Available online at www.sciencedirect.com
SCIENCE DIRECT®

Operations Research Letters 31 (2003) 12–20

Operations Research Letters
www.elsevier.com/locate/dsw

On Steiner trees and minimum spanning trees in hypergraphs

Tobias Polzin[a,*], Siavash Vahdati Daneshmand[b]

[a] *Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123, Saarbrücken, Germany*
[b] *Theoretische Informatik, Universität Mannheim, Mannheim, Germany*

Received 7 March 2002; received in revised form 11 July 2002; accepted 6 August 2002

Abstract

The bottleneck of the state-of-the-art algorithms for geometric Steiner problems is usually the concatenation phase, where the prevailing approach treats the generated full Steiner trees as edges of a hypergraph and uses an *LP*-relaxation of the minimum spanning tree in hypergraph (MSTH) problem. We study this original and some new equivalent relaxations of this problem and clarify their relations to all classical relaxations of the Steiner problem. In an experimental study, an algorithm of ours which is designed for general graphs turns out to be an efficient alternative to the MSTH approach.

Keywords: Steiner problem; Relaxation; Linear programming

1. Introduction

The Steiner problem is the problem of connecting a set of terminals (vertices in a weighted graph or points in some metric space) at minimum cost. This is a classical $\mathcal{NP}$-hard problem with many important applications (see [2]).

For geometric Steiner problems, an approach based on full Steiner trees has been successful [13]. In geometric Steiner problems, a set of points (in the plane) is to be connected at minimum cost according to some geometric distance metric. The resulting interconnection, a Steiner minimal tree (SMT), can be decomposed into its full Steiner trees by splitting its inner terminals (a full Steiner tree (FST) is a tree with no inner terminals, i.e., all terminals have degree 1). The FST approach consists of two phases. In the first phase, the FST generation phase, a set of FSTs is generated that is guaranteed to contain an SMT. In the second phase, the FST concatenation phase, one chooses a subset of the generated FSTs whose concatenation yields an SMT. Although there are point sets that give rise to an exponential number of FSTs in the first phase, usually only a linear number of FSTs are generated, and empirically the bottleneck of this approach has usually been the second phase, where originally methods like backtracking or dynamic programming have been used. A breakthrough occurred as Warme [12] observed that FST concatenation can be reduced to finding a minimum spanning tree in a hypergraph whose vertices are the terminals and whose hyperedges correspond to the generated FSTs. Although the minimum spanning tree in hypergraph (MSTH) problem is $\mathcal{NP}$-hard, a branch-and-cut approach based on the linear relaxation of an integer programming formulation of this problem has been empirically successful.

In this paper, we first compare the mentioned relaxation to some other, new relaxations of the MSTH problem. We show that all these relaxations are

* Corresponding author.
E-mail address: polzin@mpi-sb.mpg.de (T. Polzin).

0167-6377/03/$ - see front matter
PII: S0167-6377(02)00185-2

equivalent (yield the same value), and thereby refute a conjecture in the literature that a (straightforward) directed version of the original relaxation might be stronger. Then, we compare these relaxations with other relaxations which are based directly on formulations of the Steiner problem in graphs. Note that the union of (the edge sets of) the FSTs generated in the first phase is a graph and the FST concatenation problem reduces to solving the classical Steiner problem in this graph. In [6], we already constructed a hierarchy of all classical and some new relaxations of the Steiner problem; here we clarify the place of the MSTH-based relaxations in this hierarchy. Finally, we perform an experimental study, both on the quality of the relaxations and on FST concatenation methods based on them, leading to the result that a program package of ours [7,8,10], which is designed for general networks, is an efficient alternative to the MSTH-based method. Although the approach used by us was known, previous attempts had led to the assumption that it is unlikely to become competitive to the MSTH approach [12].

1.1. Definitions

The Steiner problem in networks can be stated as follows (see [2] for details): Given an (undirected, connected) network $G=(V,E,c)$ (with vertices $V=\{v_1,\ldots,v_n\}$, edges E and edge weights $c_e>0$ for all $e\in E$) and a set R, $\emptyset\neq R\subseteq V$, of *required vertices* (or *terminals*), find a minimum weight tree in G that spans R (an SMT). If we want to stress that v_i is a terminal, we will write z_i instead of v_i.

We also look at a reformulation of this problem using the (bi-)directed version of the graph, because it yields stronger relaxations: Given $G=(V,E,c)$ and R, find a minimum weight arborescence in $\vec{G}=(V,A,c)$ ($A:=\{[v_i,v_j],[v_j,v_i]\mid (v_i,v_j)\in E\}$, c defined accordingly) with a terminal (say z_1) as the root that spans $R_1:=R\setminus\{z_1\}$.

A Steiner tree T for a subset $S\subseteq R$ is called an FST if all terminals in S are leaves of T. Let F be the set of FSTs constructed in the FST generation phase. By identifying each FST $T\in F$ with its set of terminals, we get a hypergraph $H=(R,F)$. For each FST T, let c_T be the sum of its edge weights. Any FST T can be rooted from each of its k leaves, leading to a set of directed FSTs $\{\vec{T}_1,\ldots,\vec{T}_k\}$. We denote the set of directed FSTs generated from F in this way by $\vec{F}$. In the following, we use the term FST both for the tree T and the corresponding hyperedge in H, the meaning should be clear from the context.

A *cut* in $\vec{G}=(V,A,c)$ (or in $G=(V,E,c)$) is defined as a partition $C=\{\bar{W},W\}$ of V ($\emptyset\subset W\subset V$; $V=W\dot\cup\bar{W}$). We use $\delta^-(W)$ to denote the set of arcs $[v_i,v_j]\in A$ with $v_i\in\bar{W}$ and $v_j\in W$. For simplicity, we write $\delta^-(v_i)$ instead of $\delta^-(\{v_i\})$. The sets $\delta^+(W)$ and, for the undirected version, $\delta(W)$ are defined similarly. The corresponding notions for a hypergraph $H=(R,F)$ are defined similarly; here we use Δ instead of δ (for example, $\Delta(S):=\{T\in F\mid T\cap S\neq\emptyset, T\cap\bar{S}\neq\emptyset\}$).

For every integer program P, *LP* denotes the linear relaxation of P. For any (integer or linear) program Q, $v(Q)$ denotes the value of an optimal solution for Q. We compare relaxations using the predicates *equivalent* and (*strictly*) *stronger*: We call a relaxation R_1 stronger than a relaxation R_2 if the optimal value of R_1 is not less than that of R_2 for all instances of the problem. If R_2 is also stronger than R_1, we call them equivalent, otherwise we say that R_1 is strictly stronger than R_2. If neither is stronger than the other, they are *incomparable*.

2. MSTH: Formulations and relaxations

We begin with a formulation of Warme [12] for the MSTH problem

$$\mathrm{P_{FST}}:\qquad \sum_{T\in F} c_T X_T \to \min,$$

$$\sum_{T\in F}(|T|-1)X_T = |R|-1, \tag{1a}$$

$$\sum_{T,\,T\cap S\neq\emptyset}(|T\cap S|-1)X_T \leqslant |S|-1 \qquad (\emptyset\neq S\subset R), \tag{1b}$$

$$X_T\in\{0,1\}\qquad (T\in F). \tag{1c}$$

Lemma 1. *Any feasible solution of* $\mathrm{P_{FST}}$ *describes a spanning tree for the hypergraph* (R,F) *and vice versa.*

Proof. A proof (with slightly different syntax) is given in [12]. □

Using the directed counterpart of F and following the same line as for minimum spanning trees for usual graphs in [5], we get the following integer program:

$$\mathrm{P}_{\mathrm{FS}\vec{\mathrm{T}}}: \quad \sum_{\vec{T}\in\vec{F}} c_{\vec{T}} x_{\vec{T}} \to \min,$$

$$\sum_{\vec{T}\in\vec{F}} (|\vec{T}|-1)x_{\vec{T}} = |R|-1, \tag{2a}$$

$$\sum_{\vec{T},\,\vec{T}\in\Delta^-(z_t)} x_{\vec{T}} = 1 \quad (z_t\in R_1), \tag{2b}$$

$$\sum_{\vec{T},\,\vec{T}\cap S\neq\emptyset} (|\vec{T}\cap S|-1)x_{\vec{T}} \leqslant |S|-1 \quad (\emptyset\neq S\subset R), \tag{2c}$$

$$x_{\vec{T}}\in\{0,1\} \quad (\vec{T}\in\vec{F}). \tag{2d}$$

It is easy to see that $\mathrm{P}_{\mathrm{FS}\vec{\mathrm{T}}}$ is a valid formulation of the MSTH problem.

Lemma 2. *$LP_{\mathrm{FS}\vec{\mathrm{T}}}$ is equivalent to LP_{FST}.*

Proof. The equivalence can be shown by a (proper) choice of the variables representing each FST T and corresponding directed FSTs $\vec{T}_1,\ldots,\vec{T}_k$ such that $X_T = x_{\vec{T}_1}+\cdots+x_{\vec{T}_k}$. The basic ideas are similar to those in the proof of Lemma 6 in [6]. □

Now consider the following cut formulation of the MSTH problem:

$$\mathrm{P}_{\mathrm{FSC}}: \quad \sum_{\vec{T}\in\vec{F}} c_{\vec{T}} x_{\vec{T}} \to \min,$$

$$\sum_{\vec{T}\in\vec{F}} (|\vec{T}|-1)x_{\vec{T}} = |R|-1, \tag{3a}$$

$$\sum_{\vec{T},\,\vec{T}\in\Delta^-(S)} x_{\vec{T}} \geqslant 1 \quad (z_1\notin S,\ S\cap R_1\neq\emptyset), \tag{3b}$$

$$x_{\vec{T}}\in\{0,1\} \quad (\vec{T}\in\vec{F}). \tag{3c}$$

It can be verified (for example by following the proof of the next lemma) that $\mathrm{P}_{\mathrm{FSC}}$ is a valid formulation of the MSTH problem.

Lemma 3. *$LP_{\mathrm{FS}\vec{\mathrm{T}}}$ is equivalent to LP_{FSC}.*

Proof. First observe that for any x feasible for LP_{FSC} summing (3b) for all $z_t\in R_1$ we have

$$|R|-1 \leqslant \sum_{z_t\in R_1}\sum_{\vec{T}\in\Delta^-(z_t)} x_{\vec{T}} \leqslant \sum_{\vec{T}} (|\mathrm{leaves}(\vec{T})|)x_{\vec{T}} = \sum_{\vec{T}} (|\vec{T}|-1)x_{\vec{T}}. \tag{4}$$

Together with (3a) this means that x satisfies (2b). It follows that $\sum_{\vec{T}\in\Delta^-(z_1)} x_{\vec{T}} = 0$.

Now consider any x that is feasible for LP_{FST} or LP_{FSC}; we will show that in either case x is feasible for both. For any partition $S\dot\cup\bar{S}=R$, we have

$$\sum_{\vec{T},\,\vec{T}\cap S\neq\emptyset} (|\vec{T}\cap S|-1)x_{\vec{T}}$$

$$= \sum_{\vec{T},\,\vec{T}\cap S\neq\emptyset} (|\mathrm{leaves}(\vec{T})\cap S|+|\mathrm{root}(\vec{T})\cap S|-1)x_{\vec{T}} \tag{5}$$

$$= \sum_{\vec{T},\vec{T}\cap S\neq\emptyset} |\mathrm{leaves}(\vec{T})\cap S|x_{\vec{T}} - \sum_{\vec{T},\,\vec{T}\cap S\neq\emptyset,\mathrm{root}(\vec{T})\notin S} x_{\vec{T}} \tag{6}$$

$$= \sum_{z_t\in S}\sum_{\vec{T}\in\Delta^-(z_t)} x_{\vec{T}} - \sum_{\vec{T},\,\vec{T}\in\Delta^-(S)} x_{\vec{T}} \tag{7}$$

$$= |S\cap R_1| - \sum_{\vec{T},\,\vec{T}\in\Delta^-(S)} x_{\vec{T}}. \tag{8}$$

Now there are two cases:

(I) $z_1\in\bar{S}$:

$$\sum_{\vec{T},\ \vec{T}\cap S\neq\emptyset} (|\vec{T}\cap S|-1)x_{\vec{T}} = |S| - \sum_{\vec{T},\ \vec{T}\in\Delta^-(S)} x_{\vec{T}}. \tag{9}$$

This means that x satisfies (2c) if and only if it satisfies (3b).

(II) $z_1 \in S$:

$$\sum_{\vec{T},\ \vec{T}\cap S\neq\emptyset} (|\vec{T}\cap S|-1)x_{\vec{T}}$$

$$=|S|-1-\sum_{\vec{T},\ \vec{T}\in\Delta^-(S)} x_{\vec{T}}. \qquad (10)$$

So x satisfies (2c), because it is nonnegative. □

Note that we have actually proved a slightly stronger result: The sets of feasible solutions (and corresponding polyhedra) are identical for both relaxations. With respect to optimal solutions, our assumption that the edge costs are positive leads directly to the observation that $\sum_{\vec{T}\in\Delta^-(z_1)} x_{\vec{T}} = 0$. A more detailed analysis (similar to our proofs of Lemmas 8 and 9 in [6] for the dicut relaxation in graphs) leads to the observation that for any optimal solution for LP_{FSC} without (3a) and for every $z_t \in R_1$, it holds: $\sum_{\vec{T}\in\Delta^-(z_t)} x_{\vec{T}} = 1$. So dropping the constraints (3a) does not change the optimal solution value of LP_{FSC}.

3. Relation to the relaxations of the Steiner problem in graphs

The directed cut formulation of the Steiner problem was stated for the first time in [15] (the undirected version was already introduced by Aneja [1]):

$$\mathrm{P_C}: \quad \sum_{a\in A} c_a y_a \to \min,$$

$$\sum_{a\in\delta^-(S)} y_a \geqslant 1 \quad (z_1 \notin S,\ S\cap R_1 \neq \emptyset), \qquad (11a)$$

$$y_a \in \{0,1\} \quad (a\in A). \qquad (11b)$$

Lemma 4. *LP_{FSC} is (strictly) stronger than LP_{C}.*

Proof. Let x be an optimal solution of LP_{FSC}. For each arc $a\in A$ that is a part of directed FSTs $\vec{T}_1,\dots,\vec{T}_l$, let $y_a := x_{\vec{T}_1} + \cdots + x_{\vec{T}_l}$. It is easy to verify that y is feasible for LP_{C} and yields the same value as $v(LP_{\mathrm{FSC}})$. The following example shows that $v(LP_{\mathrm{FSC}})$ can indeed be larger than $v(LP_{\mathrm{C}})$. □

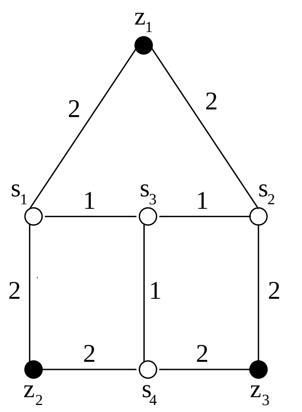

Fig. 1. Example with $v(LP_{\mathrm{C}}) < v(LP_{\mathrm{FSC}}) = v(\mathrm{P_{FSC}})$.

Example 1. The network in Fig. 1 with z_1 as the root (filled circles represent terminals) and (directed) FSTs $(z_1 \to s_1 \to z_2)$, $(z_1 \to s_2 \to z_3)$, $(z_1 \to s_1 \to s_3 \to s_4 \to z_2)$ and $(z_1 \to s_2 \to s_3 \to s_4 \to z_3)$ gives an example for $v(LP_{\mathrm{C}}) < v(LP_{\mathrm{FSC}})$: $v(\mathrm{P_{FSC}}) = v(LP_{\mathrm{FSC}}) = 8$, $v(LP_{\mathrm{C}}) = 7.5$.

Example 2. By turning s_3 to a terminal we get an example with $v(LP_{\mathrm{FSC}}) < v(\mathrm{P_{FSC}})$: $v(\mathrm{P_{FSC}})$ is still 8; but $v(LP_{\mathrm{FSC}})$ is now 7.5 (by setting to 0.5 the x-values for the FSTs (z_1,s_1,s_3,z_2), (z_1,s_2,s_3,z_3) and (s_3,s_4,z_2,z_3)).

Note also that this is an example where the choice of FSTs in the first phase influence the value of LP_{FSC}: If only the FSTs (z_1,s_1,s_2), (z_1,s_1,s_3), (z_1,s_2,z_3), (z_1,s_2,s_3) and (s_3,s_4,z_2,z_3) are generated in the first phase (an SMT can be constructed by concatenation of the second and last FST), then $v(LP_{\mathrm{FST}}) = v(\mathrm{P_{FST}}) = 8$. Note also that $v(LP_{\mathrm{C}}) = 7.5$ in both cases.

The relaxation LP_{C} can be strengthened by additional groups of constraints like the following one from [3], which we call flow-balance constraints:

$$\sum_{a\in\delta^-(v_i)} y_a \leqslant \sum_{a\in\delta^+(v_i)} y_a \quad (v_i \in V\setminus R). \qquad (11c)$$

In [6], we prove that these constraints indeed lead to a strictly stronger relaxation, which we call $LP_{\mathrm{C+FB}}$.

Lemma 5. *LP_{FSC} is (strictly) stronger than $LP_{\mathrm{C+FB}}$.*

Proof. Consider a vertex $v_i \in V \setminus R$. Any directed FST $\vec{T}$ containing an arc $a \in \delta^-(v_i)$ includes also at least one arc $a \in \delta^+(v_i)$, so the same construction as in the proof of Lemma 4 leads to a y which also satisfies the constraints (11c). The following example shows that $v(LP_{\mathrm{FSC}})$ can indeed be larger than $v(LP_{\mathrm{C+FB}})$. □

Example 3. By adding a terminal z_4 to the network in Fig. 1 and connecting it to s_3 with an edge of cost 1 we get an example for $v(LP_{\mathrm{C+FB}}) < v(LP_{\mathrm{FSC}})$ if z_1 is chosen as the root: $v(\mathrm{P}_{\mathrm{FSC}}) = v(LP_{\mathrm{FSC}}) = 9$, $v(LP_{\mathrm{C}}) = 8.5$.

Since $LP_{\mathrm{C+FB}}$ is (strictly) stronger than all but two relaxations of the Steiner problem in [6], LP_{FSC} is strictly stronger than them too. The two remaining ones are the linear relaxations of the two-terminals formulation P_{2T} and the common-flow formulation P_{F^2} (see [6] for definitions).

Lemma 6. *LP_{FSC} is incomparable to LP_{2T} or LP_{F^2}.*

Proof. The lemma follows from the following two examples. □

Example 4. The network in Fig. 1 with s_3 set to terminal gives an example with $v(LP_{2T}) = v(LP_{F^2}) > v(LP_{\mathrm{FSC}})$: As described in Example 2, $v(LP_{\mathrm{FSC}}) = 7.5$, but $v(LP_{2T}) = v(LP_{F^2}) = 8$.

Example 5. The network in Fig. 1 with s_3 set to terminal and used as the root gives an example for $v(LP_{2T}) = v(LP_{F^2}) < v(LP_{\mathrm{FSC}})$: $v(LP_{2T}) = v(LP_{F^2}) = 7.5$, but $v(\mathrm{P}_{\mathrm{FSC}}) = v(LP_{\mathrm{FSC}}) = 8$ if we assume that the following FSTs are generated in the first phase: (s_3, s_1, z_1, z_2), (s_3, s_2, z_1, z_3), (s_3, s_4, z_2) and (s_3, s_4, z_3). Note again the influence of the first phase: If the FST (s_3, s_4, z_2, z_3) was also generated, $v(LP_{\mathrm{FSC}})$ would be 7.5, too.

However, there are also examples such that $v(LP_{\mathrm{FSC}}) = v(\mathrm{P}_{\mathrm{FSC}}) > v(LP_{F^2})$ even if all possible FSTs are generated in the first phase.

4. Experimental study

In this section, we compare the empirical behaviour of the MSTH-based relaxation LP_{FST} and an exact algorithm based on it with the classical directed cut relaxation LP_{C} and an exact algorithm which uses this relaxation. For the first approach, we use GeoSteiner 3.1 [14], a software package developed by Warme, Winter and Zachariasen for solving Euclidean and rectilinear Steiner problems and the MSTH problem. GeoSteiner is by far the most efficient code for these problems. Note also that the results of the version 3.1 of GeoSteiner are significantly better than the already published results [12,13] of former versions. For the second approach, we use a software package developed by us [8] (here called simply STEINER), which is designed for treating the Steiner problem in general networks. As test data, we use the "geometric graph" instances from the library SteinLib [4]. These graphs are produced by applying the FST generation phase of GeoSteiner to some point sets which are either randomly generated (ES-instances from OR-Library) or originate from some application (TSP-instances from TSP-LIB). For the comparisons, we have excluded those instances that could not be solved by GeoSteiner in one day; results of our program on such instances are given separately in Table 2. The FST generation phase (followed by a pruning phase to reduce the number of FSTs) delivers both a hypergraph $H = (R, F)$ corresponding to the generated FSTs (which is the input for the MSTH-based concatenation phase of GeoSteiner) and a graph $G = (V, E)$ corresponding to the union of their edge sets (which is the input for our network SMT-algorithm).

All tests were performed on a PC with an AMD Athlon XP 1800+ (1.53 GHz) processor and 1 GB of main memory, using the operating system Linux 2.4.9. We used the gcc 2.96 compiler and CPLEX 7.0 as LP-solver.

In Table 1, we compare the average gaps to integer optimum and computation times for the relaxations

Table 1
Comparison of LP_{FST} and LP_{C}

Instance group	LP_{FST}		LP_{C}	
	Gap (%)	Time (s)	Gap (%)	Time (s)
ES1000FST	0.0078	99.2	0.0079	80.1
TSPFST	0.009803	129.6	0.009806	28.6

Table 2
Instances not solved by GeoSteiner in 1 day

Instance	Size			Optimum	STEINER	
	$\|R\|$	$\|V\|$	$\|E\|$		Time (s)	Nodes
es10000	10,000	27,019	39,407	716,174,280	758	1
fl1400	1400	2694	4546	17,980,523	118	1
fl3795	3795	4859	6539	25,529,856	139	1
fnl4461	4461	17,127	27,352	182,361	6148	1
pcb3038	3038	5829	7552	131,895	2.4	1
pla7397	7397	8790	9815	22,481,625	0.1	1

Table 3
Comparison of GeoSteiner (second phase) and STEINER on ES1000FST-instances

Instance	Size				Optimum	GeoSteiner		STEINER	
	$\|R\|$	$\|F\|$	$\|V\|$	$\|E\|$		Time (s)	Nodes	Time (s)	Nodes
es1000fst01	1000	2052	2865	4267	230,535,806	12.78	3	11.55	1
es1000fst02	1000	1943	2629	3793	227,886,471	9.38	1	7.79	1
es1000fst03	1000	2004	2762	4047	227,807,756	115.00	1	11.29	1
es1000fst04	1000	2024	2778	4083	230,200,846	8.41	1	12.52	1
es1000fst05	1000	1976	2676	3894	228,330,602	64.23	1	8.50	1
es1000fst06	1000	2033	2816	4164	231,028,456	409.78	10	16.13	1
es1000fst07	1000	1897	2604	3756	230,945,623	87.67	1	4.80	1
es1000fst08	1000	2047	2836	4210	230,639,115	111.38	1	12.32	1
es1000fst09	1000	2091	2846	4187	227,745,838	18.03	3	12.72	1
es1000fst10	1000	1894	2546	3620	229,267,101	112.97	5	4.76	1
es1000fst11	1000	2026	2763	4038	231,605,619	19.58	3	8.13	1
es1000fst12	1000	2136	2992	4500	230,904,712	484.46	2	16.47	1
es1000fst13	1000	1886	2532	3615	228,031,092	3.07	1	4.62	1
es1000fst14	1000	2049	2840	4200	234,318,491	791.82	13	14.92	1
es1000fst15	1000	2032	2735	4001	229,965,775	10.82	1	7.59	1
Averages:						150.6		10.3	

LP_{FST} and LP_{C}. We used GeoSteiner for LP_{FST} by taking $v(LP_{\mathrm{FST}})$ as the value of the last linear program before any branching was performed. Studying the data (detailed results on single instances can be found in [9]), one observes:

- Both relaxations yield almost always the same value. Only on a couple of instances, LP_{FST} is tighter than LP_{C} by a relatively small margin.
- Both relaxations are fairly tight on the considered instances. The average gap to integer optimum is in both cases less than 0.01%.
- The average running times for computing $v(LP_{\mathrm{C}})$ have been smaller, but this does not say much about which method is faster on a specific instance.

In Tables 3 and 4, we compare the running times of GeoSteiner and STEINER for the exact solution of the test instances. We also give the number of nodes in the branch-and-cut or branch-and-bound tree. Studying the tables, one observes:

- STEINER is in average and in most cases faster than GeoSteiner. There are a couple of instances where STEINER needs some seconds more than GeoSteiner. On the other hand, STEINER is faster by some orders of magnitude than GeoSteiner on

Table 4
Comparison of GeoSteiner (second phase) and STEINER on TSPFST-instances

Instance	Size				Optimum	GeoSteiner		STEINER	
	$\|R\|$	$\|F\|$	$\|V\|$	$\|E\|$		Time (s)	Nodes	Time (s)	Nodes
a280	280	311	314	328	2502	0.03	1	0.01	1
att48	48	101	139	202	30,236	0.02	1	0.30	1
att532	532	1065	1468	2152	84,009	195.72	1	3.74	1
berlin52	52	78	89	104	6760	0.01	1	0.01	1
bier127	127	213	258	357	104,284	0.08	1	0.02	1
d1291	1291	1361	1365	1456	481,421	0.40	1	0.01	1
d1655	1655	1879	1906	2083	584,948	3.30	1	0.04	1
d198	198	232	232	256	129,175	0.03	1	0.01	1
d2103	2103	2196	2206	2272	769,797	1.07	1	0.02	1
d493	493	966	1055	1473	320,137	261.32	1	0.58	1
d657	657	1176	1416	1978	471,589	312.40	3	1.51	1
dsj1000	1000	1884	2562	3655	17,564,659	15.71	1	1.47	1
eil101	101	295	330	538	605	0.30	1	0.83	1
eil51	51	138	181	289	409	0.11	1	1.63	1
eil76	76	196	237	378	513	0.09	2	0.58	1
fl1577	1577	2839	2413	3412	19,825,626	34.64	1	0.99	1
fl417	417	872	732	1084	10,883,190	7.01	13	0.85	1
gil262	262	447	537	723	2306	0.53	1	0.06	1
kroA100	100	165	197	250	20,401	0.01	1	0.02	1
kroA150	150	296	389	562	25,700	0.18	1	0.69	1
kroA200	200	389	500	714	28,652	0.19	1	0.24	1
kroB100	100	180	230	313	21,211	0.06	1	0.05	1
kroB150	150	297	420	619	25,217	0.13	1	0.42	1
kroB200	200	369	480	670	28,803	0.22	1	0.62	1
kroC100	100	190	244	337	20,492	0.05	1	0.10	1
kroD100	100	166	216	288	20,437	0.03	1	0.02	1
kroE100	100	176	226	306	21,245	0.04	1	0.13	1
lin105	105	190	216	323	13,429	0.29	1	0.12	1
lin318	318	589	738	1279	39,335	15.26	4	0.43	1
linhp318	318	589	678	1030	39,335	15.19	4	0.40	1
nrw1379	1379	3590	5096	8105	56,207	16410.99	32	150.14	1
p654	654	760	777	867	314,925	0.15	1	0.01	1
pcb1173	1173	1708	1912	2223	53,301	2.77	1	0.11	1
pcb442	442	494	503	531	47,675	0.07	1	0.01	1
pr1002	1002	1392	1474	1717	243,176	0.70	1	0.05	1
pr107	107	110	111	110	34,850	0.01	1	0.01	1
pr124	124	147	154	165	52,759	0.01	1	0.01	1
pr136	136	227	196	250	86,811	0.07	1	0.01	1
pr144	144	184	221	285	52,925	0.04	1	0.01	1
pr152	152	242	308	431	64,323	0.16	1	0.06	1
pr226	226	248	255	269	70,700	0.06	1	0.01	1
pr2392	2392	3311	3398	3966	358,989	6.35	1	0.06	1
pr264	264	280	280	287	41,400	0.03	1	0.01	1
pr299	299	416	420	500	44,671	0.11	1	0.01	1
pr439	439	551	572	662	97,400	0.55	1	0.01	1
pr76	76	138	168	247	95,908	0.04	1	0.02	1
rat195	195	435	560	870	2386	0.09	1	0.95	1
rat575	575	1482	1986	3176	6808	1.75	1	19.26	1
rat783	783	1784	2397	3715	8883	5.90	1	17.57	1
rat99	99	200	269	399	1225	0.06	1	0.11	1
rd100	100	168	203	257	764,269,099	0.02	1	0.01	1

Table 4 (*continued*)

Instance	Size				Optimum	GeoSteiner		STEINER	
	$\lvert R\rvert$	$\lvert F\rvert$	$\lvert V\rvert$	$\lvert E\rvert$		Time (s)	Nodes	Time (s)	Nodes
rd400	400	747	1001	1419	1,490,972,006	0.62	1	1.49	1
rl11849	11,849	13,780	13,963	15,315	8,779,590	1100.43	9	0.64	1
rl1304	1304	1514	1562	1694	236,649	0.51	1	0.04	1
rl1323	1323	1545	1598	1750	253,620	1.01	1	0.01	1
rl1889	1889	2247	2382	2674	295,208	1.56	1	0.22	1
rl5915	5915	6540	6569	6980	533,226	76.48	10	0.09	1
rl5934	5934	6739	6827	7365	529,890	41.29	3	0.09	1
st70	70	107	133	169	626	0.01	1	0.01	1
ts225	225	224	225	224	1120	0.01	1	0.01	1
tsp225	225	240	242	252	356,850	0.02	1	0.01	1
u1060	1060	1708	1835	2429	21,265,372	61.70	65	1.09	1
u1432	1432	1431	1432	1431	1465	0.02	1	0.01	1
u159	159	180	184	186	390	0.01	1	0.01	1
u1817	1817	1830	1831	1846	5,513,053	0.08	1	0.01	1
u2152	2152	2166	2167	2184	6,253,305	0.23	1	0.02	1
u2319	2319	2318	2319	2318	2322	0.03	1	0.01	1
u574	574	877	990	1258	3,509,275	0.59	1	0.17	1
u724	724	1093	1180	1537	4,069,628	1.42	1	0.22	1
vm1084	1084	1474	1679	2058	2,248,390	1.71	1	0.37	1
vm1748	1748	2488	2856	3641	3,194,670	7.07	3	1.98	1
Averages:						261.8		3.0	

the more time-consuming instances. For example, to solve the instance es10000fst from SteinLib (the largest instance of this type ever solved), GeoSteiner (actually, an unreleased version of it) needs months of cpu-time, whereas STEINER needs less than 15 min. Also, all previously unsolved geometric instances in SteinLib could be solved by STEINER in relatively small time (see Table 2).

- GeoSteiner uses branching on 18 of the 86 tested instances, whereas STEINER has used branching on none of the considered instances.
- GeoSteiner needs always more time for exact solution than for the computation of $v(LP_{\mathrm{FST}})$, since it begins the concatenation phase mainly with the computation of the latter value. This is not the case for STEINER, since it begins with applying reduction methods and uses the relaxation LP_{C} (or some extended variants of it) as explicit linear programs (if at all) only in the advanced stages of the solution process (see [7,8,10,11]).

5. Concluding remarks

The main subject of this paper has been studying, both theoretically and empirically, different approaches for the second phase of the FST method for Steiner problems. The experimental results show the potential of our program STEINER for this phase. But it should not be conceived as a competitor for GeoSteiner for solving geometric Steiner problems from scratch; GeoSeiner remains the most efficient package for these problems. Considering the second phase, a combination of the two approaches could be even more successful. The program STEINER does not use the knowledge of individual FSTs. As an algorithm for the concatenation phase, STEINER could profit from the fact that for each FST, either all or no edges can be chosen. This can be helpful for example for the computation of lower bounds, where variables could correspond to (directed) FSTs instead of single arcs, while keeping the fast method for constraint generation based on minimum cuts in usual graphs.

Also, the reduction methods could benefit from this information: Once it is established that an edge can be excluded, all FSTs which contain that edge can be discarded. On the other hand, GeoSteiner could profit from different components of STEINER, especially its sophisticated reduction techniques.

References

[1] Y.P. Aneja, An integer linear programming approach to the Steiner problem in graphs, Networks 10 (1980) 167–178.

[2] F.K. Hwang, D.S. Richards, P. Winter, The Steiner Tree Problem, in: Annals of Discrete Mathematics, Vol. 53, North-Holland, Amsterdam, 1992.

[3] T. Koch, A. Martin, Solving Steiner tree problems in graphs to optimality, Networks 32 (1998) 207–232.

[4] T. Koch, A. Martin, SteinLib, http://elib.zib.de/steinlib/steinlib.php, 2001.

[5] T.L. Magnanti, L.A. Wolsey, Optimal trees, in: M.O. Ball, et al. (Eds.), Handbooks in Operations Research and Management Science, Vol. 7, Elsevier Science, Amsterdam, 1995 (Chapter 9).

[6] T. Polzin, S. Vahdati Daneshmand, A comparison of Steiner tree relaxations, Discrete Appl. Math. 112 (2001) 241–261.

[7] T. Polzin, S. Vahdati Daneshmand, Extending reduction techniques for the Steiner tree problem: a combination of alternative- and bound-based approaches, Research Report MPI-I-2001-1-007, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2001.

[8] T. Polzin, S. Vahdati Daneshmand, Improved algorithms for the Steiner problem in networks, Discrete Appl. Math. 112 (2001) 263–300.

[9] T. Polzin, S. Vahdati Daneshmand, On Steiner trees and minimum spanning trees in hypergraphs, Research Report MPI-I-2001-1-005, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2001.

[10] T. Polzin, S. Vahdati Daneshmand, Partitioning techniques for the Steiner problem, Research Report MPI-I-2001-1-006, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2001.

[11] T. Polzin, S. Vahdati Daneshmand, Using (sub)graphs of small width for solving the Steiner problem, Research Report MPI-I-2002-1-001, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2002.

[12] D.M. Warme, Spanning Trees in Hypergraphs with Applications to Steiner Trees, Ph.D. Thesis, University of Virginia, 1998.

[13] D.M. Warme, P. Winter, M. Zachariasen, Exact algorithms for plane Steiner tree problems: a computational study, in: D-Z. Du, J.M. Smith, J.H. Rubinstein (Eds.), Advances in Steiner Trees, Kluwer Academic Publishers, Dordrecht, 2000, pp. 81–116.

[14] D.M. Warme, P. Winter, M. Zachariasen, GeoSteiner 3.1. http://www.diku.dk/geosteiner/, 2001.

[15] R.T. Wong, A dual ascent approach for Steiner tree problems on a directed graph, Math. Programming 28 (1984) 271–287.