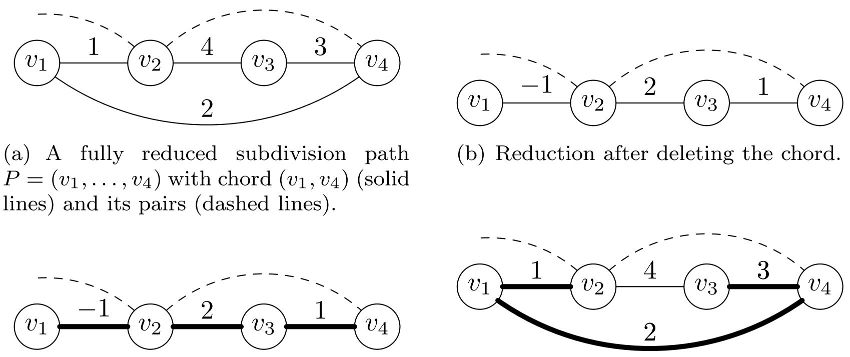
- $c(v_1, v_k) \geqslant c_{\max}(P)$: Then there exists an optimal solution that is not of Type 2 (because Type 3 is not more expensive). Hence, there exists an optimal solution of Type 1 or of Type 3. In both cases the chord (v_1, v_k) does not lie in the considered optimal solution.

  Reduction *(Deleting a chord if $c(v_1, v_k) \geqslant c_{\max}(P)$).* $I' = (G', c, \mathcal{T})$ with $G' = (V, E')$ and $E' = E \setminus \{(v_1, v_k)\}$.

  Since there exists an optimal solution that does not contain $\{(v_1, v_k)\}$, I' is a reduced instance.
- $c(v_1, v_k) < c_{\max}(P)$: Then there exists an optimal solution that is of Type 1 or of Type 2, i.e., one edge of P is missing and the chord is in- or excluded.
  If the chord is included (Type 2) then all vertices of P lie in the same connected component regardless of which edge of P is missing. Hence, $F^* \cap P = P \setminus \{e_{\max}\}$.
  If the chord is excluded (Type 1) then there is no connection between v_1 and v_k in $P \cup \{(v_1, v_k)\}$.
  Our goal is to model these two alternatives (Type 1 and Type 2) without using the chord.

  Reduction *(Deleting a chord if $c(v_1, v_k) < c_{\max}(P)$).* Let $I' = (G', c', \mathcal{T})$ with $G' = (V, E')$ and $E' = E \setminus \{(v_1, v_k)\}$ and $c'(e) = c(e)$ if $e \notin P$ and $c'(e) = c(e) + c(v_1, v_k) - c_{\max}(P)$ otherwise. We use the following *Replacement rule*: Let F' be an optimal solution of I'. If all edges of P are contained in F' then $(e_{\max}) \to (v_1, v_k)$, i.e., $f(F') = F' \cup \{(v_1, v_k)\} \setminus \{e_{\max}\}$ (if not all edges of P are in F' then $f(F') = F'$).

  Consider the example given in Fig. 5(a): $P = (v_1, v_2, v_3, v_4)$ be a fully reduced subdivision path and (v_1, v_4) is a chord. The dashed lines indicate pairs (there may be pairs that connect terminals of inside and of outside the path). An optimal solution will be of Type 1 or of Type 2, i.e., either the chord is excluded and there is no connection between v_1 and v_4 in the graph induced by $P \cup \{(v_1, v_4)\}$ or the chord is included and all vertices of P lie in the same connected



(a) A fully reduced subdivision path $P = (v_1, \dots, v_4)$ with chord (v_1, v_4) (solid lines) and its pairs (dashed lines).

(b) Reduction after deleting the chord.

(c) Solution of the reduced instance

(d) Solution of the original instance.

Fig. 5. Reduction process of deleting a chord.

component. Since $c(v_1, v_4) = 2 < \max_{e \in P} c(e) = 4$, we delete the chord and add $c(v_1, v_4) - \max_{e \in P} c(e) = -2$ to the cost coefficients of edges on P (see Fig. 5(b)). If the optimal solution of the new instance excludes one edge of P then there exists an optimal solution of the original instance with the same set of selected edges (solution of Type 1). If, however, an optimal solution of the new instance chooses all edges of P (Fig. 5(c)) then there is an optimal solution of Type 2 that includes the chord and excludes the most expensive edge of P (see Fig. 5(d)).

We have to show that this is indeed a reduction of the current instance I: We use $Opt(I) = \mathcal{F}(I)$. Let $F' \in \mathcal{F}(I')$. If there is an edge of P missing in F' then $f(F') = F'$ holds and F' is feasible. If all edges of P lie in F' then $f(F') = F' \cup \{(v_1, v_p)\} \setminus \{e_{\max}\}$. $f(F') = F$ is feasible because G_F and $G_{F'}$ have the same connected components. Therefore, $f(F')$ is a feasible solution of the old instance. Function f is bijective. It remains to compute the constant of the reduction: If $|P \setminus F'| = 1$ then

$$c\big(f(F')\big) = c'(F') - (k-1)\big(c(v_1, v_k) - c_{\max}(P)\big)$$

holds. If $|P \setminus F'| = 0$, i.e., $P \subseteq F'$ then

$$\begin{aligned} c\big(f(F')\big) &= c'(F') - k\big[c(v_1, v_k) - c_{\max}(P)\big] - c_{\max}(P) + c(v_1, v_k) \\ &= c'(F') - (k-1)\big(c(v_1, v_k) - c_{\max}(P)\big). \end{aligned}$$

Hence, the constant of the reduction is equal to $-(k-1)(c(v_1, v_k) - c_{\max}(P))$.

Putting all together, we get the following algorithm:

- Preprocessing:
  Remove crossings and determine the set of pairs K. Remove negative cost coefficients and partner-free vertices of degree 2.
- Main step (while G is not a path):
  Select a subdivision path $P = (v_1, \dots, v_k)$ with $(v_1, v_k) \in E$.
  - Remove all negative cost coefficients on P.
  - Remove all partner-free vertices of degree 2 on P.
  - Fully reduce P.
  - Delete the chord (v_1, v_k).
- Determine an optimal solution F^* (G is now a path).
- Backward Phase: Replace F^* according to the replacement rules in reverse order.

Theorem 4.4. *The Steiner Forest Problem can be solved in $\mathcal{O}(n^2)$ time on outerplanar graphs.*

Proof. The correctness of our algorithm follows from the description above. The preprocessing step takes $\mathcal{O}(n^2)$ time. Moreover, there are at most $\mathcal{O}(n)$ chords. For each chord the main step can be done in linear time. Finally, the backward phase takes again linear time. Hence, we get an $\mathcal{O}(n^2)$ time algorithm. □

5. Conclusion

In this paper we discussed the Steiner Forest Problem. We showed that this problem is strongly NP-hard on graphs with treewidth 3 while there exists a quadratic time algorithm on outerplanar graphs.

The complexity status for the special case on series-parallel graphs remains unanswered and hence opens the door for further challenging research.

References

[1] A. Agrawal, P. Klein, R. Ravi, When trees collide: an approximation algorithm for the generalized Steiner problem on networks, SIAM J. Comput. 24 (3) (1995) 440–456.
[2] H.L. Bodlaender, A tourist guide through treewidth, Acta Cybernet. 11 (1–2) (1993) 1–21.
[3] A. Brandstädt, V.B. Le, J.P. Spinrad, Graph Classes: A Survey, SIAM Monographs on Discrete Mathematics and Applications, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.
[4] M.R. Garey, D.S. Johnson, Computers and intractability, in: A Guide to the Theory of NP-Completeness, in: A Series of Books in the Mathematical Sciences, W.H. Freeman and Co., San Francisco, CA, 1979.
[5] M.X. Goemans, D.P. Williamson, A general approximation technique for constrained forest problems, SIAM J. Comput. 24 (2) (1995) 296–317.
[6] E. Korach, N. Solel, Linear time algorithm for minimum weight Steiner tree in graphs with bounded treewidth, Tech. rep., Technion – Israel Institute of Technology, Computer Science Department, Haifa, Israel, report No. 632, 1990.
[7] S.L. Mitchell, Linear algorithms to recognize outerplanar and maximal outerplanar graphs, Inform. Process. Lett. 9 (5) (1979) 229–232.
[8] H.J. Prömel, A. Steger, The Steiner Tree Problem, Advanced Lectures in Mathematics, Friedr. Vieweg & Sohn, Braunschweig, 2002.
[9] M.B. Richey, R.G. Parker, On multiple Steiner subgraph problems, Networks 16 (4) (1986) 423–438.
[10] M. Wiegers, Recognizing outerplanar graphs in linear time, in: Graph-Theoretic Concepts in Computer Science, Bernried, 1986, in: Lecture Notes in Comput. Sci., vol. 246, Springer, Berlin, 1987, pp. 165–176.