Alcom-FT Technical Report Series
ALCOMFT-TR-02-34

On Steiner Trees and Minimum Spanning Trees in Hypergraphs*

Tobias Polzin
Max-Planck-Institut für Informatik,
Saarbrücken, Germany

Siavash Vahdati Daneshmand
Theoretische Informatik, Universität Mannheim,
Mannheim, Germany

Abstract

The bottleneck of the state-of-the-art algorithms for geometric Steiner problems is the concatenation-phase, where the hitherto most successful 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 superior 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 (see Karp [3]) with many important applications in network design in general and VLSI design in particular. Background information on this problem can be found in Hwang, Richards and Winter [2].

For geometric Steiner problems, an approach based on full Steiner trees has been successful (Warme, Winter and Zachariasen [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

*This work was partially supported by the Future and Emerging Technologies programme of the EU under contract number IST-1999-14186 (ALCOM-FT).

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 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 [7], 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 surprising result that a program package of ours [8, 9, 10] which is designed for general networks is superior to the MSTH-approach.

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 (a **Steiner minimal tree**).
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 $R_1 := R \backslash \{z_1\}$.
A Steiner tree T for a subset $S \subseteq R$ is called a **full Steiner tree (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 no 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 Minimum Spanning Trees in Hypergraphs: Formulations and Relaxations

We begin with a formulation of Warme [12] for the minimum spanning tree in hypergraph problem:

$$
\begin{aligned}
\boxed{P_{FST}} \qquad \sum_{T \in F} c_T X_T &\rightarrow \min, \\
\sum_{T \in F} (|T|-1) X_T &= |R|-1, && (1.1)\\
\sum_{T,\ T \cap S \neq \emptyset} (|T \cap S|-1) X_T &\leq |S|-1 \quad (\emptyset \neq S \subset R), && (1.2)\\
X_T &\in \{0,1\} \quad (T \in F). && (1.3)
\end{aligned}
$$

Lemma 1 Any feasible solution of 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 Magnanti and Wolsey [6], we get the following integer program:

$$
\begin{aligned}
\boxed{P_{FS\vec{T}}} \qquad \sum_{\vec{T} \in \vec{F}} c_{\vec{T}} x_{\vec{T}} &\rightarrow \min, \\
\sum_{\vec{T} \in \vec{F}} (|\vec{T}|-1) x_{\vec{T}} &= |R|-1, && (2.1)\\
\sum_{\vec{T},\ \vec{T} \in \Delta^-(z_t)} x_{\vec{T}} &= 1 \quad (z_t \in R_1), && (2.2)\\
\sum_{\vec{T},\ \vec{T} \cap S \neq \emptyset} (|\vec{T} \cap S|-1) x_{\vec{T}} &\leq |S|-1 \quad (\emptyset \neq S \subset R), && (2.3)\\
x_{\vec{T}} &\in \{0,1\} \quad (\vec{T} \in \vec{F}). && (2.4)
\end{aligned}
$$

It is easy to see (and it also follows from the proof of the next lemma) that $P_{FS\vec{T}}$ is a valid formulation of the minimum spanning tree in hypergraph problem.

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

Proof:

I) $v(LP_{FS\vec{T}}) \geq v(LP_{FST})$: Let x denote an optimal solution for $LP_{FS\vec{T}}$. For each FST T with corresponding directed FSTs $\vec{T}_1, \ldots, \vec{T}_k$ define $X_T = x_{\vec{T}_1} + \cdots + x_{\vec{T}_k}$. It is easy to check that X satisfies all constraints of LP_{FST} and yields the same value as $v(LP_{FS\vec{T}})$.

II) $v(LP_{FST}) \geq v(LP_{FS\vec{T}})$: Now let X denote an optimal solution for LP_{FST}. For each FST T with corresponding directed FSTs $\vec{T}_1, \ldots, \vec{T}_k$ consider a $\Gamma_T = (\Gamma_{T_1}, \ldots, \Gamma_{T_k})$ with $\Gamma_{T_1} + \cdots + \Gamma_{T_k} = 1$ and let $x_{\vec{T}_i} := \Gamma_{T_i} X_T$. Again, it is easy to validate that x satisfies the constraints (2.1) and (2.3) and yields the same value as $v(LP_{FST})$.
The only question is, whether there is a Γ such that x satisfies the constraints (2.2), too. This question can be stated in the following way:
Is it possible to distribute the "supply" X_T of each FST T in such a way to its leaves that every $z_t \in R_1$ gets one unit at the end?
It is known that the latter problem can be viewed as a flow problem: Construct a flow network with source s, sink t, and vertices u_T (for all $T \in F$) and w_i (for all $v_i \in R$). For each T with leaves $v_1, \ldots, v_k$ we introduce arcs $[u_T, w_1], \ldots, [u_T, w_k]$ with capacity ∞. Furthermore, we add arcs $[s, u_T]$ with capacity $(|T| - 1)X_T$ and arcs $[w_i, t]$ with capacity 1 (or 0, if $v_i = z_1$).
The question above is equivalent to the question, whether a flow from s to t with value $|R| - 1$ can be constructed. The max-flow min-cut theorem says that this is possible if and only if there is no cut $C = \{U, \bar{U}\}$ (with $s \in U$ and $t \notin U$) with capacity less than $|R| - 1$ (Obviously $U = \{s\}$ and $\bar{U} = \{t\}$ correspond to cuts with capacity $|R| - 1$).
Suppose that U corresponds to a cut C with minimum capacity. Define $W := \{v_i \in R \mid w_i \in U\}$, $F_W := \{T \in F \mid T \subseteq W\}$, and $F_U := \{T \in F \mid u_T \in U\}$. For every $T \in F_U$ ($u_T \in U$) with leaves $v_1, \ldots, v_k$, all vertices $w_1, \ldots, w_k$ must belong to U ($T \in F_W$), because otherwise the capacity of C would be ∞ which is not minimal. It follows that: $F_U \subseteq F_W$.
The capacity of C is:

$$
\begin{aligned}
|W \setminus \{z_1\}| \quad &+ \quad \sum_{T \in F \setminus F_U} (|T| - 1)X_T && (3)\\
&\geq \quad |W \setminus \{z_1\}| + \sum_{T \in F \setminus F_W} (|T| - 1)X_T \qquad (\text{since } F_U \subseteq F_W) && (4)\\
&\geq \quad |W| - 1 + \sum_{T \in F} (|T| - 1)X_T - \sum_{T \in F_W} (|T| - 1)X_T && (5)\\
&= \quad |W| - 1 + |R| - 1 - \sum_{T \subseteq W} (|T| - 1)X_T \qquad (\text{because of 1.1}) && (6)\\
&\geq \quad |R| - 1. \qquad (\text{because of 1.2}) \qquad \square && (7)
\end{aligned}
$$

Now consider the following cut-formulation of the minimum spanning tree in hypergraph problem:

$$
\begin{aligned}
\boxed{P_{FSC}} \qquad \sum_{\vec{T} \in \vec{F}} c_{\vec{T}} x_{\vec{T}} \quad &\rightarrow \quad \min, &&\\
\sum_{\vec{T} \in \vec{F}} (|\vec{T}| - 1)x_{\vec{T}} \quad &= \quad |R| - 1, && (8.1)\\
\sum_{\vec{T},\ \vec{T} \in \Delta^-(S)} x_{\vec{T}} \quad &\geq \quad 1 \quad (z_1 \notin S,\ S \cap R_1 \neq \emptyset), && (8.2)\\
x_{\vec{T}} \quad &\in \quad \{0, 1\} \quad (\vec{T} \in \vec{F}). && (8.3)
\end{aligned}
$$

It is easy to see (and it also follows from the proof of the next lemma) that P_{FSC} is a valid formulation of the minimum spanning tree in hypergraph problem.

Lemma 3 $LP_{FST\vec{}}$ is equivalent to LP_{FSC}.

Proof: First observe that for any x feasible for LP_{FSC} summing (8.2) for all $z_t \in R_1$ we have:

$$|R| - 1 \leq \sum_{z_t \in R_1} \sum_{\vec{T} \in \Delta^-(z_t)} x_{\vec{T}} \leq \sum_{\vec{T}} (|leaves(\vec{T})|) x_{\vec{T}} = \sum_{\vec{T}} (|\vec{T}| - 1) x_{\vec{T}} \ . \tag{9}$$

Together with (8.1) this means that x satisfies (2.2). It follows directly 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:

$$\begin{aligned}
\sum_{\vec{T},\ \vec{T} \cap S \neq \emptyset} (|\vec{T} \cap S| - 1) x_{\vec{T}} &= \sum_{\vec{T},\ \vec{T} \cap S \neq \emptyset} (|leaves(\vec{T}) \cap S| + |root(\vec{T}) \cap S| - 1) x_{\vec{T}} && (10) \\
&= \sum_{\vec{T},\ \vec{T} \cap S \neq \emptyset} |leaves(\vec{T}) \cap S| x_{\vec{T}} - \sum_{\vec{T},\ \vec{T} \cap S \neq \emptyset,\ root(\vec{T}) \notin S} x_{\vec{T}} && (11) \\
&= \sum_{z_t \in S} \sum_{\vec{T} \in \Delta^-(z_t)} x_{\vec{T}} - \sum_{\vec{T},\ \vec{T} \in \Delta^-(S)} x_{\vec{T}} && (12) \\
&= |S \cap R_1| - \sum_{\vec{T},\ \vec{T} \in \Delta^-(S)} x_{\vec{T}} && (13)
\end{aligned}$$

Now there are two cases:

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

$$\sum_{\vec{T},\ \vec{T} \cap S \neq \emptyset} (|\vec{T} \cap S| - 1) x_{\vec{T}} = |S| - \sum_{\vec{T},\ \vec{T} \in \Delta^-(S)} x_{\vec{T}} \tag{14}$$

This means that x satisfies (2.3) if and only if it satisfies (8.2).

(II) $z_1 \in S$**:**

$$\sum_{\vec{T},\ \vec{T} \cap S \neq \emptyset} (|\vec{T} \cap S| - 1) x_{\vec{T}} = |S| - 1 - \sum_{\vec{T},\ \vec{T} \in \Delta^-(S)} x_{\vec{T}} \tag{15}$$

So x satisfies (2.3), 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 [7] for the dicut relaxation in graphs) leads to the observation that for any optimal solution for LP_{FSC} without (8.1) 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 (8.1) does not change the optimal solution value of LP_{FSC}.

3 Relation to the Relaxations of the Steiner Problem in Networks

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

$$\boxed{P_C} \qquad \sum_{a\in A} c_a y_a \;\rightarrow\; \min,$$

$$\sum_{a\in\delta^-(S)} y_a \;\geq\; 1 \quad (z_1 \notin S,\ S\cap R_1 \neq \emptyset), \tag{16.1}$$

$$y_a \;\in\; \{0,1\} \quad (a\in A). \tag{16.2}$$

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 $\vec{T}_1,\ldots,\vec{T}_l$, 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 $v(LP_{FSC})$. The following example shows that $v(LP_{FSC})$ can indeed be larger than $v(LP_C)$. □

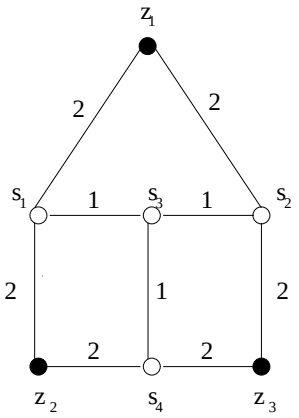

Figure 1: Example with $v(LP_C) < v(LP_{FSC}) = v(P_{FSC})$

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

Example 2 By turning s_3 to a terminal we get an example with $v(LP_{FSC}) < v(P_{FSC})$: $v(P_{FSC})$ is still 8; but $v(LP_{FSC})$ 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 $v(LP_{FST}) = v(P_{FST}) = 8$. Note also that $v(LP_C) = 7.5$ in both cases.

The relaxation LP_C can be strengthened by additional groups of constraints like the following one from Koch and Martin [4], which we call flow-balance constraints:

$$\sum_{a\in\delta^-(v_i)} y_a \leq \sum_{a\in\delta^+(v_i)} y_a \qquad (v_i \in V\setminus R). \tag{16.3}$$

In [7], we prove that these constraints indeed lead to a strictly stronger relaxation, which we call LP_{C+FB}.

Lemma 5 LP_{FSC} is (strictly) stronger than LP_{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 (16.3). The following example shows that $v(LP_{FSC})$ can indeed be larger than $v(LP_{C+FB})$. □

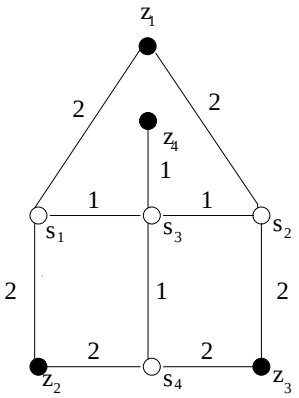

Figure 2: Example with $v(LP_{C+FB}) < v(LP_{FSC}) = v(P_{FSC})$

Example 3 The network G in Figure 2 with z_1 as the root gives an example for $v(LP_{C+FB}) < v(LP_{FSC})$: $v(P_{FSC}) = v(LP_{FSC}) = 9,\ v(LP_C) = 8.5$.

Since LP_{C+FB} is (strictly) stronger than all but two relaxations of the Steiner problem in [7], 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 [7] 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 Figure 1 with s_3 set to terminal gives an example with $v(LP_{2T}) = v(LP_{F^2}) > v(LP_C)$: As described in example 2, $v(LP_{FSC}) = 7.5$, but $v(LP_{2T}) = v(LP_{F^2}) = 8$.

Example 5 The network in Figure 1 with s_3 set to terminal and used as the root gives an example for $v(LP_{2T}) = v(LP_{F^2}) < v(LP_{FSC})$: $v(LP_{2T}) = v(LP_{F^2}) = 7.5$, but $v(P_{FSC}) = v(LP_{FSC}) = 8$ 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, $v(LP_{FSC})$ would be 7.5, too.
However, there are also examples such that $v(LP_{FSC}) = v(P_{FSC}) > v(LP_{F^2})$ even if all possible FSTs are generated in the first phase.

4 Experimental Study on the Relaxations and Corresponding Algorithms

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 the version 3.1 of GeoSteiner [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 package available for these problems. For the second approach, we use a software package developed by us [9] (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 [5]. 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 1. The FST generation phase (followed by a pruning phase to reduce the number of FSTs needed for the second phase) 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). Note that the results of the version 3.1 of GeoSteiner are very significantly better than the already published results [12, 13] of former versions.
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 Tables 2 and 4, we compare the optimal values and computation times for the relaxations LP_{FST} and LP_C. We used GeoSteiner 3.1 for LP_{FST}. We considered as $v(LP_{FST})$ the value of the last linear program before any branching was performed. This is somehow problematic, because for example the program performs fixing of variables, which can have an effect on the relaxation's value. But the authors of GeoSteiner pointed out to us that there is no better way to figure out the value of the relaxation from the program's output. We also give the optimum integer value and the respective gaps in percent. Studying these tables, one observes:

- Both relaxations yield almost always the same value. Only on four instances (es1000fst09, att532, d493 and nrw1379), 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 the integer optimum is in both cases less than 0.01%.

- With respect to running times, neither method is always faster than the other. The average running times for computing $v(LP_{FST})$ and $v(LP_C)$ are 99 respectively 80 seconds for ES1000FST-instances and 130 respectively 29 seconds for TSPFST-instances. But as obvious from the tables, these average times do not say much about which method would be faster on a specific instance.

In Tables 3 and 5, 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. The average running times of GeoSteiner and STEINER are 151 respectively 10 seconds for ES1000FST-instances and 262 respectively 3 seconds for TSPFST-instances. 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 the more time-consuming instances. For example, to solve the instance es10000fst from SteinLib (the largest instance of this type ever solved), GeoSteiner needs months of cpu-time, whereas STEINER needs less than 15 minutes, which is thousands of times faster. Also, all previously unsolved geometric instances in SteinLib could be solved by STEINER in relatively small time (see Table 1).

- 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 the value of the relaxation LP_{FST}, since it begins the concatenation phase mainly with the computation of the latter value. On the other hand, STEINER usually needs considerably less time for the exact solution than for the computation of LP_C-value. This is because STEINER begins the process of exact solution 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 [8, 9, 10, 11]).

5 Concluding Remarks

In this paper we studied one known and some new relaxations of the minimum spanning tree in hypergraph problem and compared them to the known linear relaxations of the Steiner problem in graphs when used in the concatenation phase of the FST-approach to geometric Steiner problems. We also performed some experiments on the quality of these MSTH relaxations when compared to the most used relaxation of the Steiner problem in graphs, the dicut relaxation. It turns out that although the MSTH relaxations are strictly stronger theoretically, the difference is less decisive empirically; and more surprisingly, our program STEINER which is designed for general networks performs the concatenation phase significantly faster than the program GeoSteiner, which is designed for this special case and was believed to be the fastest existing code for this problem.
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.

instance	size			optimum	STEINER	
	$\|R\|$	$\|V\|$	$\|E\|$		time (s)	nodes
es10000	10000	27019	39407	716174280	758	1
fl1400	1400	2694	4546	17980523	118	1
fl3795	3795	4859	6539	25529856	139	1
fnl4461	4461	17127	27352	182361	6148	1
pcb3038	3038	5829	7552	131895	2.4	1
pla7397	7397	8790	9815	22481625	0.1	1

Table 1: Instances not solved by GeoSteiner in one day

instance	optimum	LP_{FST}			LP_C		
		value	gap (%)	time (s)	value	gap (%)	time (s)
es1000fst01	230535806	230482682.0	0.02	7.04	230482682.0	0.02	107.64
es1000fst02	227886471	227885442.5	0.0005	7.31	227885442.5	0.0005	38.83
es1000fst03	227807756	227806391.5	0.001	113.86	227806391.5	0.001	35.15
es1000fst04	230200846	230194211.7	0.003	7.16	230194211.7	0.003	56.79
es1000fst05	228330602	228321717.5	0.004	62.18	228321717.5	0.004	56.06
es1000fst06	231028456	230990472.4	0.02	188.39	230990472.4	0.02	141.88
es1000fst07	230945623	230943876.5	0.001	86.69	230943876.5	0.001	38.14
es1000fst08	230639115	230628313.0	0.005	107.72	230628313.0	0.005	147.82
es1000fst09	227745838	227739972.4	0.003	13.62	227736751.9	0.004	95.79
es1000fst10	229267101	229237719.5	0.01	71.06	229237719.5	0.01	28.90
es1000fst11	231605619	231576667.0	0.01	11.99	231576667.0	0.01	44.43
es1000fst12	230904712	230885982.0	0.01	416.77	230885982.0	0.01	126.69
es1000fst13	228031092	228031092.0	0	3.02	228031092.0	0	45.63
es1000fst14	234318491	234270461.0	0.02	380.32	234270461.0	0.02	184.56
es1000fst15	229965775	229942631.0	0.01	10.28	229942631.0	0.01	53.58
Averages:			0.01	99.2		0.01	80.1

Table 2: Comparison of LP_{FST} and LP_C on ES1000FST-instances

instance	size				optimum	GeoSteiner		STEINER	
	$\|R\|$	$\|F\|$	$\|V\|$	$\|E\|$		time (s)	nodes	time (s)	nodes
es1000fst01	1000	2052	2865	4267	230535806	12.78	3	11.55	1
es1000fst02	1000	1943	2629	3793	227886471	9.38	1	7.79	1
es1000fst03	1000	2004	2762	4047	227807756	115.00	1	11.29	1
es1000fst04	1000	2024	2778	4083	230200846	8.41	1	12.52	1
es1000fst05	1000	1976	2676	3894	228330602	64.23	1	8.50	1
es1000fst06	1000	2033	2816	4164	231028456	409.78	10	16.13	1
es1000fst07	1000	1897	2604	3756	230945623	87.67	1	4.80	1
es1000fst08	1000	2047	2836	4210	230639115	111.38	1	12.32	1
es1000fst09	1000	2091	2846	4187	227745838	18.03	3	12.72	1
es1000fst10	1000	1894	2546	3620	229267101	112.97	5	4.76	1
es1000fst11	1000	2026	2763	4038	231605619	19.58	3	8.13	1
es1000fst12	1000	2136	2992	4500	230904712	484.46	2	16.47	1
es1000fst13	1000	1886	2532	3615	228031092	3.07	1	4.62	1
es1000fst14	1000	2049	2840	4200	234318491	791.82	13	14.92	1
es1000fst15	1000	2032	2735	4001	229965775	10.82	1	7.59	1
Averages:						150.6		10.3	

Table 3: Comparison of GeoSteiner (2nd phase) and STEINER on ES1000FST-instances

instance	optimum	LP_{FST}			LP_C		
		value	gap (%)	time (s)	value	gap (%)	time (s)
a280	2502	2502.0	0	0.02	2502.0	0	0.02
att48	30236	30236.0	0	0.02	30236.0	0	0.07
att532	84009	83996.0	0.02	184.53	83993.0	0.02	32.93
berlin52	6760	6760.0	0	0.01	6760.0	0	0.01
bier127	104284	104284.0	0	0.08	104284.0	0	0.12
d1291	481421	481421.0	0	0.40	481421.0	0	0.05
d1655	584948	584948.0	0	3.30	584948.0	0	0.39
d198	129175	129175.0	0	0.03	129175.0	0	0.01
d2103	769797	769797.0	0	1.07	769797.0	0	0.06
d493	320137	320137.0	0	260.31	320136.5	0.0002	3.66
d657	471589	471493.5	0.02	296.54	471493.5	0.02	8.52
dsj1000	17564659	17564463.5	0.001	15.07	17564463.5	0.001	10.67
eil101	605	604.7	0.1	0.24	604.7	0.1	2.11
eil51	409	408.5	0.1	0.08	408.5	0.1	0.28
eil76	513	512.5	0.1	0.03	512.5	0.1	1.06
fl1577	19825626	19825626.0	0	34.64	19825626.0	0	18.23
fl417	10883190	10881228.3	0.02	3.31	10881228.3	0.02	0.96
gil262	2306	2306.0	0	0.50	2306.0	0	0.45
kroA100	20401	20401.0	0	0.01	20401.0	0	0.02
kroA150	25700	25700.0	0	0.17	25700.0	0	0.35
kroA200	28652	28652.0	0	0.19	28652.0	0	0.30
kroB100	21211	21208.5	0.01	0.06	21208.5	0.01	0.06
kroB150	25217	25217.0	0	0.13	25217.0	0	0.67
kroB200	28803	28792.0	0.04	0.20	28792.0	0.04	0.54
kroC100	20492	20492.0	0	0.04	20492.0	0	0.12
kroD100	20437	20437.0	0	0.03	20437.0	0	0.07
kroE100	21245	21245.0	0	0.04	21245.0	0	0.07
lin105	13429	13412.3	0.1	0.21	13412.3	0.1	0.11
lin318	39335	39316.8	0.05	11.17	39316.8	0.05	1.57
linhp318	39335	39316.8	0.05	11.11	39316.8	0.05	1.49
nrw1379	56207	56199.5	0.01	7466.44	56199.0	0.01	1636.43
p654	314925	314925.0	0	0.13	314925.0	0	0.04
pcb1173	53301	53301.0	0	2.59	53301.0	0	1.01
pcb442	47675	47675.0	0	0.07	47675.0	0	0.02
pr1002	243176	243151.0	0.01	0.59	243151.0	0.01	0.48
pr107	34850	34850.0	0	0.01	34850.0	0	0.01
pr124	52759	52759.0	0	0.01	52759.0	0	0.01
pr136	86811	86811.0	0	0.07	86811.0	0	0.04
pr144	52925	52925.0	0	0.04	52925.0	0	0.04
pr152	64323	64323.0	0	0.16	64323.0	0	0.09
pr226	70700	70700.0	0	0.06	70700.0	0	0.01
pr2392	358989	358989.0	0	6.22	358989.0	0	4.28
pr264	41400	41400.0	0	0.03	41400.0	0	0.01
pr299	44671	44671.0	0	0.11	44671.0	0	0.08
pr439	97400	97400.0	0	0.55	97400.0	0	0.07
pr76	95908	95908.0	0	0.04	95908.0	0	0.04
rat195	2386	2386.0	0	0.09	2386.0	0	1.77
rat575	6808	6807.5	0.01	1.41	6807.5	0.01	89.25
rat783	8883	8883.0	0	5.90	8883.0	0	163.79
rat99	1225	1225.0	0	0.05	1225.0	0	0.31
rd100	764269099	764269099.0	0	0.02	764269099.0	0	0.06
rd400	1490972006	1490972006.0	0	0.62	1490972006.0	0	4.29
rl11849	8779590	8779240.0	0.004	813.38	8779240.0	0.004	13.08
rl1304	236649	236649.0	0	0.45	236649.0	0	0.13
rl1323	253620	253620.0	0	1.01	253620.0	0	0.19
rl1889	295208	295193.5	0.005	1.10	295193.5	0.005	1.09
rl5915	533226	533142.0	0.02	49.22	533142.0	0.02	1.27
rl5934	529890	529878.0	0.002	11.39	529878.0	0.002	2.34
st70	626	626.0	0	0.01	626.0	0	0.02
ts225	1120	1120.0	0	0.01	1120.0	0	0.01
tsp225	356850	356850.0	0	0.02	356850.0	0	0.01
u1060	21265372	21262031.3	0.02	8.26	21262031.3	0.02	3.66
u1432	1465	1465.0	0	0.02	1465.0	0	0.01
u159	390	390.0	0	0.01	390.0	0	0.01
u1817	5513053	5513053.0	0	0.08	5513053.0	0	0.02
u2152	6253305	6253305.0	0	0.23	6253305.0	0	0.01
u2319	2322	2322.0	0	0.03	2322.0	0	0.01
u574	3509275	3509275.0	0	0.59	3509275.0	0	0.69
u724	4069628	4069628.0	0	1.42	4069628.0	0	2.02
vm1084	2248390	2248390.0	0	1.71	2248390.0	0	1.89
vm1748	3194670	3194542.5	0.004	5.95	3194542.5	0.004	14.14
Averages:			0.01	129.6		0.01	28.6

Table 4: Comparison of LP_{FST} and LP_C 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	30236	0.02	1	0.30	1
att532	532	1065	1468	2152	84009	195.72	1	3.74	1
berlin52	52	78	89	104	6760	0.01	1	0.01	1
bier127	127	213	258	357	104284	0.08	1	0.02	1
d1291	1291	1361	1365	1456	481421	0.40	1	0.01	1
d1655	1655	1879	1906	2083	584948	3.30	1	0.04	1
d198	198	232	232	256	129175	0.03	1	0.01	1
d2103	2103	2196	2206	2272	769797	1.07	1	0.02	1
d493	493	966	1055	1473	320137	261.32	1	0.58	1
d657	657	1176	1416	1978	471589	312.40	3	1.51	1
dsj1000	1000	1884	2562	3655	17564659	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	19825626	34.64	1	0.99	1
fl417	417	872	732	1084	10883190	7.01	13	0.85	1
gil262	262	447	537	723	2306	0.53	1	0.06	1
kroA100	100	165	197	250	20401	0.01	1	0.02	1
kroA150	150	296	389	562	25700	0.18	1	0.69	1
kroA200	200	389	500	714	28652	0.19	1	0.24	1
kroB100	100	180	230	313	21211	0.06	1	0.05	1
kroB150	150	297	420	619	25217	0.13	1	0.42	1
kroB200	200	369	480	670	28803	0.22	1	0.62	1
kroC100	100	190	244	337	20492	0.05	1	0.10	1
kroD100	100	166	216	288	20437	0.03	1	0.02	1
kroE100	100	176	226	306	21245	0.04	1	0.13	1
lin105	105	190	216	323	13429	0.29	1	0.12	1
lin318	318	589	738	1279	39335	15.26	4	0.43	1
linhp318	318	589	678	1030	39335	15.19	4	0.40	1
nrw1379	1379	3590	5096	8105	56207	16410.99	32	150.14	1
p654	654	760	777	867	314925	0.15	1	0.01	1
pcb1173	1173	1708	1912	2223	53301	2.77	1	0.11	1
pcb442	442	494	503	531	47675	0.07	1	0.01	1
pr1002	1002	1392	1474	1717	243176	0.70	1	0.05	1
pr107	107	110	111	110	34850	0.01	1	0.01	1
pr124	124	147	154	165	52759	0.01	1	0.01	1
pr136	136	227	196	250	86811	0.07	1	0.01	1
pr144	144	184	221	285	52925	0.04	1	0.01	1
pr152	152	242	308	431	64323	0.16	1	0.06	1
pr226	226	248	255	269	70700	0.06	1	0.01	1
pr2392	2392	3311	3398	3966	358989	6.35	1	0.06	1
pr264	264	280	280	287	41400	0.03	1	0.01	1
pr299	299	416	420	500	44671	0.11	1	0.01	1
pr439	439	551	572	662	97400	0.55	1	0.01	1
pr76	76	138	168	247	95908	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	764269099	0.02	1	0.01	1
rd400	400	747	1001	1419	1490972006	0.62	1	1.49	1
rl11849	11849	13780	13963	15315	8779590	1100.43	9	0.64	1
rl1304	1304	1514	1562	1694	236649	0.51	1	0.04	1
rl1323	1323	1545	1598	1750	253620	1.01	1	0.01	1
rl1889	1889	2247	2382	2674	295208	1.56	1	0.22	1
rl5915	5915	6540	6569	6980	533226	76.48	10	0.09	1
rl5934	5934	6739	6827	7365	529890	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	356850	0.02	1	0.01	1
u1060	1060	1708	1835	2429	21265372	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	5513053	0.08	1	0.01	1
u2152	2152	2166	2167	2184	6253305	0.23	1	0.02	1
u2319	2319	2318	2319	2318	2322	0.03	1	0.01	1
u574	574	877	990	1258	3509275	0.59	1	0.17	1
u724	724	1093	1180	1537	4069628	1.42	1	0.22	1
vm1084	1084	1474	1679	2058	2248390	1.71	1	0.37	1
vm1748	1748	2488	2856	3641	3194670	7.07	3	1.98	1
Averages:						261.8		3.0	

Table 5: Comparison of GeoSteiner (2nd phase) and STEINER on TSPFST-instances

References

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

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

[3] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, 1972.

[4] T. Koch and A. Martin. Solving Steiner tree problems in graphs to optimality. *Networks*, 32:207–232, 1998.

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

[6] T. L. Magnanti and L. A. Wolsey. Optimal Trees. In M. O. Ball et al., editors, *Handbooks in Operations Research and Management Science*, volume 7, chapter 9. Elsevier Science, 1995.

[7] T. Polzin and S. Vahdati Daneshmand. A comparison of Steiner tree relaxations. *Discrete Applied Mathematics*, 112:241–261, 2001.

[8] T. Polzin and 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.

[9] T. Polzin and S. Vahdati Daneshmand. Improved algorithms for the Steiner problem in networks. *Discrete Applied Mathematics*, 112:263–300, 2001.

[10] T. Polzin and 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 and S. Vahdati Daneshmand. Using (sub)graphs of small width for solving the Steiner problem. Research Report MPI-I-2002-1-009, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany, 2002.

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

[13] D. M. Warme, P. Winter, and M. Zachariasen. Exact algorithms for plane Steiner tree problems: A computational study. In D-Z. Du, J. M. Smith, and J. H. Rubinstein, editors, *Advances in Steiner Trees*, pages 81–116. Kluwer Academic Publishers, 2000.

[14] D. M. Warme, P. Winter, and 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. *Mathematical Programming*, 28:271–287, 1984.