A New Exact Algorithm for Rectilinear Steiner Trees

David M. Warme

August 21, 1997

Abstract

Given a finite set V of points in the plane (called *terminals*), the rectilinear Steiner minimal tree is a shortest network of horizontal and vertical lines connecting all the terminals of V. The decision form of this problem has been shown to be NP-complete [10].

The FST concatenation approach has been more successful in practice than all other currently known methods for solving the geometric Steiner tree problems. The final (and most time-consuming) step in this method is to solve the FST concatenation problem, which is equivalent to finding a minimum spanning tree in a certain hypergraph.

We show that the MST in hypergraph problem is NP-complete. We then formulate this problem as an integer program, and present a branch-and-cut algorithm for solving it. We define $STHGP(n)$ — the *spanning tree in hypergraph polytope* and prove a number of its properties. In particular, we show that two of the classes of constraints in our formulation define facets of $STHGP(n)$. We present a complete list of the facets of $STHGP(n)$ for $2 \leq n \leq 4$.

This algorithm has obtained optimal solutions for **all** of the OR-library [3] problem instances having 1000 points or fewer. We present detailed computational results on these problems, as well as aggregate results from a more extensive computational study with random data. This represents a substantial advance since the best rectilinear Steiner tree results currently in the literature are 35 points [24], 40 points [18], and 55 points [7].

Using the Euclidean FST generator of Winter and Zachariasen [28], this algorithm has also obtained optimal Euclidean Steiner trees for problems as large as 2000 terminals.

1 Introduction

Given a finite set V of points in the plane (called *terminals*), the *rectilinear Steiner minimal tree* (RSMT) is a shortest network of horizontal and vertical lines connecting all the terminals of V. The resulting interconnection is a tree. Each terminal $t \in V$ is a node in this tree, although there may be others. Nodes $s \notin V$ of degree 3 or greater are known as *Steiner points*. The decision form

of RSMT has been shown to be NP-complete [10]. As is common in the literature, we say that the optimization form of an NP-complete problem is NP-hard.

Figure 1 illustrates an RSMT for 70 terminals.

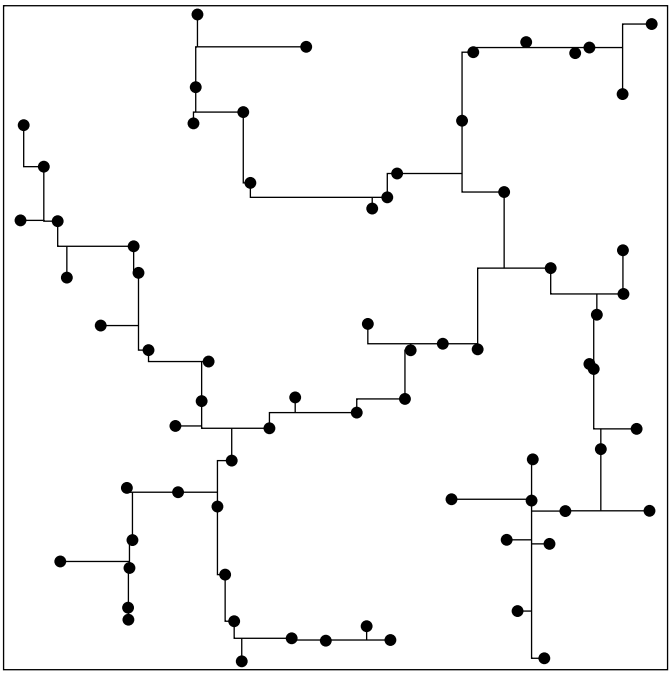

Figure 1: A rectilinear Steiner minimal tree for 70 terminals.

The restriction to using only horizontal and vertical lines is a consequence of minimizing total length as measured using the rectilinear (or L_1) distance metric — in which the distance between points (x_1, y_1) and (x_2, y_2) is given by $|x_1 - x_2| + |y_1 - y_2|$. The other distance metric commonly used for Steiner trees is the *Euclidean* distance metric, in which the distance is defined to be $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Such trees are called *Euclidean Steiner minimal trees* (ESMT). See figure 2 for an illustration of an ESMT of 100 terminals. (Figure 2 is an optimal solution of instance 1 from the `estein100.txt` file of Beasley's OR-library [2], [3]). Unless otherwise specified, all Steiner trees in this paper will be assumed to be RSMTs.

The RSMT problem has numerous applications in the area of VLSI design automation as well as printed circuit board layout. For example, an RSMT for a set of electron devices can be used as a lower bound estimate on the wire length of a route connecting all of the devices together. An RSMT of the points represents only a lower bound since a real interconnect satisfies additional constraints requiring it to avoid other obstacles that are also present on the chip. Recent work

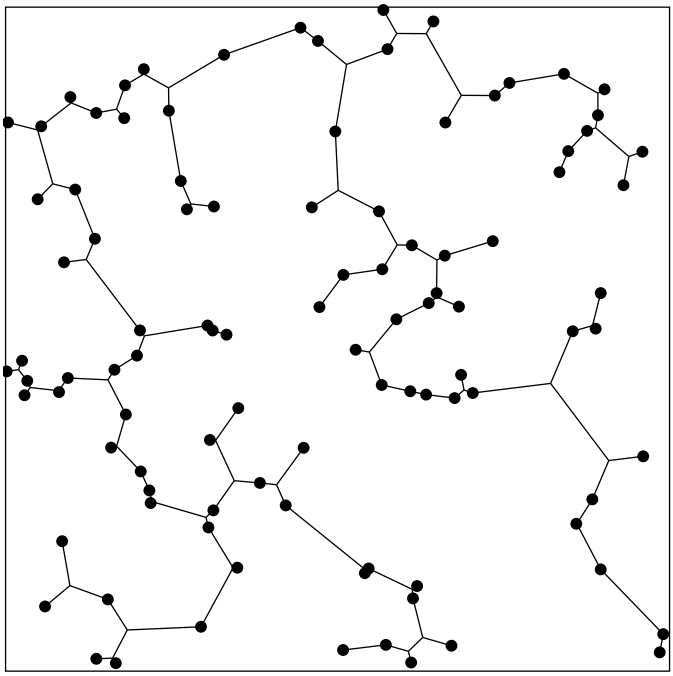

Figure 2: A Euclidean Steiner minimal tree for 100 terminals. (Problem 1 from OR-library `estein100.txt` file.)

by Ganley [8] treated such *obstacle-avoiding* RSMTs directly. In addition to global wire length estimation, RSMTs have also been used to evaluate the merit of functional block placements in floor-planners such as the *MONDRIAN* system [8]. Many of these applications require the solution of problem instances containing many hundreds or even thousands of terminals. Provably optimal solutions to such instances were well beyond the capabilities of previous methods, but are becoming feasible with the algorithm presented here.

2 Definitions

Let V be a set of n points in the plane called *terminals*. A Steiner minimal tree T for V is said to have a *full topology* if every vertex in V is a leaf in T. A terminal set V is said to be a *full set* if every Steiner minimal tree for V is a full topology. A terminal set that is a full set and also has size k is said to be a *full set of size k*. With respect to a point set V, a set $S \subseteq V$ is said to be a *full set with respect to V* if S is a full set, and there is some Steiner minimal tree for V that contains a full topology of S as a subgraph. A Steiner minimal tree T for a full set $S \subseteq V$ is said to be a *full Steiner tree* (FST) of V.

The following definitions are adapted from [4]. A *hypergraph* $H = (V, E)$ consists of a finite set V of vertices, and a set $E \subseteq 2^V$ of *hyperedges*. In this paper we use hyperedges to represent FSTs, and so we also require that $|e| \geq 2$ for all $e \in E$. We say that e is a *k-edge* of H if $e \in E$ and $|e| = k$. In a hypergraph $H = (V, E)$, a *chain of length q from v_0 to v_q* is defined to be a sequence v_0, e_1, v_1, e_2, v_2, ..., e_q, v_q such that

1. $v_0, v_1, \ldots, v_q \in V$,
2. $v_0, v_1, \ldots, v_{q-1}$ are distinct,
3. $e_1, e_2, \ldots, e_q \in E$ and are distinct, and
4. $v_{i-1} \in e_i \wedge v_i \in e_i$ for $i = 1, 2, \ldots, q$.

If $q > 1$ and $v_0 = v_q$, then this chain is called a *cycle of length q*. We may omit either or both of the phrases "length q" and "from v_0 to v_q" when they are apparent or arbitrary. $H' = (V', E')$ is a *subhypergraph* of $H = (V, E)$ if $V' \subseteq V$, and for every $e' \in E'$ there is an $e \in E$ such that $e' = e \cap V'$ and $|e'| \geq 2$. A hypergraph $H = (V, E)$ is *connected* if for every $s, t \in V$ there is a chain from s to t in H. A hypergraph $H = (V, E)$ is a *tree* if for every $s, t \in V$ there is a unique chain from s to t in H. Subhypergraph $H' = (V, E')$ of $H = (V, E)$ is a *spanning tree* if H' is a tree.

If w is a function defined on E then for any subset $F \subseteq E$ we define $w(F) = \sum_{e \in F} w_e$. For a subset S of V, let $E(S)$ denote $\{e \in E : e \subseteq S\}$, let $E(S)_k$ denote $\{e \in E : e \subseteq S \wedge |e| = k\}$, let $\delta(S)$ denote $\{e \in E : 1 \leq |e \cap S| < |e|\}$, and let $\delta(S)_k$ denote $\{e \in E : 1 \leq |e \cap S| < |e| = k\}$. For $A, B \subseteq V$, let $(A : B)$ be $\{e \in E : (e \cap A \neq \emptyset) \wedge (e \cap B \neq \emptyset)\}$. We call $(S : V - S)$ a *cut* with shores S and $V - S$. Let $H[S]$ denote the subhypergraph induced by S, i.e. the hypergraph (S, E'), where $E' = \{e \cap S : e \in E \wedge |e \cap S| \geq 2\}$.

3 The FST Concatenation Method

In this section we give a brief overview of the FST concatenation method for computing Steiner minimal trees. See [16] for a more comprehensive treatment of Steiner tree results and methods.

The following is a well-known *folk theorem* of Steiner tree lore:

Theorem 3.1 *Let V be a set of terminals, with $|V| \geq 2$. Then V has a Steiner minimal tree that consists of one or more full topologies over full sets with respect to V. These full topologies intersect only at terminals of degree two or greater.*

This theorem suggests an approach for constructing a Steiner minimal tree for V as follows:

- Compute a set $F^* \subseteq 2^V$ such that (S is a full set with respect to $V \Rightarrow S \in F^*$).
- For each $S \in F^*$ identified above, compute a Steiner minimal tree T_S. Let its length be L_S.
- Determine a set $E' \subseteq F^*$ such that hypergraph (V, E') is a spanning tree that minimizes $L(E')$.
- Then $T = \cup_{S \in E'} T_S$ is a Steiner minimal tree for V.

We refer to the first two steps as *FST generation*, while the last two steps are called *FST concatenation*.

Note that we *do not* require every member of F^* to be a full set with respect to V — only that all such full sets be present in F^*. Naturally since we will then look for a minimal spanning subset of F^*, we will want $|F^*|$ to be as small as possible. Since there is no efficient algorithm known for identifying full sets with respect to V, we refer to the members of F^* as *candidate full sets*. In the sequel we will neglect the distinction between *candidate* full sets (or FSTs) and *genuine* full sets (or FSTs).

The FST generation process is highly metric dependent. For the Euclidean metric, [27] and [28] appears to be the most efficient in practice. The first FST generation algorithm for the rectilinear metric was given in [24] and [25]. However, substantial improvements have recently been obtained by Zachariasen [29].

Cockayne and Hewgill [5], [6] discovered that a *compatibility matrix* C can be computed from F^*. Let $A, B \in F^*$. Then $C_{A,B}$ is either *compatible*, *incompatible*, or *disjoint*. A and B are *disjoint* if they have no terminals in common. They are *compatible* if they share a single common terminal and could appear together in an optimal Steiner tree for V. A and B are said to be *incompatible* if it can be shown that they *cannot* appear together in an optimal Steiner tree for V. All known tests for incompatibility ultimately appeal to the non-minimality of solutions containing the incompatible pair. Having a significant number of incompatible FST pairs greatly reduces the search space during the FST concatenation phase. In this paper we require only an *incompatible* relation on the members of F^*: the compatible and disjoint relations are not used.

An optional "FST pruning" phase iteratively deduces that certain FST pairs may be safely relabeled as *incompatible*. During this process a number of FSTs may be completely eliminated from further consideration. See [6], [25], [28] and [7] for more details on FST pruning.

The final step is FST concatenation, which is equivalent to the problem of finding a minimum spanning tree in the hypergraph $H = (V, F^*)$, with each edge $e \in F^*$ having weight L_e. We present below a proof that this problem is NP-complete. In practice, the *incompatible* relation has proven to be an important ingredient in this process — especially on very large problems.

As an example of the FST concatenation method, figures 3 and 4 illustrate all 216 rectilinear candidate full sets obtained by the Salowe-Warme algorithm [24], [25] for the 70 point problem shown in figure 1. This figure plots the FST T_S corresponding to each candidate full set S. The reader may verify that the RSMT shown in figure 1 is the union of 35 candidate full sets, each of which can be found among the 216 listed in figures 3 and 4.

The FST concatenation method therefore consists of these steps:

- *Phase 1:* Metric dependent steps
  - FST generation
  - FST compatibility information
  - FST pruning
- *Phase 2:* Metric independent step
  - FST concatenation

For the sake of brevity, we often refer to the entire phase 1 as "FST generation".

To summarize, the input to phase 1 is a point set V, whereas its output is a list F^* of candidate full sets (together with an optimal Steiner tree T_S of known length L_S for each $S \in F^*$), as well as a compatibility matrix C. Phase 2 takes the list F^*, T_S, L_S, and the compatibility matrix C and finds a minimum spanning subset of the F^*.

Let $n = |V|$ and $m = |F^*|$. Using the Salowe-Warme algorithm for generating rectilinear FSTs we typically find that $4n \leq m \leq 6n$. Using extensive FST pruning, Fossmeier and Kaufmann [7] typically achieve $m \leq 1.7\,n$, although their methods are currently quite costly. Unfortunately, the best currently known bound is $m = O(n \cdot 2^{n/2}) \approx O(n \cdot 1.42^n)$ [7]. No procedure is known for generating point sets V having pathologically large numbers of FSTs. Some results hint at an $O(n^2)$ bound in the average case [25].

The lack of a polynomial bound on the number of FSTs is a significant theoretical weakness of the FST concatenation method, and an important open problem. We conjecture that the worst-case bound is $O(n^{3/2})$ or less. We also conjecture that $E[m] = O(n)$.

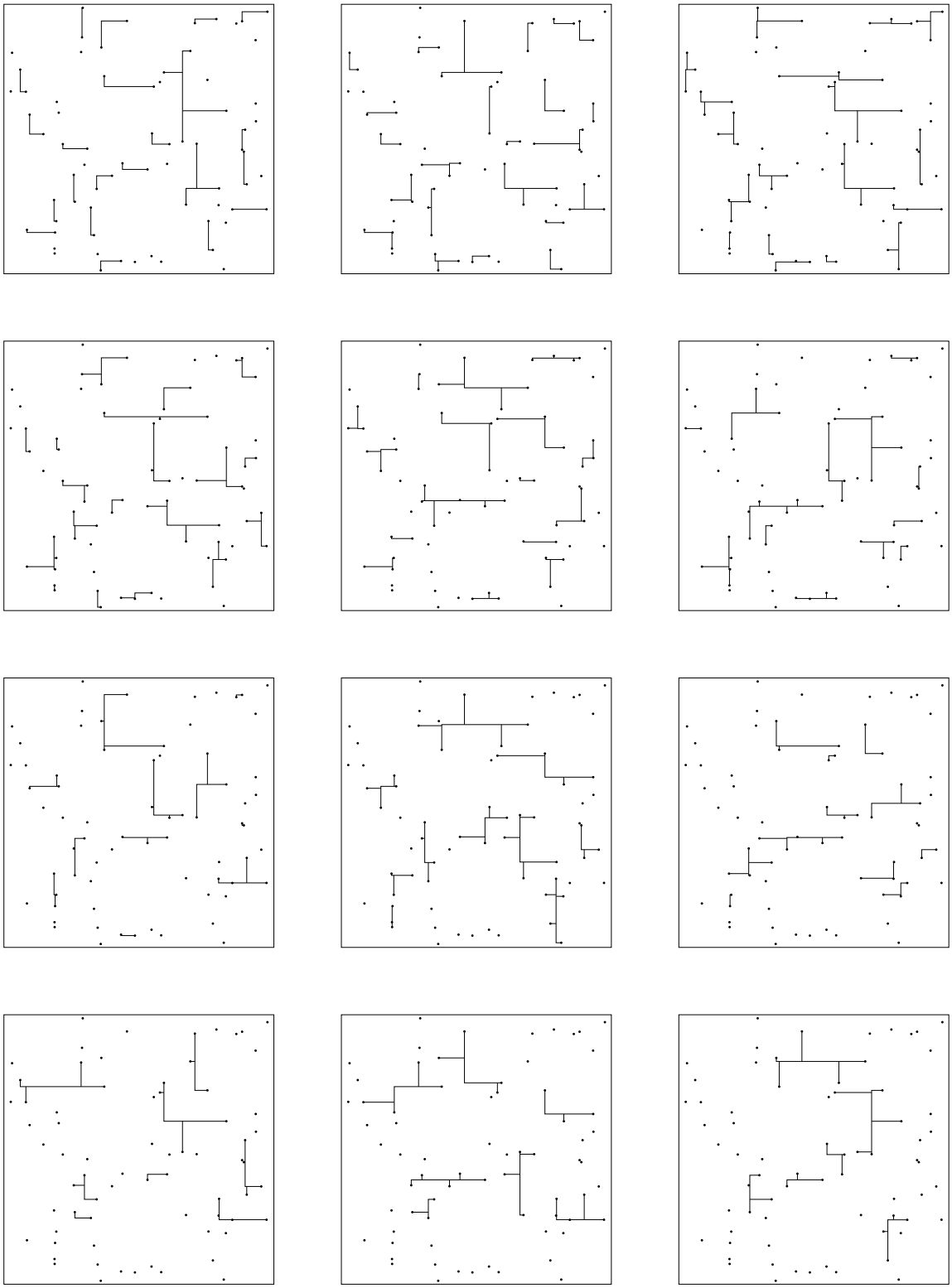

Figure 3: All rectilinear FSTs for problem in figure 1.

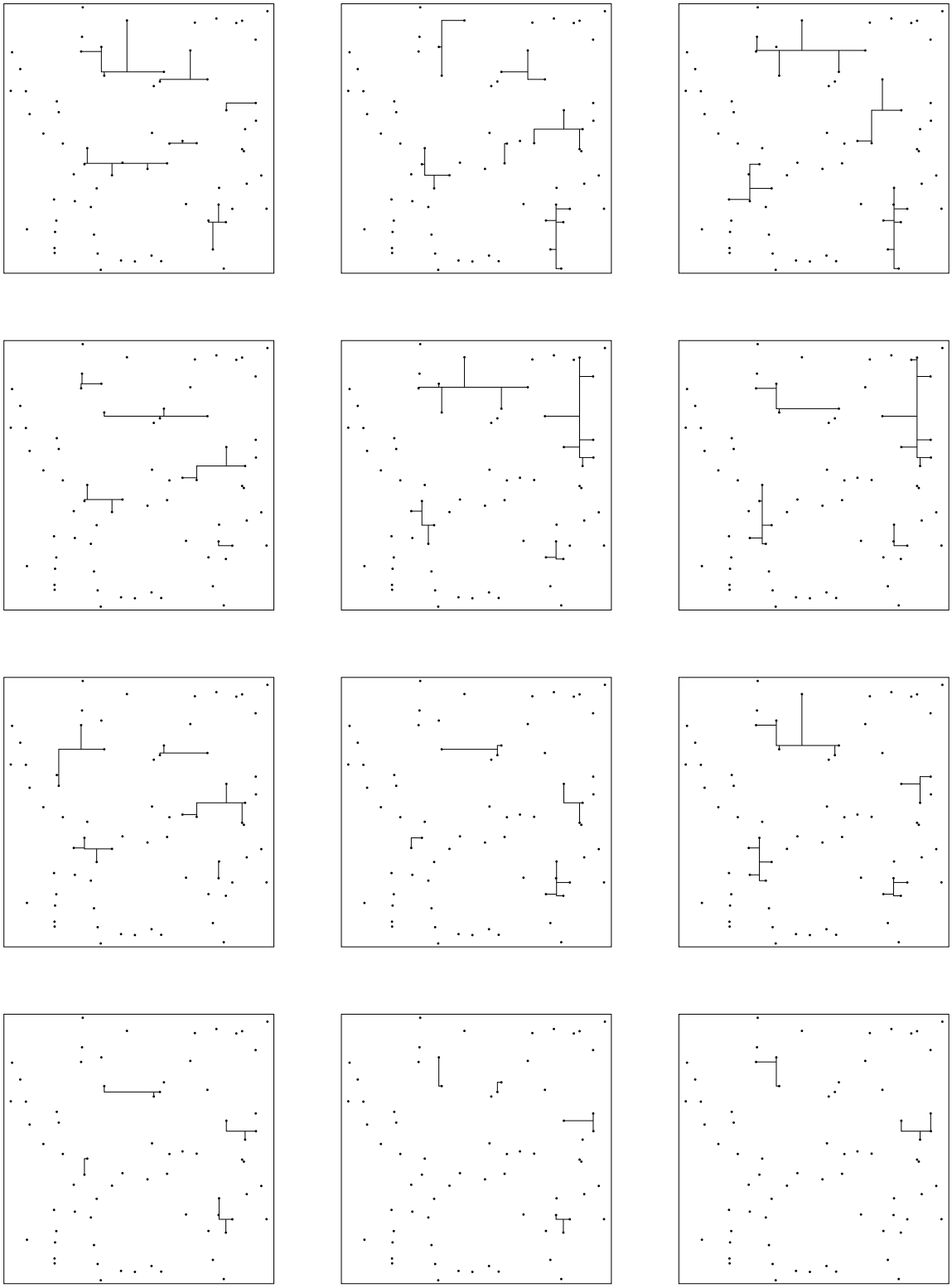

Figure 4: All rectilinear FSTs for problem in figure 1 (cont).

The rest of the paper focuses on the FST concatenation problem. For further discussion of the FST generation, compatibility and pruning problems, please refer to the referenced papers.

4 Minimum Spanning Tree in Hypergraph is NP-Complete

We define the following decision problem:
Problem MINIMUM SPANNING TREE IN HYPERGRAPH (**MSTHG**):
Instance: Finite hypergraph $H = (V_H, E_H)$, integer edge weights w_e for all $e \in E_H$, positive integer L.
Question: Is there a subhypergraph $T = (V_H, E_T)$ of H of total weight L or less such that

(i) T is connected,

(ii) if $g, h \in E_T$ then $|g \cap h| \leq 1$, and

(iii) $w(E_T) \leq L$?

The following decision problem is well-known to be NP-complete:
Problem INDEPENDENT SET: Instance: Graph $G = (V_G, E_G)$, positive integer K.
Question: Is there an independent set I of size K or more in G (i.e. a subset $I \subseteq V$ with $|I| \geq K$ such that $|\{u, v\} \cap I| \leq 1$ for all $(u, v) \in E_G$)?
Remark: Remains NP-complete if we assume that G is connected and all vertices $v \in V_G$ have degree $\deg_G(v) \geq 2$.

We now present the main result of this section:

Theorem 4.1 *Problem* **MSTHG** *is NP-complete.*

The following proof was devised by Queyranne [22]:

Proof: *Reduction from problem INDEPENDENT SET. Given G and K, construct H, w and L as follows:*
$V_H = E'_G \cup E''_G$, where E'_G is a copy of E_G and E''_G is another disjoint copy. Denote the elements of V_H as $e' \in E'_G$ and $e'' \in E''_G$ for every $e \in E_G$.
$E_H = E^*_G \cup T_0 \cup V'_H$ where
E^*_G consists of the subsets $\{e', e''\}$ for all $e \in E_G$,
T_0 is a set of edges $\{e', f'\}$ (distinct $e, f \in E_G$) that forms a spanning tree of E'_G,
V'_H is the set of all $v' = \{e', e'' : \text{ all } e \text{ incident to } v \text{ in } G\}$ for every $v \in V_G$. (That is, the image

v' of a vertex $v \in V_G$ is a hyperedge containing both copies e' and e'' of every edge e incident to v in G.) The weights are $w_h = 0$ for all $h \in E_G^*$, $w_h = 1$ for all $h \in T_0$, and $w_h = \deg_G(v) - 2$ for all $h = v' \in V_H'$.

Finally, let $L = |E_G| - 1 - K$.

To see why the reduction works, note that condition (ii) implies that if T is a spanning tree then $E_T \cap V_H'$ defines an independent set in G. Conversely, an independent set I in G induces a spanning tree $T(I)$ in H by adding edges from T_0 and E_G^* to satisfy (i). (There is some choice of which T_0 edges to include, but this does not matter as they all have the same weight of 1.) It is easy to show by induction on $|I|$ that $w(T(I)) = |E_G| - 1 - |I|$. □

5 IP Formulation

We now formulate both the *minimum spanning tree in hypergraph problem* and the *FST concatenation problem* as an integer program.

To solve a minimum spanning tree in hypergraph problem, we are given a hypergraph $H = (V, E)$ and weights c_e for all $e \in E$. In the sequel, we assume that $F^* = E$ and that the incompatibility relation is empty.

To solve the FST concatenation problem, we are given a finite set V of terminals. Let F^* be the set of candidate full sets for V. Let $n = |V|$, and $m = |F^*|$. Let C be the compatibility matrix for F^*. Let T_e be an RSMT of length c_e for all $e \in F^*$. Let c be the vector in $\Re^m$ whose components are the lengths c_e.

Let P be the polyhedron defined by the following constraints:

$$\sum_{e \in F^*} (|e| - 1)\, x_e \;=\; n - 1, \tag{5.1}$$

$$x_e + x_f \leq 1 \quad \text{for all } e, f \in F^*,\ e \neq f, \text{ and } e \text{ incompatible with } f, \tag{5.2}$$

$$x(S : V - S) \geq 1 \quad \text{for all } S \subseteq V \text{ with } 1 \leq |S| < n, \tag{5.3}$$

$$\sum_{e \in F^*} \max(|e \cap S| - 1,\, 0)\, x_e \leq |S| - 1 \quad \text{for all } S \subseteq V \text{ with } 2 \leq |S| < n, \tag{5.4}$$

$$0 \;\leq\; x_e \;\leq\; 1 \quad \text{for all } e \in F^*. \tag{5.5}$$

Then the solution to the following integer program yields an SMT for V:

$$\min \{c\, x : x \in P \cap \mathcal{Z}^m\} \tag{5.6}$$

In this formulation, the variable $x_e = 0$ for those $e \in F^*$ that are not part of the SMT, while $x_e = 1$ for those $e \in F^*$ that are in the SMT. Constraints (5.5), together with the integrality requirements of (5.6) guarantee that each full set will either be selected or not.

In section 8 we show that the single equation (5.1) is valid (i.e. satisfied by all spanning trees for V). This equation requires correct number and cardinality of hyperedges needed to construct a spanning tree. Note, however, that this equation does not demand a tree, since it neither guarantees connectedness nor prohibits cycles.

The constraints (5.2) prohibit any pair of incompatible full sets from being present in a solution. This is a reasonble thing to require, since an SMT is guaranteed to exist that does not violate any of these constraints. These constraints are omitted when solving the general *minimum spanning tree in hypergraph* problem.

Constraints (5.3) require that a solution be connected — for any cut with shores S and $V - S$, there must be at least one chosen candidate full set that crosses it.

Constraints (5.4) eliminate cycles. They extend the standard notion of subtour elimination constraints (SECs) for spanning trees on weighted graphs [20] to "generalized SECs" over weighted hypergraphs. In section 8 we show that these constraints are facet-defining.

6 Branch-and-Cut Procedure

We solve integer program (5.6) via branch-and-cut. The lower bounds are provided by the following linear program relaxation of (5.6):

$$\min \{c\, x : x \in P\} \tag{6.7}$$

Because there are an exponential number of constraints (5.3) and (5.4), we solve this LP via iterations of optimization followed by separation.

Let P_0 be the polyhedron defined by the following constraints:

$$\sum_{e \in F^*} (|e| - 1)\, x_e \;=\; n - 1, \tag{6.8}$$

$$x_e + x_f \leq 1 \quad \text{for all } e, f \in F^*,\ e \neq f, \text{ and } e \text{ incompatible with } f, \tag{6.9}$$

$$x(\{t\} : V - \{t\}) \geq 1 \quad \text{for all } t \in V, \tag{6.10}$$

$$0 \;\leq\; x_e \;\leq\; 1 \quad \text{for all } e \in F^*. \tag{6.11}$$

P_0 is the initial set of constraints — the total degree constraint, all of the incompatibility constraints, and the single-terminal cutset constraints. Begin the iteration with $j = 0$. Let $\vec{x_j}$ be an optimal solution to the following linear program:

$$\min \{c\, x : x \in P_j\} \tag{6.12}$$

Let $\mathcal{C}_j$ be any collection of constraints from (5.3) and/or (5.4) that are violated by $\vec{x_j}$. If no such constraints exist, then $\vec{x_j}$ is an optimal solution to (6.7). Otherwise, define polyhedron P_{j+1} by adding the constraints $\mathcal{C}_j$ to those of polyedron P_j, increment j, and iterate. Grötschel, Lovasz and Schrijver ([11], [12]) have shown that this process terminates in a polynomial number of iterations.

The optimal value of LP (6.7) is the lower bound used in the branch-and-bound procedure. This lower bound has been extremely tight in practice. For most problems in the computational study, the optimal solution to (6.7) is integral.

6.1 Separation of Cutset Constraints

Let $\bar{E} = \{e \in F^* : x_e > 0\}$. We use two procedures for identifying violated constraints (5.3) in the hypergraph $H = (V, \bar{E})$ with edge $e \in \bar{E}$ having weight x_e.

The first is a quick heuristic that finds the connected components $C_1, C_2, \ldots, C_k$ of the hypergraph $H = (V, \bar{E})$. If $k > 1$ there are cutsets of zero weight present. If $k < 12$ we combinatorially generate cutset constraints for every partition of the components C_i into two non-empty sets. If $k \geq 12$ we generate only the constraints $x(C_i : V - C_i) \geq 1$ for each $1 \leq i \leq k$.

The other cutset separation procedure is a network flow formulation. For each FST $e \in \bar{E}$, introduce two distinct vertices p_e and q_e. Let $P = \{p_e : e \in \bar{E}\}$ and $Q = \{q_e : e \in \bar{E}\}$. Let $G = (N, A)$ be a directed graph such that:

$$N = V \cup P \cup Q \tag{6.13}$$

$$A = \cup_{e \in \bar{E}} W_e \tag{6.14}$$

$$W_e = \{(t, p_e) : t \in e\} \cup \{(p_e, q_e)\} \cup \{(q_e, t) : t \in e\} \tag{6.15}$$

Let each arc in W_e have capacity x_e. Then any cut $(C : N - C)$ in this digraph with weight less than 1 represents a violation of the cutset constraint defined by $S = C \cap V$. Let $t \in V$ be an arbitrarily chosen sink node. For each $s \in V$ with $s \neq t$ compute the maximum flow in G from s to t. If this flow is less than one, then generate the cutset constraint for $S = C_s \cap V$, where C_s

are the vertices on the same side of the cut as s. At most $n-1$ maximum flow computations on a graph containing at most $n+2m$ vertices and $m+2k$ edges, where $k = \sum_{e \in F^*} |e|$.

This computation can be accelerated considerably with some preprocessing that iteratively collapses all FSTs $e \in F^*$ having $x_e \geq 1$, merging all of the terminals of e into one. After doing this it may happen that FSTs f and g connect the same sets of terminals (i.e. $f = g$ after merging terminals). In this case f and g may be "merged" by deleting g and letting f's new weight be $x_f + x_g$. This may produce new FSTs with weight ≥ 1, in which case the process is repeated. This normally reduces the size of the flow graph considerably, as well as the number of maximum flow computations that must be done on it.

6.2 Separation of Generalized Subtour Elimination Constraints

We use a number of methods to efficiently separate constraints (5.4). First of all we apply two very quick heuristics that locate cycles that are *integral* as well as cycles that are *nearly integral* (i.e. integral except for a single fractional edge). The first procedure uses depth-first traversal over all FSTs $e \in F^*$ for which $x_e = 1$. Any terminal that is visited more than once implies a cycle that can be read off the stack. During this walk, the integral FSTs traversed are recorded, yielding the "integrally connected components". Nearly integral cycles are discovered by checking each fractional FST $e \in F^*$ (i.e. with $0 < x_e < 1$) against each integrally connected component. Any fractional full set that has two or more terminals in common with a single integrally connected component represents a violated subtour that is "nearly integral."

Following Padberg and Wolsey in [20], we can eliminate many terminals from consideration using the following idea, which is adapted from their proposition 2 (i):

Let $b_t = x(\delta(\{t\}))$. Define $f(S) = \sum_{e \in F^*} \max(|e \cap S| - 1, 0) x_e$. If $b_t \leq 1$ and $f(S \cup \{t\}) > |S \cup \{t\}| - 1$, then it can be shown that $f(S) > |S| - 1$. We say that a terminal t such that $b_t \leq 1$ is *uncongested*, or is *congestion-free*. By iteratively eliminating all uncongested terminals, we are left with a core set C of *congested* terminals. We need only examine subsets $S \subseteq C$ for violations.

Let $H = (V, F^*)$ be the weighted hypergraph to separate. The congested sub-hypergraph is then $\hat{H} = (C, \hat{F^*})$ where $\hat{F^*} = \{e \cap C : (e \in F^*) \wedge (|e \cap C| \geq 2) \wedge (x_e > 0)\}$. Let the connected components of $\hat{H}$ be $\hat{H}_1$, $\hat{H}_2$, $\hat{H}_3$, ... Now if $S \subseteq V$ is a violated SEC, then there is some $\hat{H}_j$ such that $S^* = S \cap \hat{H}_j$ is a violated SEC. Thus we may further confine our search to within single connected components. This is just a generalization of proposition 1 of [20] to hypergraphs.

It is a trivial matter to extend this argument to the biconnected components of $\hat{H}$. Note that if a connected component is further divided into several biconnected components, some termi-

nals that were congested may become congestion-free. This can trigger another entire round of decomposition.

Now we turn to the matter of identifying violations within a single component sub-hypergraph $H_j = (C_j, F_j)$ such that no further decomposition of H_j is possible. If $|C_j|$ is small (e.g. 10 or fewer), then it is reasonable to completely enumerate all subsets of C_j.

On larger components, we apply a suite of methods both heuristic and decisive. First we enumerate small-cardinality subsets of C_j. The maximum cardinality checked is a decreasing function of $|C_j|$.

Next we apply a method that heuristically reduces the hypergraph H_j to an undirected graph $\bar{H}_j$ and then apply Padberg and Wolsey's method [20] directly. The reduction is as follows: let $S \in F_j$. Let $k_S = |S| - 1$. Let $\bar{T}_S$ be any set of k_S edges from $\{(s,t) : (s \in S) \wedge (t \in S)\}$ that forms a spanning tree for S. Assign each of these edges weight x_S. Taking the union of the $\bar{T}_S$ we obtain a weighted multigraph. By merging equivalent edges and summing their weights we obtain a weighted graph to which we can apply the method [20].

This method is heuristic in that violations will be detected or not based upon the particular choices of spanning tree for each full set. Lacking a better way to proceed, we arbitrarily choose minimum spanning trees. One interesting possibility that we have not pursued is to choose the trees randomly — possibly repeating the flow procedure on several random reductions. Other promising alternatives choose spanning trees by ranking the terminals of S in order by congestion level, and using either a "chain" of the terminals or a "star" about the most congested terminal.

Previously, the only known algorithm for decisively separating inequalities (5.4) was to find the minimum of the submodular function

$$f(S) = |S| - \sum_{e \in F^*} \max(|e \cap S| - 1, 0) x_e \tag{6.16}$$

using the "ellipsoid" method of Grötschel, Lovasz, and Schrijver [11], [12]. See the presentation in sections III.3.6, III.3.7, I.6.3, and I.4.5 of [19] for more details. A violation exists for any $S \neq \emptyset$ such that $f(S) < 1$. This method was exceedingly slow on congested components larger than about 80 congested terminals.

Maurice Queyranne [22] noticed that maximizing $-f(S)$ can be reduced to an instance of the "selection problem," as defined by Rhys [23] and Balinski [1]. These are equivalent to finding a "maximal closure of a graph," as defined by Picard [21]. All of these problems are solvable by minimum cut on a simple bipartite directed graph.

Let $s_j = 1$ if $j \in S$ and $s_j = 0$ otherwise. Let $z_j = 1 - s_j$ be the complementary $0-1$ variables. Let $b_j = x(\delta(\{t\}))$. b_j is called the "congestion level" of terminal j.

We maximize $-f(S)$ written in terms of the z_j as follows:

$$\begin{aligned}
-f(S) &= \left[\sum_{e \in F^*} \max(|e \cap S| - 1,\, 0)\, x_e\right] - |S| \\
&= \sum_{e \in F^*} \left[\left(\sum_{j \in e} s_j\right) - 1 + \prod_{j \in e}(1 - s_j)\right] x_e - \sum_{j \in V} s_j \\
&= \sum_{e \in F^*} \left[\left(\sum_{j \in e}(1 - z_j)\right) - 1 + \prod_{j \in e} z_j\right] x_e - \sum_{j \in V}(1 - z_j) \\
&= \sum_{e \in F^*} \left[|e| - \sum_{j \in e} z_j - 1 + \prod_{j \in e} z_j\right] x_e - n + \sum_{j \in V} z_j \\
&= \sum_{e \in F^*} (|e| - 1)\, x_e - \sum_{e \in F^*} \left(x_e \sum_{j \in e} z_j\right) + \sum_{e \in F^*} \left(x_e \prod_{j \in e} z_j\right) - n + \sum_{j \in V} z_j \\
&= \sum_{e \in F^*} (|e| - 1)\, x_e - \sum_{j \in V} \left(z_j \sum_{e : j \in e} x_e\right) + \sum_{e \in F^*} \left(x_e \prod_{j \in e} z_j\right) - n + \sum_{j \in V} z_j \\
&= \sum_{e \in F^*} (|e| - 1)\, x_e - \sum_{j \in V} b_j z_j + \sum_{e \in F^*} \left(x_e \prod_{j \in e} z_j\right) - n + \sum_{j \in V} z_j \\
&= \sum_{e \in F^*} \left(x_e \prod_{j \in e} z_j\right) + \sum_{j \in V} (1 - b_j) z_j + \sum_{e \in F^*} (|e| - 1)\, x_e - n
\end{aligned}$$

The last two terms do not depend upon the z_j and are therefore constants that can be ignored. Thus we seek the maximum of a non-linear polynomial over $0-1$ variables, with all non-linear coefficients being positive. This is a selection problem. Note that if the linear term coefficient of z_j is positive, then $z_j = 1$ in any optimal solution. This is precisely the reduction criteria $b_j \leq 1 \Longrightarrow x_j = 0$ mentioned above.

To construct the flow network, introduce one vertex t_j for each terminal j, and one vertex f_e for each FST $e \in F^*$. Introduce two additional distinct vertices s and t as source and sink nodes. Add arc (s, f_e) of capacity x_e for each FST $e \in F^*$. For every terminal j, add arc (t_j, t) of capacity $(1 - b_j)$. For every full set $e \in F^*$, and terminal $j \in e$, add arc (f_e, t_j) of infinite capacity. Let W be a minimum $s - t$ cut in this directed graph such that $s \in W$. Let $z_j = 1$ for every $t_j \in W$ and $z_j = 0$ otherwise. The S resulting from this assignment of the z_j is a maximum of $-f(S)$.

This algorithm finds only constraints that are maximally-violated. These tend to be quite "weak" in terms of their ability to increase the objective value of (6.7), the lower bound. A

constraint normally becomes maximally-violated only by combining the effects of several distinct violations of lesser magnitude. Often times the LP will "dodge" such a "combo constraint" by adjusting some of the connecting material between the actual local violations — which either remain or reappear in later iterations. In such cases it is preferable to have one constraint carefully targeted at each of the distinct local violations. The LP solver then has no place to "cheat."

Some arrangement must therefore be made to ferret out constraints that are more "local" — and therefore more effective at raising the lower bound. We currently use an approach that also addresses the requirement that $S \neq \emptyset$ while minimizing $f(S)$. Let $t \in V$. Define a new function $f_t : (V - \{t\}) \mapsto \Re$ as:

$$f_t(S) = f(S \cup \{t\}) \tag{6.17}$$

Let S_t^* be a minumum of $f_t(S)$. Then $S = S_t^* \cup \{t\}$ is a minumum of $f(S)$ involving terminal t, and is therefore non-empty. Finding a minimum of $f_t(S)$ corresponds to forcing $z_t = 0$ when setting up the flow network for the selection problem. Terminal t can then be removed from consideration and another terminal chosen. Our implementation chooses t to be a least-congested terminal on each iteration. After deletion of terminal t, some terminals that were congested may become congestion-free and the remaining structure can decompose further using the methods mentioned above.

To obtain stronger constraints we *clean up* every solution S by performing all of the decomposition steps: removal of uncongested terminals, connected components, biconnected components. etc. Occasionally this procedure will split a single "maximal" violation into 2 or more components that are lesser violations but much stronger constraints.

7 Implementation Details

In this section we present some implementation details of our branch-and-cut procedure.

7.1 Constraint Pool

A *constraint pool* is used to manage the constraints of the LP relaxation (6.7). This pool is distinct from the LP tableaux given to CPLEX. Each constraint in the pool is represented as a row of coefficients — not as "logical" constraints (i.e. by giving an $S \subseteq V$ representing a subtour). All coefficients for polytope P are small integers so they are stored in the pool as 16-bit integers, making

them quite compact. (This data structure also isolates the rest of our code from the particulars of the LP solver used.)

When adding new constraints to the pool, a hash table is used to determine if the constraint is already present. This hash function operates only on the left hand side of the constraint, not the operator or the right hand side — it is possible that a new constraint is identical to a previous one except for the operator and right hand side. In this case, it may be possible to discard the new entry or tighten the existing constraint by modifying the right hand side.

Each constraint in the pool can be either:

- not present in the LP tableaux,
- present in row k of the LP tableaux, or
- pending addition to the LP tableaux.

The constraint pool is initially *seeded* with the total degree constraint (5.1), all one-terminal cutsets (equation (5.3) with $|S| = 1$), the two-terminal subtours (equation (5.4 with $|S| = 2$), and all incompatibility constraints (5.2) that are not shadowed by one of the two-terminal subtour constraints.

All of the initial constraints are marked "not present in the LP tableaux", except for the total degree constraint and the one-terminal cutset constraints, which are marked "pending addition to the LP tableaux."

The initial LP tableaux is constructed from the "pending" entries of the initial constraint pool, at which point they transition to the "present at row k of the LP tableaux" state.

A single "solve LP over pool" operation involves iteratively solving the present LP tableaux and then scanning the constraint pool for rows that are violated. All such rows are added to the tableaux which is then reoptimized. The process terminates when the LP solution satisfies all constraints in the pool. (No rows are deleted from the LP tableaux during the process.) When we count LPs in the empirical results reported here we are actually counting the number of "solve LP over pool" operations, not the individual LP invocations that make up each "solve LP over pool" operation. Our reasoning is that this is the number of LPs that would be required if all constraints were kept in the LP tableaux all the time — leaving the slack rows out of the LP tableaux is really just a very effective efficiency hack.

After obtaining the optimum over the current pool, the rows of the LP tableaux are examined. If a sufficient number of them are slack (at least 10% when using CPLEX, 1 or more when using

`lp_solve_2.0`) then all such slack rows are deleted from the LP tableaux (but are retained in the constraint pool). The state of each pool entry is updated appropriately.

Any constraints discovered by the separation procedures are added to the pool. If they represent violations they are marked "pending addition to the LP tableaux," otherwise they are marked "not in LP tableaux." All "pending" constraints are then added to the LP tableaux before iterating.

In most cases only a small fraction of the constraints in the pool are "tight" at any one time. Normally there are substantially more constraints in the initial pool than are ever generated via separation. The large majority of the initial constraints are *always* slack. By maintaining only the "active" constraints in the LP tableaux we obtain a substantial speedup in the LP solver — a factor of 10 is typical.

The size of the constraint pool (in units of non-zero coefficients) is limited to some fraction of the largest LP tableaux seen since the start of the computation. Whenever newly added constraints would exceed this size limit, the pool is garbage collected to eliminate constraints that have not been used recently. Constraints are garbage collected in ascending order of *effectiveness*, which is the ratio N/S where N is the number of iterations since the constraint was last seen to be binding, and S is the size of the constraint (in units of non-zero coefficients).

Each constraint in the pool has a reference count that indicates the number of *inactive* nodes for which the constraint was binding. Constraints that are currently in the LP or that have non-zero reference counts are exempted from garbage collection. When a node is deactivated, it is given a list of its binding constraints and their reference counts are incremented. When a node is reactivated, its list of binding constraints have their reference counts decremented. Without such a mechanism, inactive nodes would lose their constraints when the active node garbage collects the pool. This would result in digression of the objective value for the inactive nodes, and termination would not be guaranteed.

7.2 Node Processing

The processing of a node begins by solving the current LP tableaux (with variable bounds set appropriately for the node) using a "solve LP over pool" iteration. The result can be one of the following:

- LP infeasible,
- LP value meets or exceeds best known integer feasible solution, or
- LP optimal solution found, objective value is less than best known integer feasible solution.

If the either of the first two cases happens, then processing of the node terminates and the node is deleted since the subproblem is either IP infeasible or cutoff.

In the final case we delete all slack rows from the LP tableaux and then check the solution to see if it is integer feasible (i.e. integral and connected). If so, the current solution becomes the best known integer feasible solution, processing of the node terminates and the node is deleted. Otherwise, we apply the various separation algorithms as follows:

1. Check for zero-weight cutsets.
2. Check for integer and nearly integer cycles.
3. Decompose solution into its congested components C_1, C_2, C_3, ..., C_k to efficiently find violated SECs.
4. For every C_i with $|C_i| \leq 10$:
   (a) Enumerate *all* subsets S of C_i looking for SEC violations.
   (b) Delete congested component C_i.
5. For each remaining C_i, use the heuristic flow formulation to find SEC violations.
6. For each remaining C_i, enumerate all subsets $S \subseteq C_i$ with $2 \leq |S| \leq k$ to find violated SECs. k is a decreasing function of $|C_i|$.
7. If one or more violated constraints have been generated, delete all remaining congested components and terminate this separation run.
8. For each remaining C_i:
   (a) Let t be a terminal in C_i that is least congested.
   (b) Solve selection problem for C_i under the constraint that $z_t = 0$, yielding $S \subseteq C_i - \{t\}$. Generate SEC for $S \cup \{t\}$ if it is violated by x.
   (c) Delete terminal t from C_i, decomposing it further if possible.
   (d) Iterate until C_i is empty or decomposes completely.
9. If no constraints have been found yet, use the non-heuristic flow formulation to identify violated cutsets.

If one or more violated constraints are discovered by this procedure, they are added to the constraint pool and then the LP tableaux. The processing of the node then iterates back to the "solve LP over pool" operation. If separation discovers no violated constraints, then processing of the node terminates by branching. Two subproblem nodes are created and the current node is deleted.

A remark: the flow formulation for cutsets is placed last for two reasons:

- We have not yet implemented the merging of terminals reduction and this separator is quite slow.
- The cutset constraints are very "weak" (i.e. they produce very little improvement in the objective value).

In all of the runs reported here, the cutset flow separator was actually disabled. The deployment of this separator will probably change once we implement the terminal merging reductions.

7.3 Selection of Branch Variables

A command line argument permits the user to choose between two different methods of choosing fractional variables to branch on. The default method is to choose a variable x_e that maximizes $|\frac{1}{2} - x_e|$, with preference given to full sets $e \in F^*$ that maximize the ratio $c_e/(|e|-1)$. This method does not work well on problems that require lots of branching.

Specifying the **-b** argument enables *best choice* branching. Under this method, the two subproblems for each possible branch variable x_e are solved (or partially solved), yielding objective values Z_e^0 and Z_e^1 for the $x_e = 0$ and $x_e = 1$ cases, respectively. We choose the x_e that maximizes the value of $\min(Z_e^0, Z_e^1)$. Note that we can often avoid solving the $x_e = 1$ subproblem if the value of Z_e^0 is already too low to be a maximum. Note also that the process can terminate immediately if an x_e is found for which both Z_e^0 and Z_e^1 meet or exceed the best known integer feasible solution. We also examine each trial branch solution to see if it happens to represent an integer feasible solution. If branching is needed, this is usually the first place where integer feasible solutions are discovered.

A trial branch subproblem can be "partially solved" for example by limiting the LP iteration count. This process can also be accelerated by saving off the basis from the parent node and restoring it after solving each trial branch subproblem.

When using `lp_solve_2.0`, we first reinvert the matrix, save off the basis, and record the current number of eta vectors. To solve a trial branch subproblem we change the bounds for variable x_i appropriately, and then perform LP pivots until either the optimal solution is reached, the problem

becomes infeasible, division by zero is attempted, or an iteration limit is exceeded. (No reinversion of the matrix is permitted during this process, since that would completely rebuild the eta vectors from scratch.) The final objective value is then noted before restoring the basis and truncating the list of eta vectors back to its original length. This process is extremely fast since the matrix is reinverted only once during the entire branch variable selection process.

When using CPLEX we have less control over the internal state of the LP solver. Currently we do not set an iteration limit during the subproblems, but we do note the initial basis and restore it after solving each trial branch subproblem. This is still substantially slower than our hacks to `lp_solve_2.0`. However, for problems that require lots of branching it reduces the overall computation time significantly. There may be some tricks that we can do to the CPLEX version that will approach the efficiency of our `lp_solve_2.0` hacks.

7.4 Node Selection

We use "best node first" node selection strategy. That is, we always choose the outstanding node having the lowest objective value to process next. Although this can require more memory than the "most recent node first" strategy, it tends to raise the lower bound more rapidly. This problem formulation seems to be "tight" enough to avoid huge numbers of nodes, however.

We do not record an LP basis for each node, only two bit vectors indicating which variables are fixed, and if so the value they are fixed at — 0 or 1. Saving an entire LP basis for each node consumes substantially more memory. The loss of speed this causes appears to be a negligible percentage of the run time.

7.5 Node Preemption

Earlier versions of this program demonstrated some undesirable behavior during branching. Processing of a node would terminate only under the following circumstances:

- an integer feasible solution is encountered,
- the subproblem is (integer or fractionally) infeasible,
- a fractional solution with no violations is reached.

It was not unusual for a node A to launch off into a very lengthy sequence of optimize/separate iterations, pushing the node's objective value Z_A way up at great computational cost until one of the above three criteria are met. Often the subsequent processing of another node B would discover

an integer feasible solution with objective Z_B such that $Z_A > Z_B$, thereby cutting off node A on which we expended so much work.

The current algorithm prevents this using a *node preemption* method. Under this scheme, processing of a node A can be *suspended* once there is another node B for which $Z_A > Z_B$. When this happens, we say that node A is *preempted* by node B.

After generating some good constraints it is not unusual to process a number of nodes in turn (each preempted by the next) before encountering a node that is not preemptible — at which time we run the separation procedures. This has the effect of simply re-solving the LP for each of these nodes using the recently discovered constraints.

8 STHGP(n) — The Spanning Tree in Hypergraph Polytope

In this section we define STHGP(n), the spanning tree in hypergraph polytope. We then prove five important properties of STHGP(n):

1. the validity of equation (5.1),
2. its dimensionality,
3. every spanning tree is an extreme point,
4. the non-negativity constraints define facets for $n \geq 4$,
5. the subtour constraints (5.4) define facets for $n \geq 3$.

Finally, we list all of the facets of STHGP(n) for $2 \leq n \leq 4$.

8.1 Definitions

We define $\mathcal{K}_n = (V, E)$ such that $|V| = n$ and $E = \{e \subseteq V : |e| \geq 2\}$. Let $m = |E| = 2^n - n - 1$.

To every subhypergraph $H' = (V, E')$ of $\mathcal{K}_n = (V, E)$, we associate an incidence vector $x \in \{0, 1\}^{|E|}$ defined by $x_e = 1$ if $e \in E'$ and 0 otherwise. Let $\mathrm{ST}_n \subset \{0, 1\}^{2^n - n - 1}$ denote the set of incidence vectors of spanning trees of $\mathcal{K}_n$.

Define $\mathrm{STHGP}(n) = \mathrm{conv}(\mathrm{ST}_n)$.

8.2 Theoretical Results

Theorem 8.1 : *Let $n \geq 2$. Let $(V, E) = \mathcal{K}_n$. Let $x \in \mathrm{ST}_n$. Then*

$$\sum_{e \in E} (|e| - 1)\, x_e = n - 1. \tag{8.18}$$

Proof: *From [4] we know that a hypergraph is acyclic if*

$$\sum_{e \in E} (|e| - 1) = n - p,$$

where p is the number of connected components. x is a spanning tree and therefore acyclic with $p = 1$. □

Theorem 8.1 gives an equation satisfied by all $x \in \mathrm{ST}_n$. We now show there are no other such equations.

Lemma 8.2 : *Let $n \geq 3$. Let $(V, E) = \mathcal{K}_n$. Let $m = |E| = 2^n - n - 1$. Let $c\,x = b$ be any equation satisfied by every $x \in \mathrm{ST}_n$. Then there is an α such that $b = \alpha(n-1)$ and $c_e = \alpha(|e| - 1)$ for every $e \in E$.*

Proof: *Let c and b be as stated. Let $e_1, e_2 \in E$ such that $|e_1| = |e_2| = 2$. We will first show that $c_{e_1} = c_{e_2}$. Assume $c_{e_1} \neq c_{e_2}$. Let $S \subseteq V$ be a cut of V crossed by both e_1 and e_2, i.e. if $f(x) \equiv (x \cap S \neq \emptyset) \wedge (x \cap (V \setminus S) \neq \emptyset)$ then we require $f(e_1) \wedge f(e_2)$. A suitable cut S always exists when $n \geq 3$. Construct spanning trees $S_1 = (S, E_1)$ and $S_2 = (V - S, E_2)$ for each side of the cut using only 2-edges. We can now construct spanning trees $T^1 = (V, E_1 \cup \{e_1\} \cup E_2)$ and $T^2 = (V, E_1 \cup \{e_2\} \cup E_2)$ for V having incidence vectors x^1 and x^2, respectively. We have $c\,x^1 = b$ and $c\,x^2 = b \Longrightarrow c\,x^1 - c\,x^2 = b - b \Longrightarrow c_{e_1} - c_{e_2} = 0$, a contradiction. Let α be the common coefficient of all 2-edges. Certainly $c_e = \alpha(|e| - 1)$ holds for every 2-edge $e \in E$. We deduce that $b = \alpha(n-1)$ by noting that we can construct spanning trees for V entirely out of 2-edges ($n - 1$ of them).*

Let $T^3 = (V, E_3)$ be a spanning tree with $e \in E_3$ such that $3 \leq |e| = k$. Let x^3 be the incidence vector of T^3. We can construct a new spanning tree T^4 by replacing e with any spanning tree constructed using only 2-edges from $E(e)_2$ ($k - 1$ of them). Let x^4 be the incidence vector of T^4. We have $c\,x^3 = b$ and $c\,x^4 = b \Longrightarrow c\,x^3 - c\,x^4 = b - b \Longrightarrow c_e - (k-1)\alpha = 0 \Longrightarrow c_e = \alpha(|e| - 1)$, completing the proof. □

Theorem 8.3 :

$$\dim(\mathrm{STHGP}(n)) = 2^n - n - 2 \quad \textit{for } n \geq 2. \tag{8.19}$$

Proof: *Suppose* $n = 2$. $|\mathrm{ST}_2| = 1$ *so that* $\mathrm{STHGP}(2)$ *is a single point with dimension* 0 *and the theorem holds.*

Now suppose $n \geq 3$. *Let* $m = 2^n - n - 1$. $x \in \mathrm{ST}_n \Longrightarrow x \in \Re^m$. *Theorem 8.1 gives one equation satisfied by every* $x \in \mathrm{ST}_n$. *Lemma 8.2 shows that there are no other such equations. Therefore* $\dim(\mathrm{STHGP}(n)) = m - 1 = 2^n - n - 2$. □

Theorem 8.4 *: Every* $x \in \mathrm{ST}_n$ *is an extreme point of* $\mathrm{STHGP}(n)$.

Proof: *Let* $x \in \mathrm{ST}_n$. *Let* $c \in \Re^m$, *such that*

$$c_e = \begin{cases} 3(|e| - 1), & \text{if } x_e = 1; \\ |e| - 1, & \text{otherwise.} \end{cases}$$

By theorem 8.1 we have $cx = 3n - 3$. *Let* $y \in \mathrm{ST}_n$ *and* $y \neq x$. *We must have* $cy \leq cx - 2 = 3n - 5$. *Therefore, x lies on one side of the hyperplane* $cx = 3n - 4$ *and y lies on the other side.* □

Corollary: *Let* $x \in \mathrm{ST}_n$. *x cannot be expressed as a convex combination of the elements of* $\mathrm{ST}_n \setminus \{x\}$.

To prove that the non-negativity constraints are facet-defining, we will need two technical lemmas:

Lemma 8.5 *: Let* $n \geq 4$. *Let* $(V, E) = \mathcal{K}_n$. *Let* $m = |E| = 2^n - n - 1$. *Let* $e, e_1, e_2 \in E$ *be distinct edges such that* $|e_1| = |e_2| = 2$. *Then there is an* $S \subset V$ *such that* $1 \leq |S| < n$ *with the following properties:*

1. $e_1, e_2 \in (S : V - S)$.
2. *There exist spanning trees* $S_1 = (S, E_1)$ *and* $S_2 = (V - S, E_2)$ *such that* $e \notin E_1$ *and* $e \notin E_2$.

Proof: **case 1**. *Suppose* $|e| \geq 3$. *We can satisfy property 2 by constructing spanning trees for S and* $V - S$ *using only* $2-$*edges. If* $e_1 \cap e_2 = \{v_1\}$, *it suffices to choose* $S = \{v_1\}$. *Otherwise choose any* $v_1 \in e_1$ *and* $v_2 \in e_2$ *and let* $S = \{v_1, v_2\}$.

case 2. *Suppose* $|e| = 2$. *There are 3 major subcases to consider.*

case 2a. *Suppose* $n = 4$. *If* $e_1 \cap e_2 = \{v_1\}$, *it suffices to choose* $S = \{v_1\}$, *which obviously satisfies property 1. It also satisfies property 2 since the spanning tree for S contains no edges, and the spanning tree for the other 3 vertices* $V - S$ *can be any 2 of the 3 edges in* $E(V - S)_2$ *— at most 1 of these edges can be e. Otherwise, we have* $e_1 \cap e_2 = \emptyset$, *which is isomorphic to* $e_1 = \{v_1, v_2\}$, $e_2 = \{v_3, v_4\}$, $e = \{v_1, v_3\}$, $S = \{v_1, v_4\}$, $E_1 = \{\{v_1, v_4\}\}$, $E_2 = \{\{v_2, v_3\}\}$.

case 2b. *Suppose* $n = 5$. *Suppose* $e_1 \cap e_2 = \{v_1\}$. *If* $v_1 \in e$ *then* $S = \{v_1\}$ *satisfies both properties for reasons similar to case 2a. Now suppose* $e_1 \cap e_2 = \emptyset$. *Let* $v_1 \in V$ *be the unique vertex that is not contained in either* e_1 *or* e_2. *If* $v_1 \notin e$, *we can construct the* $n = 4$ *case for* $V \setminus \{v_1\}$ *and then arbitrarily add* v_1 *to* S *using any* 2*-edge* $\{v_1, v_2\}$ *with* $v_2 \in S$ *chosen arbitrarily. Otherwise, we have* $v_1 \in e$. *Let* v_2 *be the other vertex in* e. *Then either* $v_2 \in e_1$ *or* $v_2 \in e_2$. *Assume w.l.o.g. that* $e_1 = \{v_2, v_3\}$ *and* $e_2 = \{v_4, v_5\}$. *Then* $S = \{v_3, v_4\}$ *satisfies both properties, since we can take* $E_1 = \{\{v_3, v_4\}\}$ *and* $E_2 = \{\{v_1, v_5\}, \{v_2, v_5\}\}$.

case 2c. *Suppose* $n \geq 6$. *We can assure* $|S| \geq 3$ *and* $|V - S| \geq 3$ *so that the spanning trees of property 2 each consist of single hyperedges. If* $e_1 \cap e_2 = \{v\}$ *then we can choose* v *plus two other members of* $V \setminus e_1 \setminus e_2$. *Otherwise, we pick* $v_1 \in e_1$, $v_2 \in e_2$, $v_3 \in V \setminus e_1 \setminus e_2$ *and let* $S = \{v_1, v_2, v_3\}$. □

Lemma 8.6 : *Let* $n \geq 4$. *Let* $(V, E) = \mathcal{K}_n$. *Let* $m = |E| = 2^n - n - 1$. *Let* $e \in E$. *Let* $F = \{x \in \mathrm{ST}_n : x_e = 0\}$. *Let* $c\,x = b$ *be any equation satisfied by every* $x \in F$. *Then there exists an* α *such that* $b = \alpha(n-1)$ *and* $c_{e'} = \alpha(|e'| - 1)$ *for all* $e' \in E$, $e' \neq e$.

Proof: *Let* e_1 *and* e_2 *be* 2*-edges distinct from* e. *By lemma 8.5 there is a cut* $(S, V - S)$, $1 \leq |S| < |V|$ *crossed by both* e_1 *and* e_2. *Also by lemma 8.5 there exist spanning trees* $S_1 = (S, E_1)$ *and* $S_2 = (V - S, E_2)$ *for* S *and* $V - S$ *respectively, such that* $e \notin E_1$ *and* $e \notin E_2$. *Then* $T^1 = (V, E_1 \cup \{e_1\} \cup E_2)$ *and* $T^2 = (V, E_1 \cup \{e_2\} \cup E_2)$ *are spanning trees for* V *that do not contain edge* e. *Let* x^1 *be the incidence vector of* T^1 *and* x^2 *the incidence vector of* T^2. $x^1, x^2 \in F$ *since* $x^1_e = x^2_e = 0$ *and so must satisfy* $c\,x^1 = b$ *and* $c\,x^2 = b \Longrightarrow c\,x^1 - c\,x^2 = b - b \Longrightarrow c_{e_1} - c_{e_2} = 0$. *Every* 2*-edge* $e' \neq e$ *therefore has the same coefficient* $c_{e'} = \alpha$.

Let $T^3 = (V, E_3)$ *be a spanning tree with incidence vector* $x^3 \in F$. *Suppose* $e' \in E_3$ *such that* $3 \leq |e'| = k$. *Construct a new spanning tree* T^4 *by replacing edge* e' *with a spanning tree constructed using only* 2*-edges* ($k-1$ *of them) from* $E(e')_2$. *Let* x^4 *be the incidence vector of* T^4. *So* $x^3 \in F$ *and by construction* $x^4_e = 0 \Longrightarrow x^4 \in F$. *This implies* $c\,x^3 = b$ *and* $c\,x^4 = b \Longrightarrow c\,x^3 - c\,x^4 = b - b \Longrightarrow c_{e'} - \alpha(|e'| - 1) = 0$. *Therefore the coefficient* $c_{e'} = \alpha(|e'| - 1)$ *for all* $e' \in E$, $e' \neq e$. *We must have* $b = (n-1)\alpha$, *since a spanning tree for* V *can be always be constructed using exactly* $n - 1$ 2*-edges from* $E(V)_2 \setminus \{e\}$. □

Theorem 8.7 : *Let* $n \geq 4$. *Let* $(V, E) = \mathcal{K}_n$. *Let* $m = |E| = 2^n - n - 1$. *Let* $e \in E$. *Then the inequality* $x_e \geq 0$ *defines a facet of* $\mathrm{STHGP}(n)$.

Proof: *First note that* $x_e \geq 0$ *is satisfied by every* $x \in \mathrm{ST}_n$ *and is therefore a valid inequality. Let* $F = \{x \in \mathrm{ST}_n : x_e = 0\}$. *Let* $c\,x = b$ *be any equation that is satisfied by every* $x \in F$. *By lemma 8.6*

we know that equation $cx = b$ is such that $b = \alpha(n-1)$ and $c_{e'} = \alpha(|e'|-1)$ for all $e' \in E$, $e' \neq e$. Equation $cx = b$ can therefore be obtained by taking α times equation (8.18) plus $c_e - \alpha(|e|-1)$ times equation $x_e = 0$. The set F therefore has dimension $m - 2 = \dim(\mathrm{STHGP}(n)) - 1$, proving that $x_e \geq 0$ is facet-defining. □

In order to prove the subtour constraints are facet-defining, we need two technical lemmas.

Lemma 8.8 *: Let $n \geq 3$. Let $(V,E) = \mathcal{K}_n$. Let $m = |E| = 2^n - n - 1$. Let $S \subset V$ such that $|S| \geq 2$. Let $F = \{x \in \mathrm{ST}_n : \sum_{e \in E} \min(|e \cap S| - 1, 0)x_e = |S| - 1$. Let $cx = b$ be any equation satisfied by every $x \in F$. Then:*

1. *$c_{e_1} = c_{e_2}$ for all $e_1, e_2 \in E(S)_2$,*
2. *$c_{e_1} = c_{e_2}$ for all $e_1, e_2 \in \delta(S)_2$,*
3. *$c_{e_1} = c_{e_2}$ for all $e_1, e_2 \in E(V-S)_2$ and*
4. *$c_{e_1} = c_{e_2}$ for all $e_1 \in \delta(S)_2$ and all $e_2 \in E(V-S)_2$.*

Proof: *First we prove part 1. This is vacuously true if $|S| = 2$ since there is only one such edge. Suppose $|S| \geq 3$. Let $e_1, e_2 \in E(S)_2$ such that $c_{e_1} \neq c_{e_2}$. Since $|S| \geq 3$ there exists a cut $(S', S - S')$ crossed by both e_1 and e_2. Let $S^1 = (S', E_1)$ and $S^2 = (S - S', E_2)$ be spanning trees constructed using only 2-edges from $E(S')_2$ and $E(S - S')_2$, respectively. Let e_3 be any 2-edge in $\delta(S)_2$. Construct spanning tree $S^3 = (V - S, E_3)$ using 2-edges from $E(V-S)_2$. Then $T^1 = (V, E_1 \cup \{e_1\} \cup E_2 \cup \{e_3\} \cup E_3)$ and $T^2 = (V, E_1 \cup \{e_2\} \cup E_2 \cup \{e_3\} \cup E_3)$ are spanning trees for V. Let x^1 be the incidence vector for T^1 and x^2 be the incidence vector for T^2. We have $x^1, x^2 \in F$, since they each have exactly $|S| - 1$ 2-edges taken from $E(S)_2$. Therefore, $cx^1 = b$ and $cx^2 = b \Longrightarrow cx^1 - cx^2 = b - b \Longrightarrow c_{e_1} - c_{e_2} = 0$, a contradiction.*

Now we prove part 2. Let $S' = V - S$. This is vacuously true if $|S'| \leq 1$, so we assume that $|S'| \geq 2$. Let $e_1, e_2 \in \delta(S)_2$ such that $c_{e_1} \neq c_{e_2}$. Let $S^1 = (S, E_1)$ be a spanning tree constructed entirely out of 2-edges from $E(S)_2$. Let $S^2 = (S', E_2)$ be a spanning tree for S'. Then $T^1 = (V, E_1 \cup \{e_1\} \cup E_2)$ and $T^2 = (V, E_1 \cup \{e_2\} \cup E_2)$ are spanning trees for V. Let x^1 be the incidence vector for T^1 and x^2 be the incidence vector for T^2. Then $x^1, x^2 \in F$, since they each contain exactly $|S| - 1$ 2-edges from $E(S)_2$. Therefore $cx^1 = b$ and $cx^2 = b \Longrightarrow cx^1 - cx^2 = b - b \Longrightarrow c_{e_1} - c_{e_2} = 0$, a contradiction.

Now we prove part 3. Let $S' = V - S$. This is vacuously true if $|S'| \leq 2$, so we assume that $|S'| \geq 3$. Let $e_1, e_2 \in E(S')_2$ such that $c_{e_1} \neq c_{e_2}$. Let $S^1 = (S, E_1)$ be a spanning tree

constructed entirely out of 2-edges from $E(S)_2$. Let e_3 be any 2-edge from $\delta(S)_2$. There exists a cut $(U, S' - U)$ of S' crossed by both e_1 and e_2. Let $S^2 = (U, E_2)$ be a spanning tree for U, and let $S^3 = (S' - U, E_3)$ be a spanning tree for $S' - U$. Then $T^1 = (V, E_1 \cup \{e_3\} \cup E_2 \cup \{e_1\} \cup E_3)$ and $T^2 = (V, E_1 \cup \{e_3\} \cup E_2 \cup \{e_2\} \cup E_3)$ are spanning trees for V. Let x^1 be the incidence vector for T^1 and x^2 be the incidence vector for T^2. Then $x^1\, x^2 \in F$, since they each contain exactly $|S| - 1$ 2-edges from $E(S)_2$. Therefore $c\, x^1 = b$ and $c\, x^2 = b \Longrightarrow c\, x^1 - c\, x^2 = b - b \Longrightarrow c_{e_1} - c_{e_2} = 0$, a contradiction.

Finally we prove part 4. Let $S' = V - S$. This is vacuously true if $|S`| \leq 1$, so we assume $|S'| \geq 2$. Let $e'_2 = \{v_2, v_3\} \in E(S')_2$. Let $v_1 \in S$. Let $e'_1 = \{v_1, v_2\}$, so that $e'_1 \in \delta(S)_2$. Suppose $c_{e'_1} \neq c_{e'_2}$. Let $e'_3 = \{v_1, v_3\}$. Let $(U, S' - U)$ be a cut that is crossed by e'_2. Assume w.l.o.g. that $v_2 \in U$. Let $S^1 = (S, E_1)$ be a spanning tree for S containing only 2-edges from $E(S)_2$. Let $S^2 = (U, E_2)$ be a spanning tree for U. Let $S^3 = (S' - U, E_3)$ be a spanning tree for $S' - U$. Then $T^1 = (V, E_1 \cup E_2 \cup E_3 \cup \{e'_3\} \cup \{e'_1\})$ and $T^1 = (V, E_1 \cup E_2 \cup E_3 \cup \{e'_3\} \cup \{e'_2\})$ are spanning trees for V. Let x^1 be the incidence vector for T^1 and x^2 be the incidence vector for T^2. Then $x^1, x^2 \in F$, since they both contain exactly $|S| - 1$ 2-edges from $E(S)_2$. Therefore $c\, x^1 = b$ and $c\, x^2 = b \Longrightarrow c\, x^1 - c\, x^2 = b - b \Longrightarrow c_{e'_1} - c_{e'_2} = 0$, contradicting the assumption that $c_{e'_1} \neq c_{e'_2}$. Thus we have proven the equality $c_{e'_1} = c_{e'_2}$ for a single pair of edges $e'_1 \in \delta(S)_2$ and $e'_2 \in E(V_S)_2$. Using the results of parts 2 and 3 and the transitivity of equality, this result holds for all $e_1 \in \delta(S)_2$ and $e_2 \in E(V - S)_2$. □

Lemma 8.9 *: Let $n \geq 3$. Let $(V, E) = \mathcal{K}_n$. Let $m = |E| = 2^n - n - 1$. Let $S \subset V$ such that $|S| \geq 2$. Let $F = \{x \in \mathrm{ST}_n : \sum_{e \in E} \min(|e \cap S| - 1, 0) x_e = |S| - 1\}$. Let $c\, x = b$ be any equation satisfied by every $x \in F$. Then there exist α and β such that $b = \alpha(|S| - 1) + \beta(|V| - |S|)$ and $c_e = \alpha \min(|e \cap S| - 1, 0) + \beta(|e| - 1 - \min(|e \cap S| - 1, 0))$ for all $e \in E$.*

Proof: *By lemma 8.8, $c_e = \alpha$ for every 2-edge $e \in E(S)_2$ and $c_e = \beta$ for every 2-edge $e \in E(V)_2 \setminus E(S)_2$. Let $T^1 = (V, E_1)$ be any spanning tree with incidence vector x^1 such that $x^1 \in F$. This implies that $c\, x^1 = b$. Let $e \in E_1$ be any edge of this tree. Suppose $|e \cap S| \leq 1$. Then e can be replaced with any spanning tree constructed using only 2-edges in $E(e)_2$ ($|e| - 1$ of them, each having weight β). The result will be a spanning tree T^2 with incidence vector x^2 and $x^2 \in F$. Therefore $c\, x^1 = b$ and $c\, x^2 = b \Longrightarrow c\, x^1 - c\, x^2 = b - b \Longrightarrow c_e - (|e| - 1)\beta = 0 \Longrightarrow c_e = (|e| - 1)\beta$.*

Suppose $e \subseteq S$. Then e can be replaced with any spanning tree constructed using only 2-edges in $E(e)_2$ ($|e| - 1$ of them, each having weight α). The result will be a spanning tree T^3 with incidence vector $x^3 \in F$. Therefore $c\, x^1 = b$ and $c\, x^3 = b \Longrightarrow c\, x^1 - c\, x^3 = b - b \Longrightarrow c_e - (|e| - 1)\alpha = 0 \Longrightarrow$

$c_e = (|e| - 1)\alpha$.

Finally suppose $2 \leq |e \cap S| < |e|$. *We construct a spanning tree* $T^4 = (V, E_4)$ *with incidence vector* x^4 *by replacing* e *with a spanning tree of 2-edges from* $E(e)_2$ *(*$|e| - 1$ *of them). But in order to force* $x^4 \in F$ *we must choose* $|e \cap S| - 1$ *of these edges from* $E(e \cap S)_2$ *and the remaining* $|e| - |e \cap S|$ *of them from* $E(e)_2 \setminus E(S)_2$. *Therefore* $c\,x^1 = b$ *and* $c\,x^4 = b \Longrightarrow c\,x^1 - c\,x^4 = b - b \Longrightarrow$ $c_e - (|e \cap S| - 1)\alpha - (|e| - |e \cap S|)\beta \Longrightarrow c_e = (|e \cap S| - 1)\alpha + (|e| - |e \cap S|)\beta$.

It is easy to verify that the equation $c_e = \alpha \min(|e \cap S| - 1,\, 0) + \beta(|e| - 1 - \min(|e \cap S| - 1,\, 0))$ *generalizes all three of the above cases.*

If we reduce all edges to 2-edges in this fashion, we will have exactly $|S| - 1$ *2-edges in* $E(S)_2$ *and exactly* $|V| - |S|$ *2-edges in* $E(V)_2 \setminus E(S)_2$. *Thus we must have* $b = \alpha(|S| - 1) + \beta(|V| - |S|)$.

□

Theorem 8.10 *: Let* $n \geq 3$. *Let* $(V, E) = \mathcal{K}_n$. *Let* $m = |E| = 2^n - n - 1$. *Let* $S \subseteq V$ *such that* $2 \leq |S| < n$. *The inequality*

$$\sum_{e \in E} \min(|e \cap S| - 1,\, 0)\, x_e \leq |S| - 1 \tag{8.20}$$

defines a facet of $\mathrm{STHGP}(n)$.

Proof: *First note that (8.20) is a valid inequality, since if* $\sum_{e \in E} \min(|e \cap S| - 1, 0)\, x_e > |S| - 1$ *we have a cycle residing entirely within* S, *a contradiction since every spanning tree* $x \in \mathrm{ST}_n$ *is acyclic. Let* F *be the set of all* $x \in \mathrm{ST}_n$ *that satisfy the equation*

$$\sum_{e \in E} \min(|e \cap S| - 1,\, 0)\, x_e = |S| - 1. \tag{8.21}$$

Let $c\,x = b$ *be any equation that is satisfied by every* $x \in F$. *By lemma 8.9 we know that equation* $c\,x = b$ *can be written in the form:* $b = \alpha(|S| - 1) + \beta(|V| - |S|)$ *and* $c_e = \alpha \min(|e \cap S| - 1,\, 0) +$ $\beta(|e| - 1 - \min(|e \cap S| - 1,\, 0))$ *for all* $e \in E$. *We can therefore obtain this equation by taking* β *times equation (8.18) plus* $(\alpha - \beta)$ *times equation (8.21). The set* F *therefore has dimension* $m - 2 = \dim(\mathrm{STHGP}(n)) - 1$, *proving that inequality (8.20) is facet-defining.* □

8.3 Facets of $\mathrm{STHGP}(n)$ for $2 \leq n \leq 4$

In this section we assume for the sake of concreteness that $V = \{0, 1, \ldots, n - 1\}$. Suppose edge $e = \{1, 3, 5\}$. For simplicity we write x_e as x_{135}.

Let $n = 2$. Then $\mathrm{STHGP}(n)$ consists of the single point $x_{01} = 1$. There are no facets.

Let $n = 3$. Then the 4 hyperedges are indicated (in sequence) by variables x_{01}, x_{02}, x_{12}, and x_{012}. STHGP(3) has 4 extreme points: $(0, 1, 1, 0)$, $(1, 0, 1, 0)$, $(1, 1, 0, 0)$ and $(0, 0, 0, 1)$. There are 4 facets: 3 of these are the 2-terminal subtour constraints, and the last is the non-negativity constraint $x_{0123} \geq 0$.

Let $n = 4$. Then the 11 hyperedges are indicated (in sequence) by variables x_{01}, x_{02}, x_{03}, x_{12}, x_{13}, x_{23}, x_{012}, x_{013}, x_{023}, x_{123} and x_{0123}. STHGP(4) has 29 extreme points and 22 facets:

- 6 two-terminal subtours
- 4 three-terminal subtours
- 11 non-negativity constraints
- $x_{012} + x_{013} + x_{023} + x_{123} + x_{0123} \leq 1$

The final facet is essentially a maximal clique of the incompatibility graph.

The facets of STHGP(5) are known, but have not yet been fully classified. Let $n = 5$. There are 26 hyperedges, indicated by similar variables. STHGP(5) has 311 extreme points and 172 facets:

- 10 two-terminal subtours
- 10 three-terminal subtours
- 5 four-terminal subtours
- 26 non-negativity constraints
- 121 unidentified inequalities

It is hoped that new general classes of facet-defining inequalities can be identified by analysis of the unidentified inequalities.

9 Empirical Results

A large number of problems were attempted (1251), all of which were solved to optimality. Since we currently have no heuristic for generating good feasible integer solutions, early termination of the algorithm implies no solution at all — unless branching discovers a feasible integer solution.

All computations reported here were performed on a 4 processor Sparc 20 with 256 megabytes of memory. The code runs sequentially on a single CPU, but using "poor man's parallelism" we can handle up to 4 runs concurrently. CPLEX version 4.0 was used to solve all the LPs. Because of

its high cost, FST pruning was disabled for all runs reported here. Note that the flow-based cutset separation algorithm was also disabled.

Solutions are given for several problem sets from the literature, including those of Soukup and Chow [26], and and all of the problems from Beasley's OR-library [2], [3] having 1000 or fewer terminals. Because the OR-library problems jump directly from 100 points to 250 we included 15 random problem instances each of 110, 120, ..., 240 points to fill in the gaps in our plots. We also included a more thorough study of smaller random problem instances: 100 problems each of sizes $15, 20, 25, 30, 35, 40, 45$ and 50 points. In all the study contains 1251 problems ranging in size from 3 to 1000 terminals — all of which were solved to proven optimality.

Certificates for these solutions (in the form of lists of Steiner point coordinates) are available on the web at "`http://www.s3i.com./~warme`", as is the postscript version of this paper.

All problems were solved using the "`-b`" argument, which enables *best choice* branching.

Figure 5 contains a plot of phase 1 execution time as a function of number of terminals for all problems in the study. This plot includes empirical upper and lower bounds, both of which are cubic functions. As can be seen, these bounds are rather tight for this data set.

Figure 6 contains a plot of the branch-and-cut (phase 2) execution time as a function of the number of terminals for all problems in the study. Figure 7 plots the total execution time. Figure 8 plots the total execution time as a function of the number of FSTs.

Figures 9, 10 and 11 plot the number of FSTs versus the number of terminals. As can be seen from these plots, the number of FSTs seems to rise linearly with the number of terminals. We have included linear upper and lower bounds in all of these plots for reference purposes.

Finally, we tabulate the specific computational details of each OR-library problem instance solved. In tables 1 through 5, N is the number of terminals, M is the number of FSTs. Z is the length of the optimal RSMT. The "Z Root" column is the final LP objective value of the root node. The "% Gap" column is: $100(\text{Z} - \text{Z Root})/\text{Z}$. "Nds" is the number of branch-and-bound nodes required — 1 node indicates that optimality was proven at the root node (without branching). "LPs" is the total number of "solve LP over pool" operations of the form (6.12) that were required. The "IRow" column is the initial number of constraints in the constraint pool. The "RTight" column is the number of tight constraints in the final LP tableaux for the root node. Finally, figure 12 presents the solution for a 1000 point problem (instance 1 from the OR-library `estein1000.txt` file).

We have also used this FST concatenation algorithm on Euclidean FSTs generated by the FST generator of Winter and Zachariasen [28] with excellent results — all OR-library instances of 1000

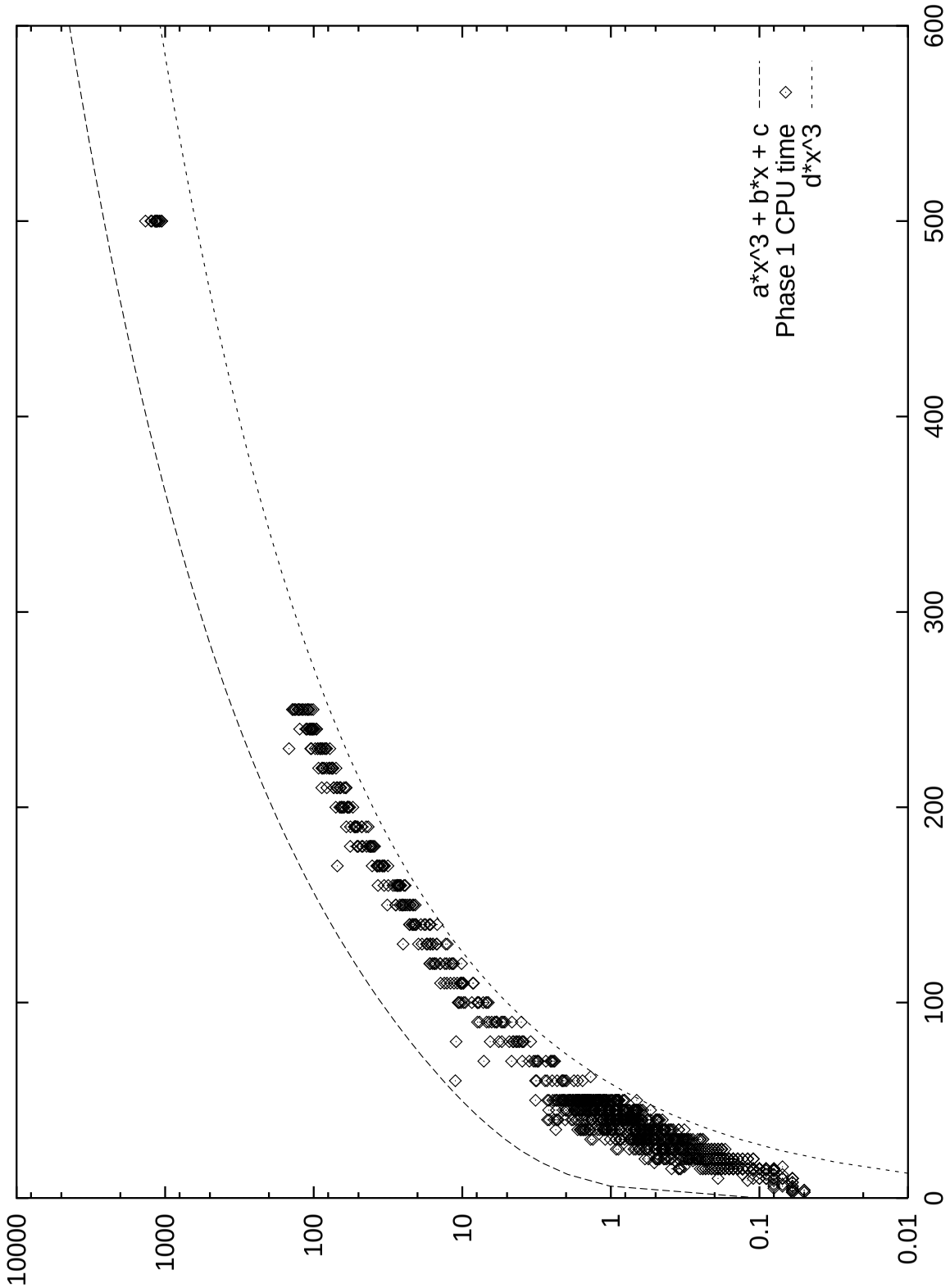

Figure 5: Plot of phase 1 CPU time vs. number of terminals.

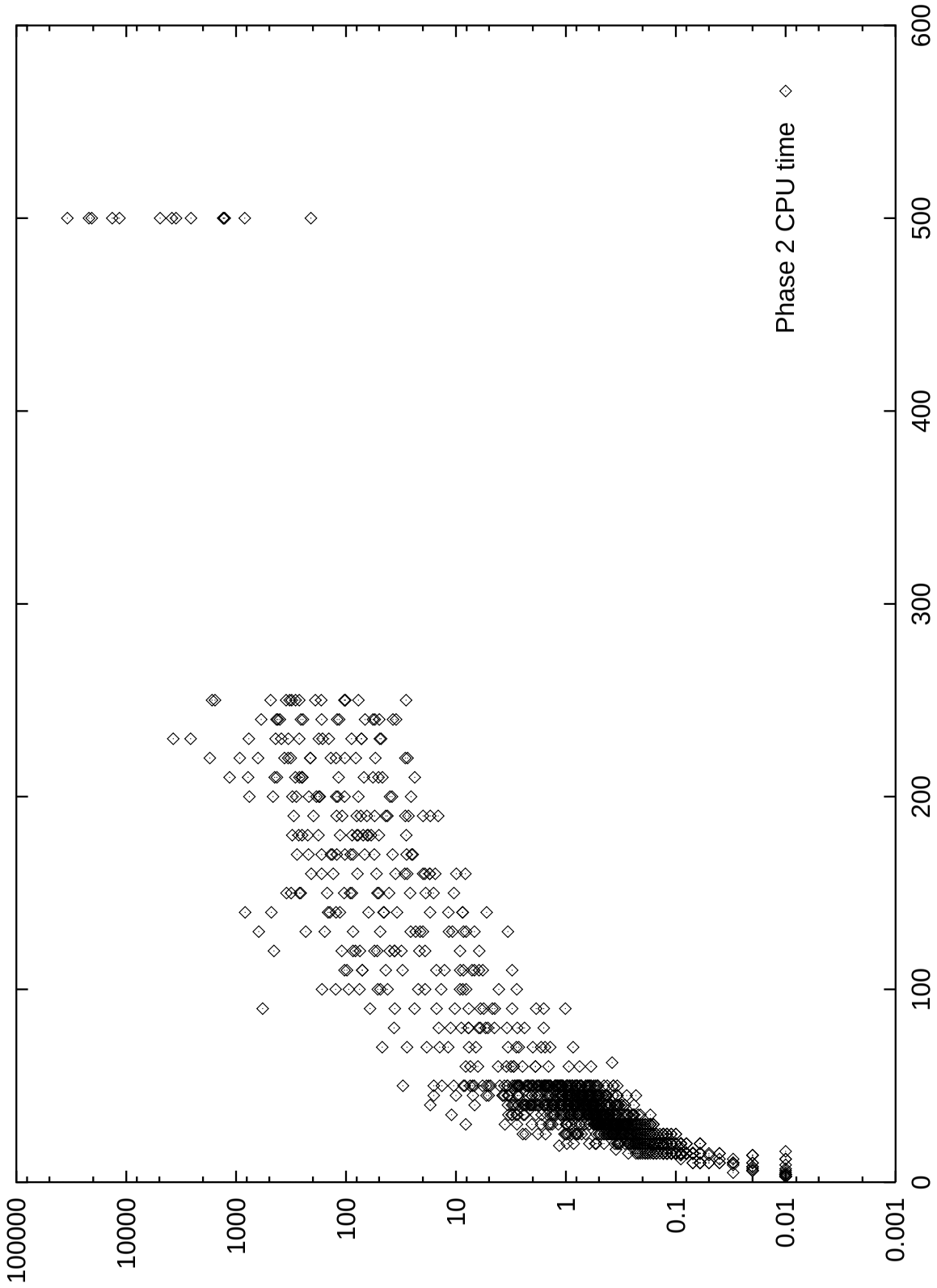

Figure 6: Plot of phase 2 CPU time vs. number of terminals.

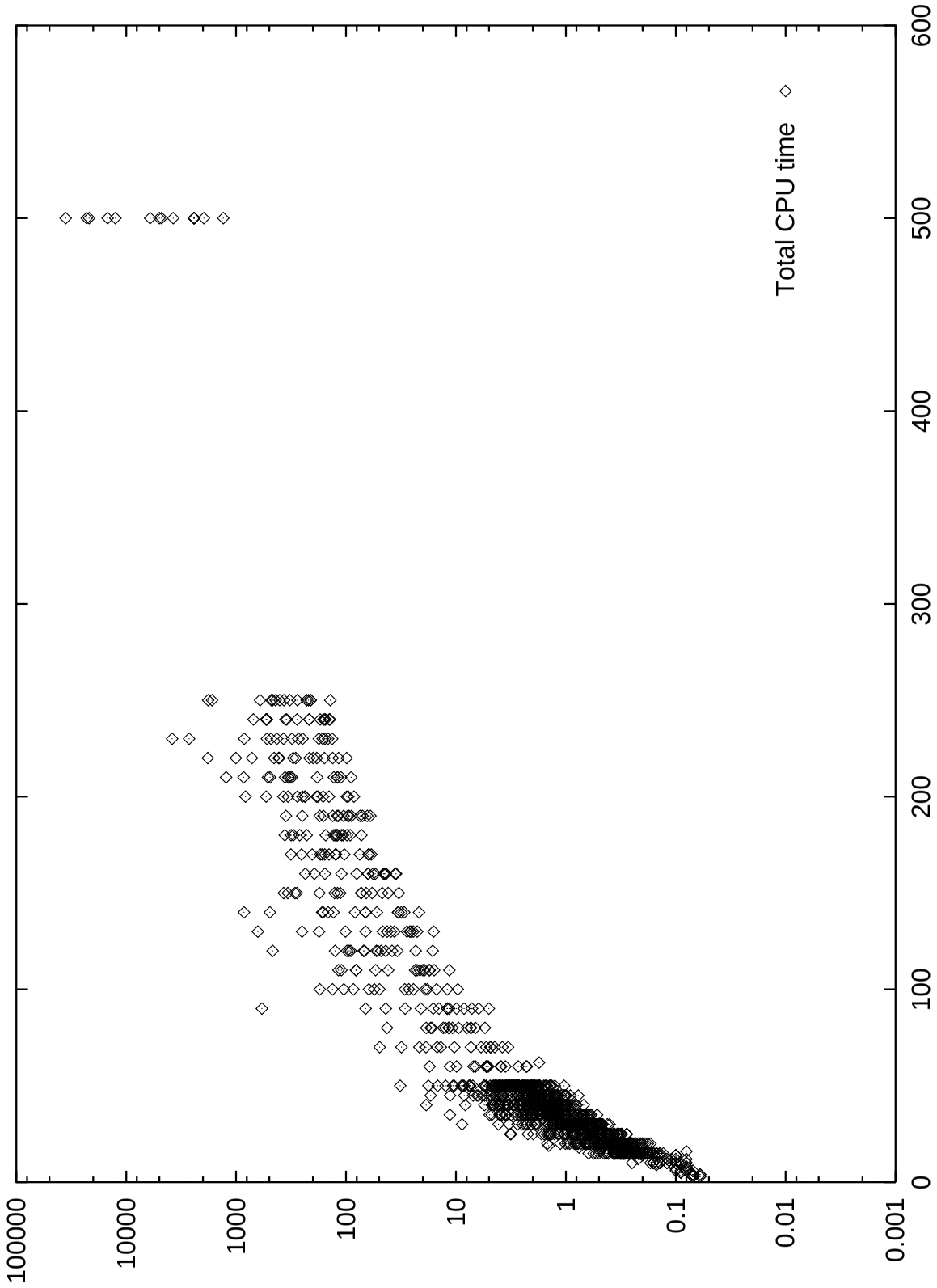

Figure 7: Plot of total CPU time vs. number of terminals.

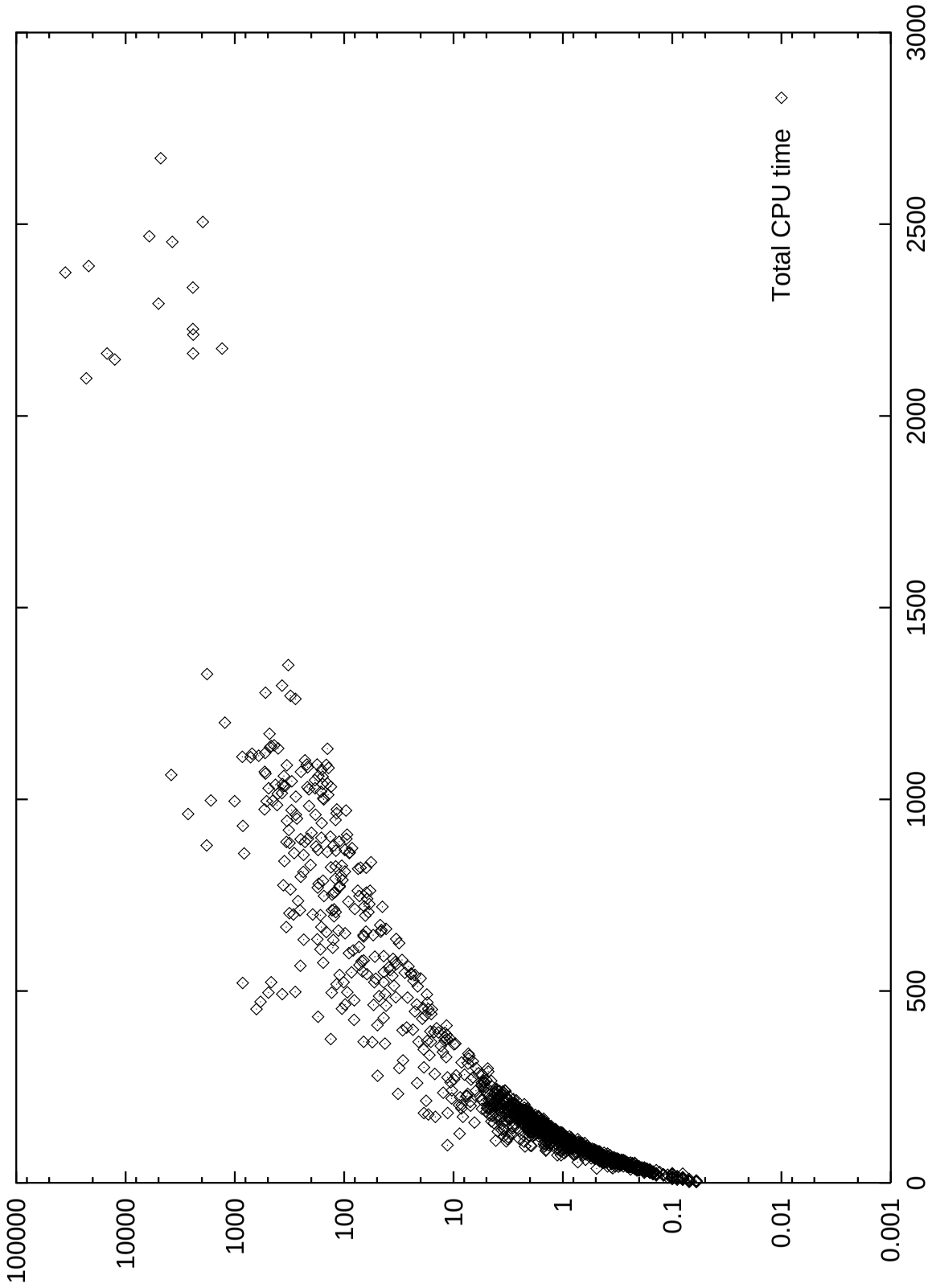

Figure 8: Plot of total CPU time vs. number of FSTs.

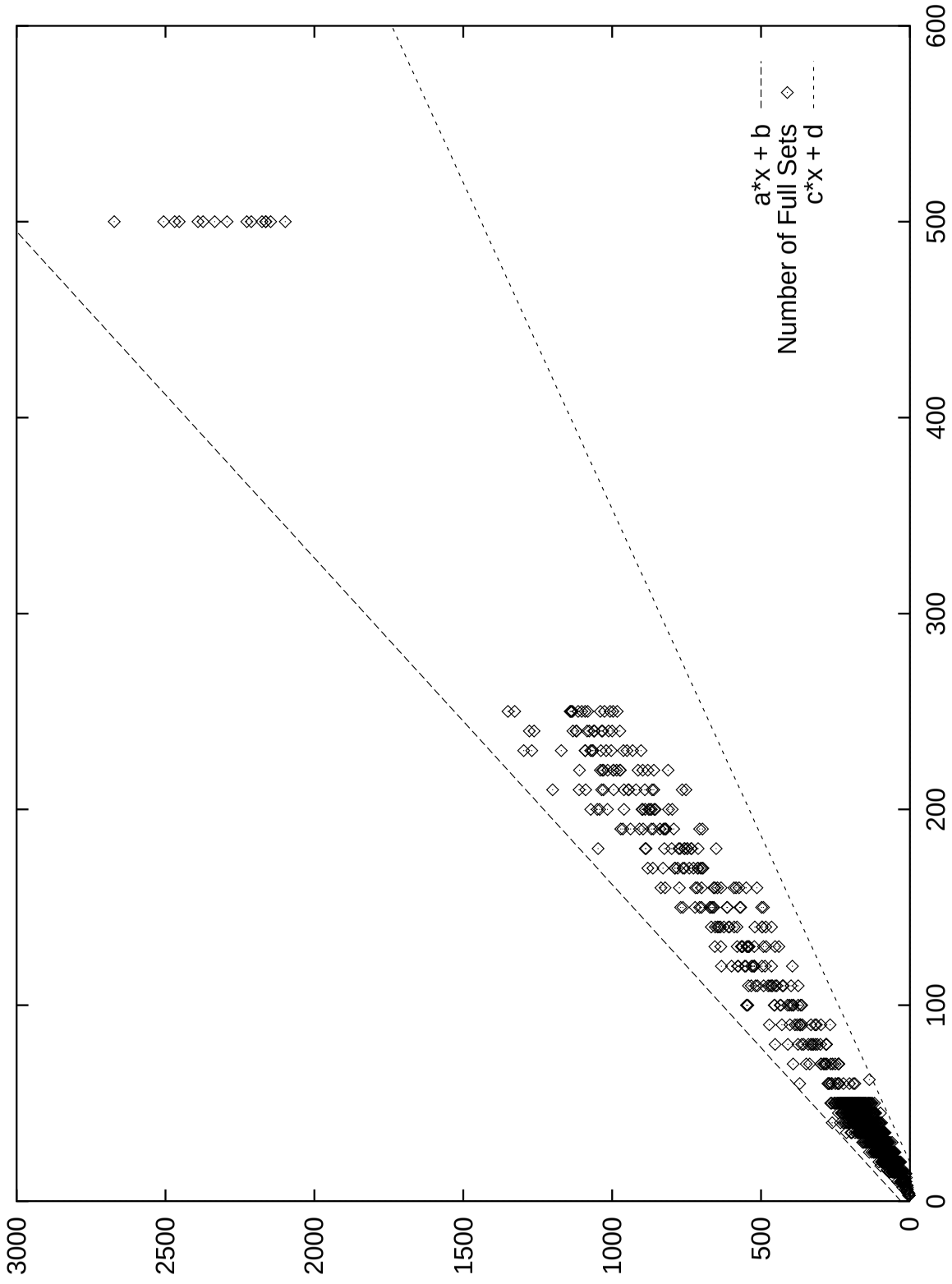

Figure 9: Plot of number of FSTs vs. number of points.

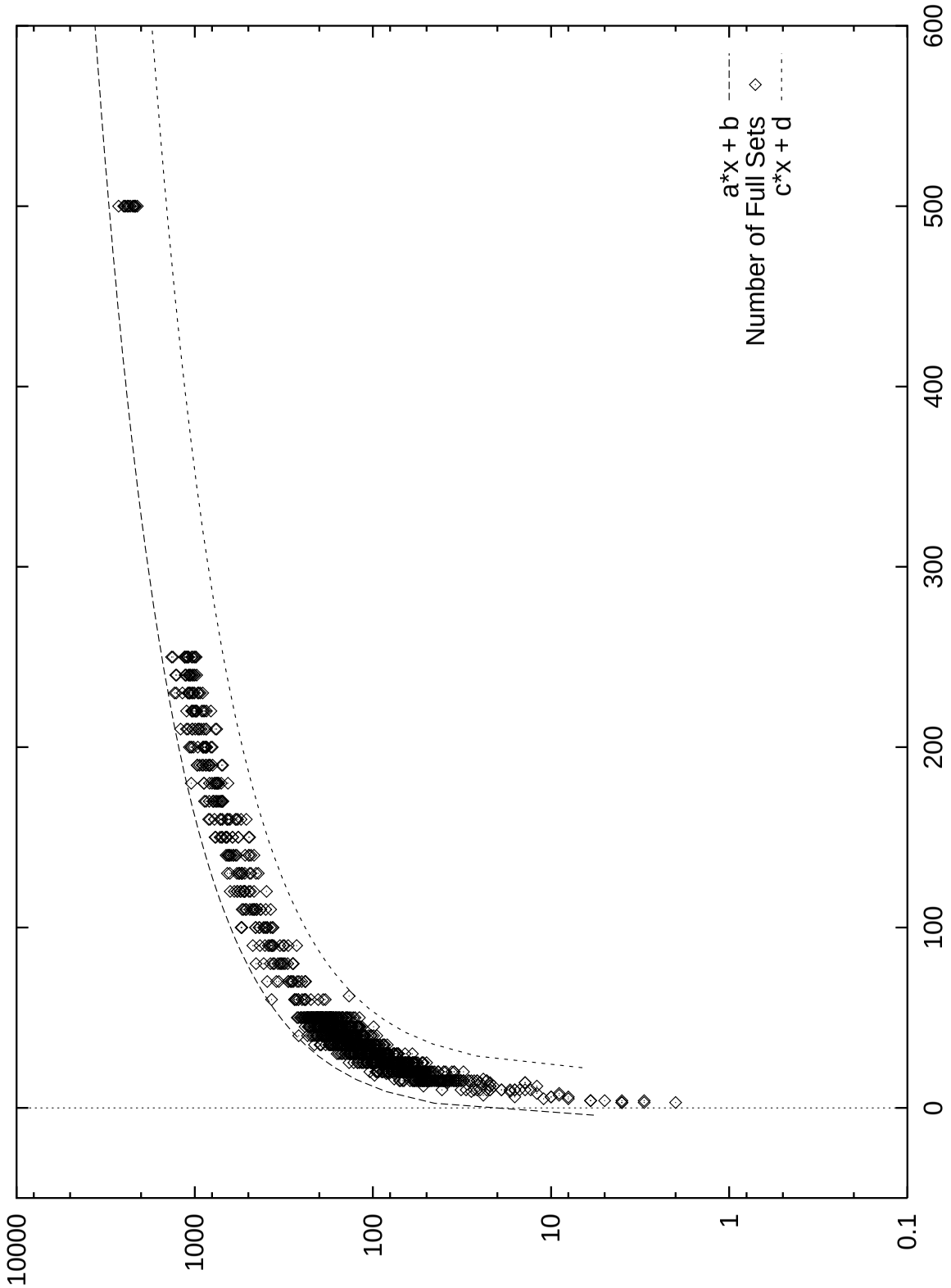

Figure 10: Logarithmic plot of number of FSTs vs. number of points.

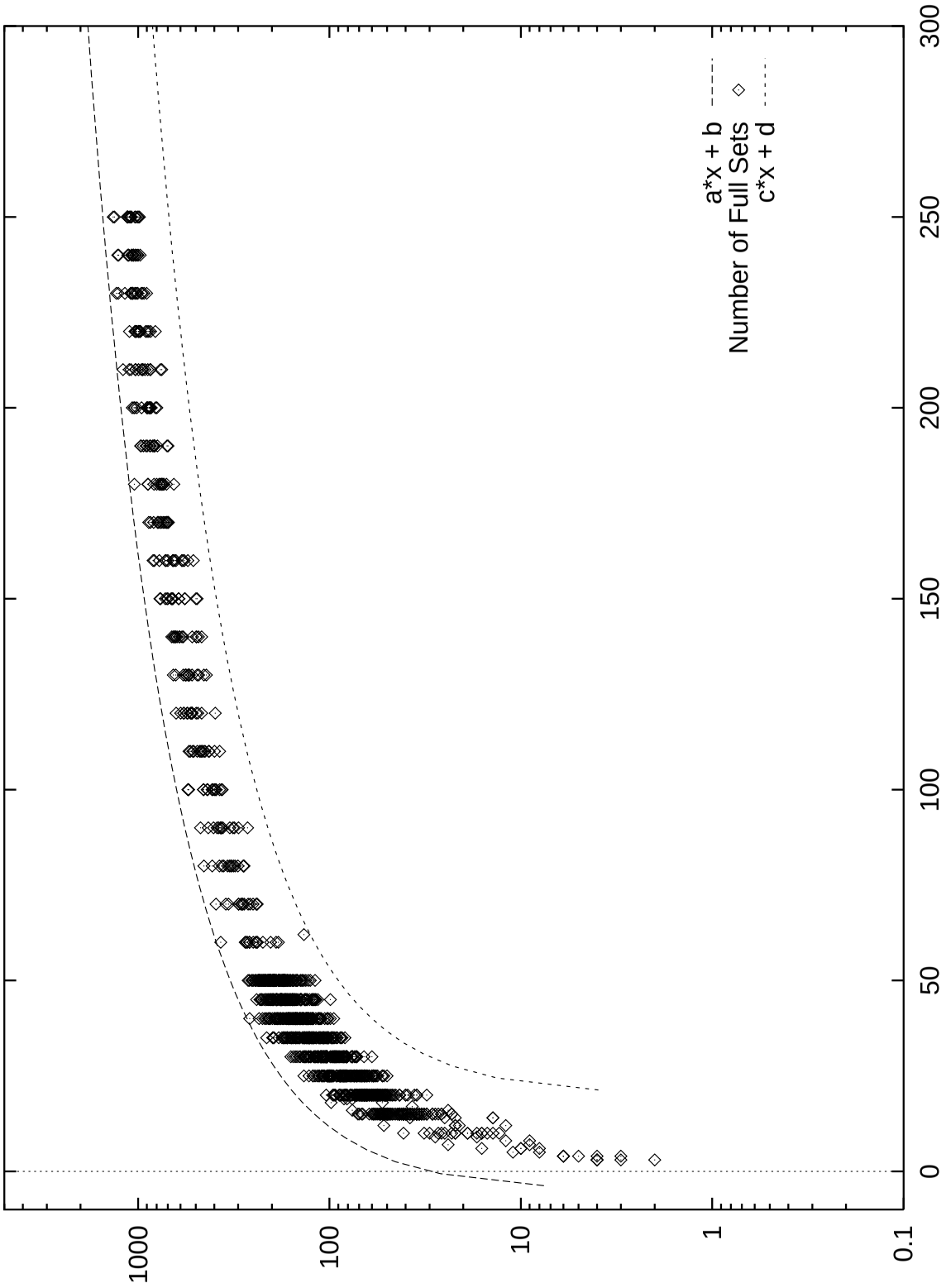

Figure 11: Expanded logarithmic plot of number of FSTs vs. number of points.

Prob	N	M	Z	Z	%	Nds	LPs	Constraints		CPU seconds		
				Root	Gap			IRow	RTight	Phase 1	Phase 2	Total
1	5	8	1.87	1.870000	0.00000	1	1	19	8	0.06	0.01	0.07
2	6	10	1.64	1.640000	0.00000	1	1	18	7	0.06	0.01	0.07
3	7	9	2.36	2.360000	0.00000	1	1	15	8	0.07	0.02	0.09
4	8	12	2.54	2.540000	0.00000	1	1	21	13	0.06	0.02	0.08
5	6	10	2.26	2.260000	0.00000	1	1	18	7	0.08	0.01	0.09
6	12	22	2.42	2.420000	0.00000	1	2	35	15	0.08	0.03	0.11
7	12	22	2.48	2.480000	0.00000	1	1	35	13	0.09	0.02	0.11
8	12	21	2.36	2.360000	0.00000	1	3	35	17	0.08	0.04	0.12
9	7	24	1.64	1.640000	0.00000	1	1	84	8	0.08	0.01	0.09
10	6	16	1.77	1.770000	0.00000	1	1	44	10	0.07	0.02	0.09
11	6	8	1.44	1.440000	0.00000	1	1	16	9	0.06	0.01	0.07
12	9	28	1.80	1.800000	0.00000	1	1	103	10	0.12	0.02	0.14
13	9	17	1.50	1.500000	0.00000	1	1	48	10	0.08	0.01	0.09
14	12	12	2.60	2.600000	0.00000	1	1	13	13	0.07	0.02	0.09
15	14	22	1.48	1.480000	0.00000	1	2	40	31	0.09	0.04	0.13
16	3	2	1.60	1.600000	0.00000	1	1	4	4	0.05	0.01	0.06
17	10	14	2.00	2.000000	0.00000	1	1	28	15	0.08	0.02	0.10
18	62	136	4.04	4.040000	0.00000	1	6	274	164	1.37	0.26	1.63
19	14	38	1.88	1.880000	0.00000	1	2	140	45	0.11	0.07	0.18
20	3	4	1.12	1.120000	0.00000	1	1	10	4	0.06	0.01	0.07
21	5	11	1.92	1.920000	0.00000	1	1	26	6	0.08	0.01	0.09
22	4	6	.63	0.630000	0.00000	1	1	15	5	0.06	0.01	0.07
23	4	5	.65	0.650000	0.00000	1	1	10	5	0.06	0.01	0.07
24	4	6	.30	0.300000	0.00000	1	1	14	5	0.05	0.02	0.07
25	3	4	.23	0.230000	0.00000	1	1	10	4	0.06	0.01	0.07
26	3	3	.15	0.150000	0.00000	1	1	7	4	0.06	0.01	0.07
27	4	4	1.33	1.330000	0.00000	1	1	8	6	0.05	0.02	0.07
28	4	6	.24	0.240000	0.00000	1	1	12	5	0.06	0.01	0.07
29	3	4	2.00	2.000000	0.00000	1	1	10	4	0.05	0.01	0.06
30	12	52	1.10	1.100000	0.00000	1	4	219	25	0.13	0.06	0.19
31	14	25	2.59	2.590000	0.00000	1	1	49	15	0.08	0.02	0.10
32	19	76	3.13	3.130000	0.00000	1	2	326	71	0.23	0.09	0.32
33	18	53	2.68	2.680000	0.00000	1	3	166	37	0.15	0.07	0.22
34	19	81	2.41	2.410000	0.00000	1	2	336	39	0.37	0.07	0.44
35	18	98	1.51	1.510000	0.00000	1	2	595	62	0.51	0.11	0.62
36	4	3	.90	0.900000	0.00000	1	1	5	5	0.06	0.01	0.07
37	8	9	.90	0.900000	0.00000	1	1	15	13	0.06	0.02	0.08
38	14	14	1.66	1.660000	0.00000	1	1	18	18	0.08	0.02	0.10
39	14	14	1.66	1.660000	0.00000	1	1	18	18	0.08	0.03	0.11
40	10	19	1.55	1.550000	0.00000	1	2	41	21	0.08	0.04	0.12
41	20	31	2.24	2.240000	0.00000	1	2	53	37	0.11	0.06	0.17
42	15	44	1.53	1.530000	0.00000	1	1	160	34	0.15	0.05	0.20
43	16	76	2.55	2.550000	0.00000	1	2	420	42	0.32	0.09	0.41
44	17	37	2.52	2.520000	0.00000	1	7	82	48	0.14	0.18	0.32
45	19	84	2.20	2.200000	0.00000	1	8	417	54	0.28	0.53	0.81
46	16	24	1.50	1.500000	0.00000	1	1	17	17	0.07	0.02	0.09

Table 1: Results for Soukup and Chow problems.

Prob	N	M	Z	Z	%	Nds	LPs	Constraints		CPU seconds		
				Root	Gap			IRow	RTight	Phase 1	Phase 2	Total
1	10	27	2.2920745	2.292075	0.00000	1	4	67	22	0.10	0.06	0.16
2	10	17	1.9134104	1.913410	0.00000	1	1	33	17	0.07	0.03	0.10
3	10	16	2.6003678	2.600368	0.00000	1	2	32	20	0.06	0.04	0.10
4	10	19	2.0461116	2.046112	0.00000	1	2	42	30	0.07	0.05	0.12
5	10	13	1.8818916	1.881892	0.00000	1	1	22	13	0.06	0.03	0.09
6	10	41	2.6540768	2.654077	0.00000	1	1	178	19	0.19	0.04	0.23
7	10	25	2.6025072	2.602507	0.00000	1	2	60	23	0.09	0.05	0.14
8	10	26	2.5056214	2.505621	0.00000	1	2	77	24	0.10	0.05	0.15
9	10	22	2.2062355	2.206236	0.00000	1	2	54	23	0.07	0.07	0.14
10	10	16	2.3936095	2.393610	0.00000	1	1	30	18	0.06	0.03	0.09
11	10	32	2.2239535	2.223953	0.00000	1	3	110	23	0.11	0.05	0.16
12	10	15	1.9626318	1.962632	0.00000	1	1	26	14	0.06	0.02	0.08
13	10	22	1.9483914	1.948391	0.00000	1	1	61	17	0.07	0.03	0.10
14	10	30	2.1856128	2.185613	0.00000	1	2	88	26	0.09	0.06	0.15
15	10	23	1.8641924	1.864192	0.00000	1	2	65	21	0.09	0.05	0.14
1	20	78	3.3703886	3.370389	0.00000	1	1	371	33	0.31	0.06	0.37
2	20	60	3.2639486	3.263949	0.00000	1	2	174	61	0.21	0.11	0.32
3	20	50	2.7847417	2.784742	0.00000	1	2	165	59	0.14	0.09	0.23
4	20	93	2.7624394	2.750218	0.44241	1	2	519	50	0.59	0.20	0.79
5	20	88	3.4033163	3.392034	0.33152	1	12	443	78	0.40	0.33	0.73
6	20	60	3.6014241	3.601424	0.00000	1	2	222	42	0.20	0.08	0.28
7	20	74	3.4934874	3.493487	0.00000	1	2	271	115	0.44	0.17	0.61
8	20	70	3.8016346	3.788557	0.34401	1	5	242	50	0.30	0.14	0.44
9	20	36	3.6739939	3.673994	0.00000	1	3	65	45	0.11	0.08	0.19
10	20	60	3.4024740	3.402474	0.00000	1	4	215	88	0.20	0.19	0.39
11	20	55	2.7123908	2.712391	0.00000	1	3	174	99	0.18	0.18	0.36
12	20	96	3.0451397	3.045140	0.00000	1	2	506	48	0.50	0.13	0.63
13	20	51	3.4438673	3.443867	0.00000	1	2	145	47	0.15	0.08	0.23
14	20	57	3.4062374	3.406237	0.00000	1	2	161	43	0.18	0.07	0.25
15	20	59	3.2303746	3.230375	0.00000	1	2	163	49	0.19	0.08	0.27
1	30	118	4.0692993	4.069299	0.00000	1	3	482	76	0.55	0.18	0.73
2	30	128	4.0900061	4.089173	0.02037	1	28	621	86	0.66	1.38	2.04
3	30	107	4.3120444	4.312044	0.00000	1	8	445	85	0.62	0.26	0.88
4	30	97	4.2150958	4.215096	0.00000	1	7	394	93	0.44	0.22	0.66
5	30	91	4.1739748	4.173975	0.00000	1	4	333	88	0.41	0.23	0.64
6	30	138	3.9955139	3.995514	0.00000	1	4	740	101	0.71	0.29	1.00
7	30	104	4.3761391	4.376139	0.00000	1	6	566	87	0.57	0.24	0.81
8	30	111	4.1691217	4.169122	0.00000	1	4	540	125	0.85	0.30	1.15
9	30	73	3.7133658	3.713366	0.00000	1	3	191	132	0.25	0.19	0.44
10	30	76	4.2686610	4.268661	0.00000	1	2	224	95	0.36	0.11	0.47
11	30	125	4.1647993	4.164799	0.00000	1	5	679	127	0.86	0.41	1.27
12	30	82	3.8416720	3.841672	0.00000	1	2	247	87	0.29	0.12	0.41
13	30	96	3.7406646	3.740665	0.00000	1	6	357	74	0.37	0.22	0.59
14	30	155	4.2897025	4.289702	0.00000	1	2	914	251	1.32	0.55	1.87
15	30	136	4.3035576	4.303558	0.00000	1	5	969	105	0.65	0.50	1.15
1	40	128	4.4841522	4.484152	0.00000	1	3	442	120	0.68	0.21	0.89
2	40	140	4.6811310	4.681131	0.00000	1	6	654	121	0.72	0.35	1.07
3	40	134	4.9974157	4.997416	0.00000	1	3	528	139	0.83	0.24	1.07
4	40	114	4.5289864	4.528986	0.00000	1	3	391	140	0.54	0.21	0.75
5	40	182	5.1940413	5.181185	0.24752	5	33	1157	125	1.49	3.05	4.54
6	40	137	4.9753385	4.975339	0.00000	1	2	584	152	0.65	0.30	0.95
7	40	132	4.5639009	4.563901	0.00000	1	5	541	114	0.67	0.27	0.94
8	40	150	4.8745996	4.874600	0.00000	1	7	604	101	1.00	0.41	1.41
9	40	194	5.1761789	5.176179	0.00000	1	4	1062	201	2.28	0.88	3.16
10	40	186	5.7136852	5.713685	0.00000	1	4	1180	114	1.42	0.29	1.71
11	40	133	4.6734214	4.673421	0.00000	1	11	435	240	0.65	0.69	1.34
12	40	147	4.3843378	4.384338	0.00000	1	7	735	216	0.84	0.78	1.62
13	40	131	5.1884545	5.188454	0.00000	1	3	481	113	0.62	0.25	0.87
14	40	152	4.9166952	4.916695	0.00000	1	3	517	174	0.80	0.36	1.16
15	40	175	5.0828067	5.082807	0.00000	1	3	864	208	1.01	0.44	1.45

Table 2: Results for OR-library problems 10–40 points.

Prob	N	M	Z	Z	%	Nds	LPs	Constraints		CPU seconds		
				Root	Gap			IRow	RTight	Phase 1	Phase 2	Total
1	50	227	5.4948660	5.494866	0.00000	1	13	1481	176	2.67	1.48	4.15
2	50	200	5.5484245	5.548425	0.00000	1	17	907	172	1.52	1.17	2.69
3	50	220	5.4691035	5.469104	0.00000	1	4	1382	167	2.02	1.04	3.06
4	50	154	5.1535766	5.153577	0.00000	1	6	500	190	1.03	0.35	1.38
5	50	171	5.5186015	5.518601	0.00000	1	7	722	125	1.44	0.43	1.87
6	50	226	5.5804287	5.580429	0.00000	1	23	1349	146	1.90	1.87	3.77
7	50	214	4.9961178	4.996118	0.00000	1	17	1117	143	1.89	1.61	3.50
8	50	129	5.3754708	5.375471	0.00000	1	2	396	136	0.92	0.23	1.15
9	50	193	5.3456773	5.343995	0.03146	1	6	1085	313	1.49	1.51	3.00
10	50	211	5.4037963	5.403796	0.00000	1	10	1297	155	1.75	0.90	2.65
11	50	174	5.2532923	5.253292	0.00000	1	5	679	149	1.25	0.40	1.65
12	50	166	5.3409291	5.325255	0.29347	7	19	726	187	1.17	1.38	2.55
13	50	147	5.3891019	5.389102	0.00000	1	4	555	141	0.88	0.30	1.18
14	50	185	5.3551419	5.355142	0.00000	1	3	1051	188	1.31	0.68	1.99
15	50	196	5.2180862	5.218086	0.00000	1	7	946	148	2.08	0.66	2.74
1	60	275	5.3761423	5.376142	0.00000	1	21	1786	240	3.21	1.92	5.13
2	60	370	5.5367804	5.530190	0.11902	1	6	3704	168	11.12	1.42	12.54
3	60	238	5.6566797	5.656680	0.00000	1	5	1206	177	2.11	0.68	2.79
4	60	270	5.5371042	5.537104	0.00000	1	9	1826	210	2.48	1.27	3.75
5	60	203	5.4704991	5.462873	0.13941	1	6	689	195	1.77	0.59	2.36
6	60	262	6.0421961	6.042196	0.00000	1	11	1698	172	2.69	1.09	3.78
7	60	275	5.8978041	5.897804	0.00000	1	5	1742	356	3.16	1.25	4.41
8	60	269	5.8138178	5.813818	0.00000	1	11	1754	290	2.77	1.65	4.42
9	60	250	5.5877112	5.587711	0.00000	1	4	1505	232	2.09	1.30	3.39
10	60	223	5.7624488	5.762449	0.00000	1	8	1015	205	1.99	0.92	2.91
11	60	185	5.6141666	5.614167	0.00000	1	3	683	156	1.67	0.36	2.03
12	60	262	5.9791362	5.979136	0.00000	1	13	1357	164	2.30	0.90	3.20
13	60	243	6.1213533	6.121353	0.00000	1	16	1134	195	2.14	1.67	3.81
14	60	240	5.6035528	5.603553	0.00000	1	10	1129	180	2.04	1.07	3.11
15	60	190	5.6622581	5.662258	0.00000	1	3	590	213	1.55	0.45	2.00
1	70	298	6.2058863	6.205886	0.00000	1	3	1532	231	3.31	0.66	3.97
2	70	260	6.0928488	6.092849	0.00000	1	13	1101	294	3.11	1.42	4.53
3	70	279	6.1934664	6.193466	0.00000	1	15	1357	233	2.55	1.90	4.45
4	70	241	6.2938583	6.293858	0.00000	1	27	1047	290	2.74	2.40	5.14
5	70	284	6.2256993	6.225699	0.00000	1	26	1287	230	3.07	2.62	5.69
6	70	337	6.2124528	6.212453	0.00000	1	8	2281	274	3.95	1.16	5.11
7	70	290	6.2223666	6.222367	0.00000	1	4	1517	251	3.14	0.67	3.81
8	70	299	6.1872849	6.187285	0.00000	1	31	1588	243	3.54	4.37	7.91
9	70	286	6.2986133	6.297066	0.02457	1	4	1703	339	3.22	1.14	4.36
10	70	251	6.2511830	6.249459	0.02757	1	6	999	182	2.43	0.74	3.17
11	70	348	6.6455760	6.643072	0.03768	1	14	2122	263	4.66	1.78	6.44
12	70	266	6.3047132	6.304713	0.00000	1	16	1182	257	2.48	1.35	3.83
13	70	239	6.2912258	6.291226	0.00000	1	3	1090	258	2.39	0.67	3.06
14	70	239	6.0411124	6.041112	0.00000	1	3	918	209	2.50	0.43	2.93
15	70	392	6.2318458	6.231846	0.00000	1	3	4140	186	7.17	0.81	7.98
1	80	333	7.0927442	7.092744	0.00000	1	18	1845	254	5.42	2.23	7.65
2	80	317	6.5273810	6.527381	0.00000	1	4	1435	254	4.27	1.19	5.46
3	80	328	6.5332546	6.533255	0.00000	1	16	1644	393	3.96	3.34	7.30
4	80	357	6.4193446	6.419345	0.00000	1	21	2027	231	4.15	2.28	6.43
5	80	282	6.6350529	6.634241	0.01224	1	6	1123	313	3.47	1.76	5.23
6	80	281	7.1007444	7.100744	0.00000	1	4	1042	292	3.86	0.86	4.72
7	80	410	6.8228475	6.822847	0.00000	1	5	3147	277	6.49	1.24	7.73
8	80	453	6.7452377	6.745238	0.00000	1	6	3357	200	11.01	1.44	12.45
9	80	376	6.9825651	6.977550	0.07183	2	18	2588	245	4.57	2.30	6.87
10	80	280	6.5497988	6.549799	0.00000	1	13	1192	279	4.12	2.17	6.29
11	80	311	6.6283099	6.628310	0.00000	1	10	1482	306	3.92	1.67	5.59
12	80	301	6.5070089	6.507009	0.00000	1	37	1832	293	4.35	4.70	9.05
13	80	363	6.8022647	6.802265	0.00000	1	25	2188	219	5.68	3.64	9.32
14	80	341	7.0077902	7.007790	0.00000	1	6	1785	255	4.83	1.26	6.09
15	80	324	6.9939071	6.984967	0.12783	2	7	1754	310	4.51	1.52	6.03

Table 3: Results for OR-library problems 50–80 points.

Prob	N	M	Z	Z	%	Nds	LPs	Constraints		CPU seconds		
				Root	Gap			IRow	RTight	Phase 1	Phase 2	Total
1	90	299	6.8350357	6.835036	0.00000	1	5	1270	255	4.63	0.78	5.41
2	90	319	7.1294845	7.129485	0.00000	1	36	1281	292	5.18	3.97	9.15
3	90	430	7.4817473	7.481114	0.00847	1	25	2612	320	7.65	5.21	12.86
4	90	361	7.0910063	7.091006	0.00000	1	11	1693	324	5.46	1.91	7.37
5	90	332	7.1831224	7.183122	0.00000	1	5	1257	294	5.31	0.96	6.27
6	90	386	6.8640346	6.864035	0.00000	1	5	2129	390	6.85	2.36	9.21
7	90	314	7.2036885	7.201542	0.02980	1	9	1236	276	5.34	1.24	6.58
8	90	371	7.2341668	7.234167	0.00000	1	24	1840	291	6.05	2.79	8.84
9	90	368	6.7856007	6.782613	0.04402	3	17	1918	309	5.85	3.72	9.57
10	90	403	7.2310409	7.231041	0.00000	1	12	2223	346	6.63	3.59	10.22
11	90	472	7.2310039	7.227387	0.05003	2	51	4483	324	7.91	13.90	21.81
12	90	368	6.9367257	6.936726	0.00000	1	11	1745	323	5.91	3.29	9.20
13	90	379	7.2810663	7.278959	0.02894	3	10	2161	328	6.34	2.23	8.57
14	90	268	6.9188992	6.918899	0.00000	1	4	915	253	4.01	0.58	4.59
15	90	368	7.1778294	7.177251	0.00806	3	18	2230	252	5.88	3.53	9.41
1	100	454	7.2522165	7.252217	0.00000	1	25	2418	376	10.45	6.86	17.31
2	100	544	7.5176630	7.517663	0.00000	1	20	3961	352	10.56	4.40	14.96
3	100	367	7.2746006	7.274601	0.00000	1	19	1850	341	6.68	3.61	10.29
4	100	391	7.4342392	7.434239	0.00000	1	5	2001	344	7.90	1.66	9.56
5	100	363	7.5670198	7.567020	0.00000	1	6	1598	273	6.83	1.34	8.17
6	100	546	7.4414990	7.441499	0.00000	1	8	4327	281	10.67	3.56	14.23
7	100	549	7.7740576	7.774058	0.00000	1	60	4152	364	10.18	10.80	20.98
8	100	405	7.3033178	7.303253	0.00088	1	25	2061	396	7.72	6.25	13.97
9	100	456	7.7952027	7.795203	0.00000	1	6	2936	337	9.80	3.11	12.91
10	100	433	7.5952202	7.595220	0.00000	1	25	2385	370	7.90	5.31	13.21
11	100	411	7.8674859	7.858919	0.10890	3	27	2396	268	7.84	3.49	11.33
12	100	393	7.6131099	7.613110	0.00000	1	25	1654	358	6.91	2.82	9.73
13	100	436	7.4604990	7.460499	0.00000	1	12	2749	348	9.57	3.09	12.66
14	100	375	7.8632795	7.859664	0.04598	2	78	1644	362	8.62	8.89	17.51
15	100	398	7.0446493	7.044649	0.00000	1	9	1994	328	7.23	1.68	8.91
1	250	1102	11.6609813	11.660981	0.00000	1	16	6975	874	126.29	17.06	143.35
2	250	1007	11.5150079	11.514005	0.00871	2	72	5216	954	109.25	42.14	151.39
3	250	983	11.4650399	11.465040	0.00000	1	108	5032	922	132.14	36.10	168.24
4	250	1139	11.7819530	11.780478	0.01252	1	29	7129	882	139.41	29.89	169.30
5	250	1091	11.6927089	11.692709	0.00000	1	260	6229	743	117.73	68.40	186.13
6	250	997	11.6256250	11.625306	0.00275	2	120	5478	864	104.09	70.12	174.21
7	250	1026	11.5277351	11.527735	0.00000	1	52	5187	845	107.38	26.02	133.40
8	250	1350	11.6833323	11.677163	0.05280	5	37	13101	861	134.65	32.60	167.25
9	250	1040	11.6821988	11.682199	0.00000	1	391	5382	866	101.02	262.45	363.47
10	250	1327	11.6857628	11.678762	0.05991	5	91	10731	786	137.29	101.13	238.42
11	250	1133	11.2889613	11.287079	0.01668	4	41	7359	834	113.88	36.38	150.26
12	250	1114	11.9035256	11.902872	0.00549	3	71	6565	927	119.90	55.39	175.29
13	250	1136	11.6049496	11.601749	0.02758	10	360	6421	1032	127.76	546.93	674.69
14	250	1082	11.6188791	11.618879	0.00000	1	16	6586	866	110.64	9.41	120.05
15	250	1141	11.5558198	11.555820	0.00000	1	64	7206	804	123.95	26.14	150.09
1	500	2163	16.2978810	16.297268	0.00376	1	45	12688	1806	1145.52	114.22	1259.74
2	500	2672	16.0756854	16.074292	0.00866	1	54	22178	1981	1246.65	119.77	1366.42
3	500	2506	16.2664661	16.266466	0.00000	1	203	18332	1702	1131.73	1408.78	2540.51
4	500	2227	16.4110997	16.411100	0.00000	1	151	14458	1815	1138.92	600.23	1739.15
5	500	2147	16.0586161	16.053088	0.03443	8	463	12160	1661	1053.85	3576.47	4630.32
6	500	2469	16.4685074	16.468507	0.00000	1	36	17369	1722	1157.88	133.61	1291.49
7	500	2176	16.0124233	16.011407	0.00635	1	36	12024	2007	1098.60	68.94	1167.54
8	500	2391	16.1248138	16.124644	0.00105	1	169	16539	1781	1174.61	1170.02	2344.63
9	500	2293	16.2100435	16.207268	0.01712	4	52	13656	1783	1150.01	172.46	1322.47
10	500	2163	15.5581203	15.558120	0.00000	1	78	11183	1757	1354.59	276.33	1630.92
11	500	2374	16.1674316	16.167418	0.00008	1	263	15081	1738	1229.21	2169.60	3398.81
12	500	2212	16.4009591	16.400625	0.00204	2	79	14191	1621	1086.06	477.21	1563.27
13	500	2098	16.1324201	16.131619	0.00497	4	60	10863	1704	1056.70	188.48	1245.18
14	500	2454	16.5984329	16.592090	0.03821	4	124	15826	1691	1165.24	1166.30	2331.54
15	500	2335	16.0758467	16.074649	0.00745	2	50	15510	1625	1120.82	95.68	1216.50

Table 4: Results for OR-library problems 90–500 points.

Prob	N	M	Z	Z Root	% Gap	Nds	LPs	Constraints		CPU seconds		
								IRow	RTight	Phase 1	Phase 2	Total
1	1000	4946	23.0535806	23.042695	0.04722	15	194	35583	3370	12990.01	3144.24	16134.25
2	1000	4838	22.7886471	22.788544	0.00045	2	105	32348	3816	11196.06	1101.57	12297.63
3	1000	4748	22.7807756	22.780639	0.00060	2	488	32508	3311	11857.87	25389.82	37247.69
4	1000	4772	23.0200846	23.017442	0.01148	7	501	29912	3571	11973.91	2996.16	14970.07
5	1000	4637	22.8330602	22.832172	0.00389	2	804	30412	3226	11965.21	35295.41	47260.62
6	1000	5024	23.1028456	23.095362	0.03239	14	3129	38001	3170	12327.95	463851.93	476179.88
7	1000	4660	23.0945623	23.093270	0.00560	2	885	30679	3542	11242.66	44177.91	55420.57
8	1000	5143	23.0639115	23.062650	0.00547	9	782	42136	3412	11338.99	60331.32	71670.31
9	1000	5251	22.7745838	22.773655	0.00408	7	106	44569	3439	16069.58	1342.32	17411.90
10	1000	4734	22.9267101	22.923663	0.01329	5	1365	30067	3412	11500.11	90181.37	101681.48
11	1000	4558	23.1605619	23.157667	0.01250	5	423	29557	3971	11632.65	2004.75	13637.40
12	1000	5490	23.0904712	23.088224	0.00973	4	1447	48009	3281	14826.25	269793.38	284619.63
13	1000	4518	22.8031092	22.803109	0.00000	1	2438	28873	3196	11544.44	259561.98	271106.42
14	1000	5144	23.4318491	23.426697	0.02199	18	1860	39659	3356	12080.61	320826.61	332907.22
15	1000	4867	22.9965775	22.994263	0.01006	2	242	35197	3457	12503.11	2255.47	14758.58

Table 5: Results for OR-library problems 1000 points.

terminals or fewer solved to optimality. We have also solved a single 2000 terminal Euclidean instance consisting of problems 1 and 2 combined (from the 1000 point OR-library problem set). See figure 13 for a plot of the optimal solution. Further details of this work will appear in a joint paper with those authors.

10 Conclusions

A new algorithm for the rectilinear Steiner minimal tree problem has been presented. It vastly outperforms any other published results of which the author is currently aware. Its nearest competitors seem to be the recent results of Alexander Martin and Thorsten Koch at Konrad Zuse in Berlin [18] (who report solutions to problems containing up to 40 terminals), and the more recent result of Fössmeier and Kaufmann [7] at Tübingen Universität (who report 55 point solutions, but have solved at least one instance of 100 terminals).

The new network flow separation algorithm for SECs is a major breakthrough — accomplishing in 10 CPU seconds what typically required 24 CPU hours with the previous submodular function minimization method of Grötschel, Lovasz and Schrijver. The new separation algorithm provides an overall execution time speedup of about 10-fold on large problems.

11 Acknowledgements

I would like to thank Karla Hoffman for introducing me to integer programming and the latest in branch-and-cut techniques. Her patience with me and her many insights into my problem are deeply appreciated. I am also most grateful for the CPU *months* and access to CPLEX that she *happily* provided.

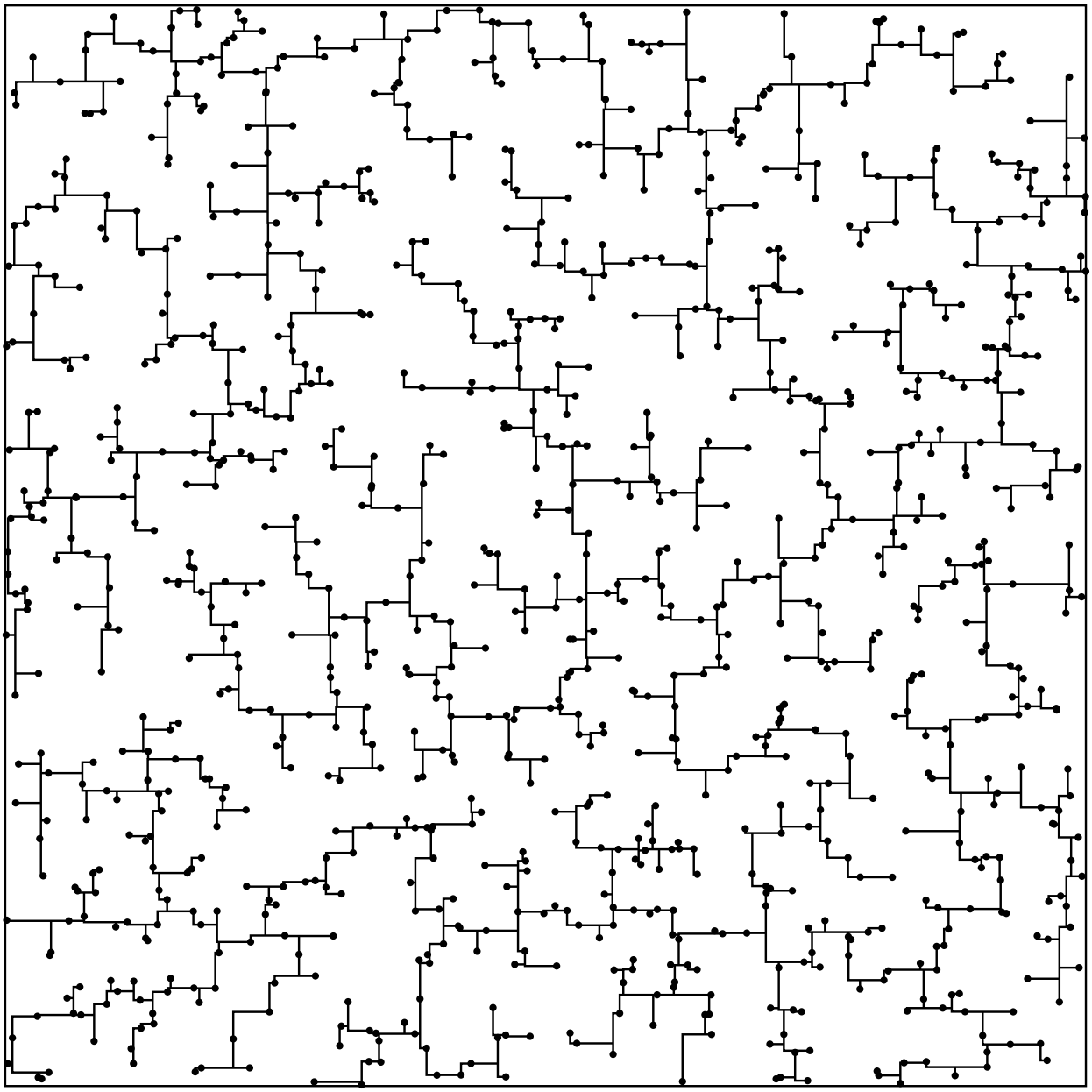

Figure 12: A rectilinear Steiner minimal tree for 1000 terminals. (Problem 1 from OR-library `estein1000.txt` file.)

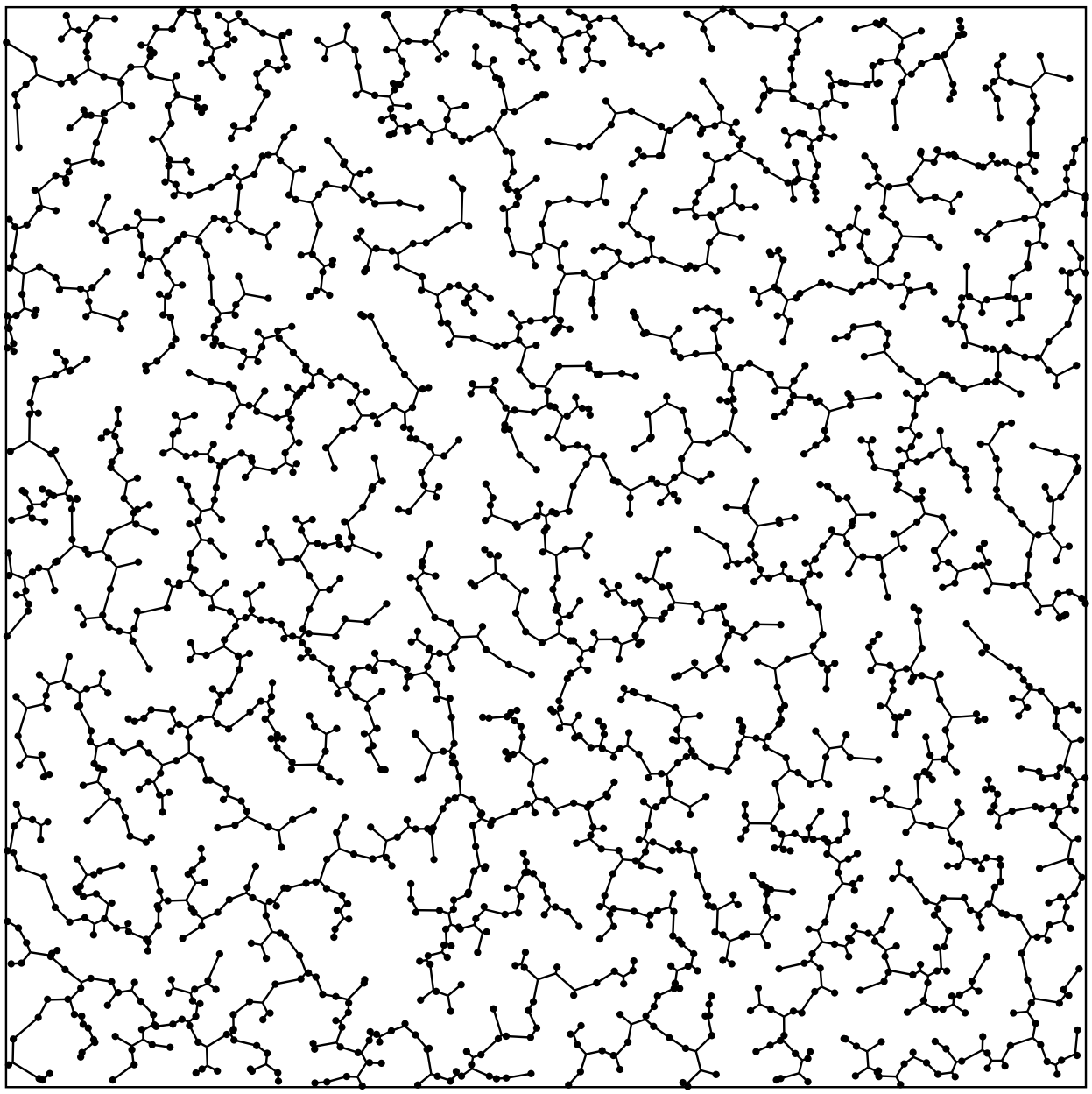

Figure 13: A Euclidean Steiner minimal tree for 2000 terminals. (Problems 1 and 2 combined from OR-library `estein1000.txt` file.)

I would also like to thank Maurice Queyranne for his keen insight that resulted in the major breakthrough on separating the subtour constraints (5.4). His NP-completeness proof for the minimum spanning tree in hypergraph problem was unsolicited, but greatly appreciated, as were our several stimulating conversations that contributed to the ideas in this paper.

The author would also like to express his gratitude to Abilio Lucena for his willingness to collaborate with me on my first experimental application of integer programming to the RSMT problem. This joint work solved RSMT problems by reduction to the Steiner problem in graphs. Rather than using the standard Hanan grid graph reduction [13] we used the graph that results when all of the FSTs from the Salowe-Warme phase 1 are overlaid. These graphs are *very* sparse compared to grid graphs (even convex ones). The resulting Steiner graph problems were solved using Lucena's existing branch-and-cut code. On standard grid graphs this code bogged down at 30 terminals. Using the overlaid FST graphs, however, a number of the 100 point problems from the OR-library were solved during April of 1996 — the first optimal solutions of these problems known to this author. These encouraging results led me to the present method that applies IP techniques directly to the FST concatenation problem.

I would also like to sincerely thank Pawel Winter and Martin Zachariasen for their splendid cooperation in getting their Euclidean code to interoperate with my rectilinear phase 1 and branch-and-cut FST concatenator. It has been a pleasure working with them and I anticipate even more impressive results from this partnership in the future.

References

[1] M. L. Balinski, On a selection problem, Management Sci., 17:230-231, 1970.

[2] J. E. Beasley, A heuristic for Euclidean and rectilinear Steiner problems, *European J. Oper. Res.*, 58:284–292, 1992.

[3] J. E. Beasley, OR-library — a collection of data sets for a variety of OR problems, (`http://mscmga.ms.ic.ac.uk/`).

[4] C. Berge, Graphs and hypergraphs, North-Holland, Amsterdam, Netherlands, 1973.

[5] E. J. Cockayne and D. E. Hewgill, Exact computation of Steiner minimal trees in the plane, *Inform. Process. Lett.*, 22:151–156, 1986.

[6] E. J. Cockayne and D. E. Hewgill, Improved computation of plane Steiner minimal trees, *Algorithmica*, 7:219–229, 1992.

[7] U. Fössmeier and M. Kaufmann, On exact solutions for the rectilinear Steiner problem, part I: theoretical results, *Technical report, WSI-96-09*, Universität Tübingen, Germany, 1996.

[8] J. L. Ganley, Geometric interconnection and placement algorithms, Ph.D. Dissertation, The University of Virginia, May 1995.

[9] M. R. Garey, R. L. Graham and D. S. Johnson, The complexity of computing Steiner minimal trees, *SIAM J. Appl. Math.*, 32:835–859, 1977.

[10] M. R. Garey and D. S. Johnson, The rectilinear Steiner problem is NP-complete, *SIAM J. Appl. Math.*, 32:826–834, 1977.

[11] M. Grötschel, L. Lovasz, and A. Schrijver, The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica*, 1:169–197.

[12] M. Grötschel, L. Lovasz, and A. Schrijver, Corregendum to our paper "The ellipsoid method and its consequences in combinatorial optimization", *Combinatorica*, 4:291–295.

[13] M. Hanan, On Steiner's problem with rectilinear distance, *SIAM J. Appl. Math.*, 14:255–265, 1966.

[14] Karla Hoffman, personal communication.

[15] F. K. Hwang, On Steiner minimal trees with rectilinear distance, *SIAM J. Appl. Math.*, 30:104–114, 1976.

[16] F. K. Hwang, D. S. Richards and P. Winter, The Steiner tree problem, volume 53 of *The Annals of Discrete Mathematics*, North-Holland, Amsterdam, Netherlands, 1992.

[17] A. B. Kahng and G. Robins, A new class of iterative Steiner tree heuristics with good performance, *IEEE Trans. Computer-Aided Design*, 11:893–902, 1992.

[18] T. Koch and A. Martin, Solving Steiner problems in graphs to optimality, Technical report SC 96-42, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 1996.

[19] G. L. Nemhauser and L. A. Wolsey, Integer and Combinatorial Optimization, Wiley, New York, 1988.

[20] M. W. Padberg and L. A. Wolsey, Trees and cuts, volume 17 of *The Annals of Discrete Mathematics*, 511–517, North-Holland, Amsterdam, Netherlands, 1983.

[21] J.-C. Picard, Maximal closure of a graph and application to combinatorial problems, Management Sci., 22:1268–1272, 1976.

[22] M. Queyranne, Personal communication, May 1997.

[23] J. M. W. Rhys, A selection problem of shared fixed costs and network flows, Management Sci., 17:200-207, 1970.

[24] J. S. Salowe and D. M. Warme, An exact rectilinear Steiner tree algorithm, *Proceedings of the International Conference on Computer Design*, Cambridge, Massachusetts, Oct 3–6, 1993.

[25] J. S. Salowe and D. M. Warme, Thirty-five point rectilinear Steiner minimal trees in a day, *Networks*, 25:69–87, 1995.

[26] J. Soukup and W. F. Chow, Set of test problems for the minimum length connection networks, *ACM/SIGMAP Newsletter*, 15:45–81, 1973.

[27] P. Winter, An algorithm for the Steiner problem in the Euclidean plane, *Networks*, 15:323–345, 1985.

[28] P. Winter and M. Zachariasen, Euclidean Steiner minimum trees: an improved exact algorithm, *Networks*, to appear.

[29] M. Zachariasen, Personal communication, April 1997.