Packing Steiner Trees: A Cutting Plane Algorithm and Computational Results

M. Grötschel A. Martin R. Weismantel

Abstract

In this paper we describe a cutting plane algorithm for the Steiner tree packing problem. We use our algorithm to solve some switchbox routing problems of VLSI-design and report on our computational experience. This includes a brief discussion of separation algorithms, a new LP-based primal heuristic and implementation details. The paper is based on the polyhedral theory for the Steiner tree packing polyhedron developed in our companion paper [GMW92] and meant to turn this theory into an algoritmic tool for the solution of practical problems.

1 Introduction

Given a graph $G = (V, E)$ and a node set $T \subseteq V$, we call an edge set $S \subseteq E$ a *Steiner tree for* T if, for each pair of nodes $u, v \in T$, S contains a $[u, v]$-path. The *Steiner tree packing problem*, as introduced in [GMW92], can be stated as follows. Given an undirected graph $G = (V, E)$ with edge capacities $c_e \in \mathbb{N}$ for all $e \in E$ and a list of node sets $\mathcal{N} = \{T_1, \ldots, T_N\}$, $N \in \mathbb{N}$, find Steiner trees S_k for T_k, $k = 1, \ldots, N$, such that each edge $e \in E$ is contained in at most c_e of the edge sets $S_1, \ldots, S_N$. Every collection of Steiner trees $S_1, \ldots, S_N$ with this property is called a Steiner tree packing. If a weighting of the edges is given in addition and a (with respect to this weighting) minimal Steiner tree packing must be found, we call this the *weighted Steiner tree packing problem*.

The motivation for studying this problem arises from the design of electronic circuits, i. e., the task of casting a given (complex) logic function in silicon. In a first phase (*logical design*) it is specified which of the given basic logical operations are combined to logical units (so-called *cells*) and which of these cells must be connected via wires. The points at which wires have to contact the cells are called *terminals*, and a set of terminals that must be connected is called a *net*. The list of cells and the list of nets are the input of the second phase, the *physical design*. The task here consists in assigning (placing) the cells to a given area and connecting (routing) the nets via wires. The problem, in fact, is more complicated than sketched above, since various company given design rules and technical contraints have to be taken into account and an objective function like the resulting area has to be minimized. Due to the inherent complexity, the problem is usually decomposed into the *placement problem* and the *routing problem*. We are interested in the routing problem. Roughly speaking, this problem can be stated as follows.

> Given an area (typically a rectangle with some "forbidden zones" occupied by the cells) and a list of nets. The routing problem is to connect (route) the terminals of each net by wires on the area such that certain technical side constraints are satisfied and some objective function is minimized.

The routing problem strongly depends on the used fabrication technology and the underlying design rules. The design rules specify, for instance, the routing area (i. e., the area that is available for connecting the nets) or the objective function (possible choices are, for example, the wiring length or the resulting area). The routing itself takes place on so-called *layers*. Each layer is divided into *tracks* on which the wires run. The tracks and the *vias*, the points where wires change the layers, must meet certain distance requirements.

The routing problem in its general form is still too complex to be solved in one step. In practice, the problem is generally decomposed into two subproblems. In a first step, one determines how wires "maneuver around the cells" (*global routing*). Here, the design rules are only partially considered. Thereafter, the wires are assigned to the layers and tracks according to the homotopy which was specified in the global routing phase (*detailled routing*). This decomposition scheme gives rise to many variants of the routing problem.

A number of the routing problems resulting from this approach can be modelled as a (weighted) Steiner tree packing problem. We will illustrate two examples in the following.

For modelling the global routing problem, the routing area is subdivided into subareas and these are represented by nodes in a graph. Of course, there are many ways to do this. One possible way of subdividing the routing area is illustrated in Figure 1. The enclosing rectangle represents the given area. The rectangular units with a diagonal between their lower left and upper right corner denote the cells. The routing area is subdivided into rectangular subareas (by means of the additional dotted lines in Figure 1). This subdivision of the routing area is represented by a graph as follows. We define a node for each subarea and introduce an edge between two nodes, if the corresponding subareas are adjacent. Let $G = (V, E)$ denote the resulting graph. Additionally, a capacity

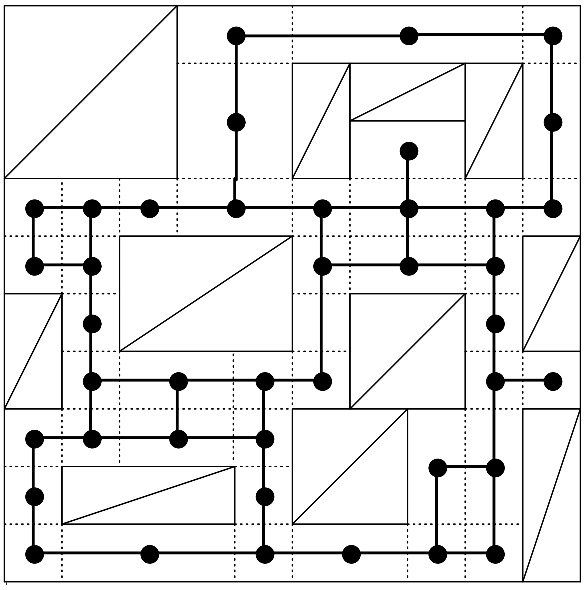

Figure 1:

$c_{uv} \in \mathbb{N}$ is assigned to an edge $uv \in E$ limiting the number of nets that may run between the subareas associated with the two nodes u and v. The weight of an edge w_{uv} corresponds to the distance between the two midpoints of the according subareas. The terminals of a net are assigned to those nodes, whose corresponding subareas contain the terminal or are as close as possible to the position of the terminal. The global routing problem consists in routing all nets in G such that the capacity constraints are satisfied and the total wiring length is as small as possible. Obviously, this task defines an instance of the Steiner tree packing problem.

The second example we want to mention is a variant of the detailled routing problem, called *switchbox routing problem* (see Figure 2). Here, the underlying graph is a complete rectangular grid graph and the terminal sets are located on the four sides of the grid. Remember that the task of detailled routing is to assign the wires to layers and tracks. Detailled routing problems, and thus also switchbox routing problems, are classified by distinguishing whether or to which extent the layers are taken into account while the nets are assigned to tracks. Here, the following modells are of special interest.

Multiple layer model Given a k-dimensional grid graph (that is a graph obtained by stacking k copies of a grid graph on top of each other and connecting related nodes by perpendicular lines), where k denotes the number of layers. The nets have to be routed in a node disjoint fashion. The multiple layer model is well suited to reflect reality. The disadvantage is that in general the resulting graphs are very large.

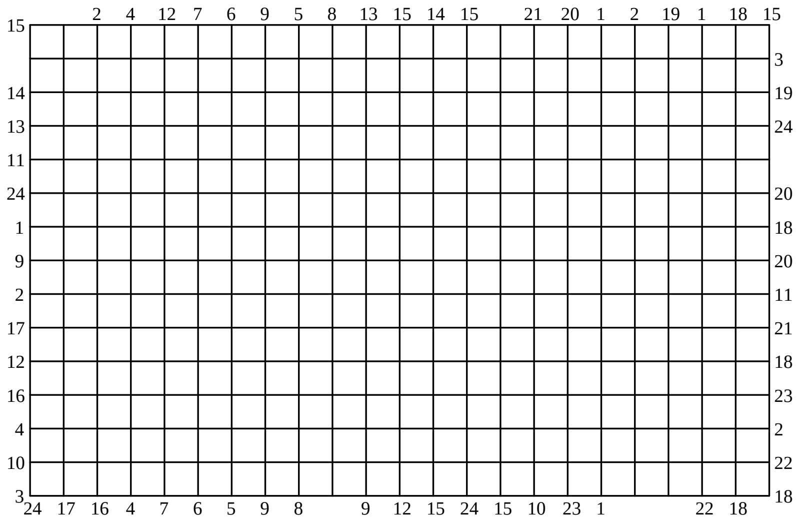

Figure 2:

Manhattan model Given a (subgraph of a) complete rectangular grid graph. The nets must be routed in an edge disjoint fashion with the additional restriction that nets that meet at some node are not allowed to bend at this node, i. e., so-called *knock-knees* (cf. Figure 3) are not allowed. This restriction guarantees that the resulting routing can be realized on two layers at the possible expense of causing long detours.

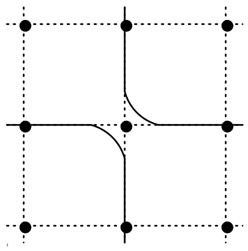

Figure 3:

Knock-knee model Again, a (subgraph of a) complete rectangular grid graph is given and the task is to find an edge disjoint routing of the nets. In this model knock-knees are possible. Very frequently, the wiring length of a solution in this case is smaller than in the Manhattan model. The main drawback is that the assignment to layers is neglected. Brady and Brown [BB84] have designed an algorithm that guarantees that any solution in this model can be routed on four layers. It was shown in [Li84] that it is $\mathcal{NP}$-complete to decide whether a realization on three layers is possible.

As in the case of the global routing problem the weighted Steiner tree packing problem is a natural mathematical model of the switchbox routing problem in the knock-knee mode. All examples that this computational study reports on are instances of this type of switchbox routing problems.

The paper is organized as follows. In section 2 we summarize some polyhedral results for the Steiner tree packing polyhedron. In particular, several classes of facet-defining inequalities are presented. In section 3 we briefly discuss the separation problem for these inequalities. Implementation issues of our branch and cut algorithm will be mentioned in section 4. Finally, we report on computational results for several switchbox routing instances in section 5.

Notation

We use the same graphtheoretic notation as in [GMW92]. Thus, we restrict ourselves in this subsection to briefly summarize the main notation concerning the (weighted) Steiner tree packing problem.

Let $G = (V, E)$ be a graph and $T \subseteq V$ a node set of G. An edge set S is called a *Steiner tree for* T, if the subgraph $(V(S), S)$ contains a path from s to t for all pairs of nodes $s, t \in T, s \neq t$. A Steiner tree that is a tree and whose leaves are terminals is called *edge-minimal*. We call an edge e a *Steiner bridge with respect to* T, if every Steiner tree for T in G contains e.

Problem 1.1 (The weighted Steiner tree packing problem)

Instance:

A graph $G = (V, E)$ with positive, integer capacities $c_e \in \mathbb{N}$ and nonnegative weights $w_e \in \mathbb{R}_+$, $e \in E$.

A list of node sets $\mathcal{N} = \{T_1, \ldots, T_N\}$, $N \geq 1$, with $T_k \subseteq V$ for all $k = 1, \ldots, N$.

Problem:

Find edge sets $S_1, \ldots, S_N \subseteq E$ such that

(i) S_k *is a Steiner tree in* G *for* T_k *for all* $k = 1, \ldots, N$,

(ii) $\sum_{k=1}^{N} |S_k \cap \{e\}| \leq c_e$ *for all* $e \in E$,

(iii) $\sum_{k=1}^{N} \sum_{e \in S_k} w_e$ *is minimal.*

If requirement (iii) in Problem 1.1 is omitted we call the corresponding problem the *Steiner tree packing problem* without the prefix "weighted". The list of node sets $\mathcal{N}$ is called a *net list*. The net list $\mathcal{N}$ is called *disjoint*, if $T_i \cap T_j = \emptyset$ for all $i, j \in \{1, \ldots, N\}$, $i \neq j$. Any element $T_k \in \mathcal{N}$ is called a *set of terminals* and the nodes $t \in T_k$ are called *terminals*. Instead of terminal set T_k we will often simply say *net* k. We call an N-tupel $(S_1, \ldots, S_N)$ of edge sets a *Steiner tree packing*

or *packing of Steiner trees* if the sets $S_1, \ldots, S_N$ statisfy (i) and (ii) of Problem 1.1. We will refer to an instance of the weighted Steiner tree packing problem by $(G, \mathcal{N}, c, w)$ and to an instance of the Steiner tree packing problem by $(G, \mathcal{N}, c)$.
We assume throughout the paper that every terminal set of the net list $\mathcal{N}$ has at least cardinality two and that $N \geq 1$.
Note that the Steiner tree packing problem as well as its weighted variant are $\mathcal{NP}$-complete or $\mathcal{NP}$-hard, respectively (see, for example, [K72], [GJ77], [KL84]). The problem remains hard in the case of switchbox routing problems in the knock-knee modell (see [S87]).

2 Polyhedral Results: A Short Review

In this section we give a short summary on some of the results of our companion paper [GMW92]. We use the notation of that paper. The *Steiner tree packing polyhedron* is defined as follows:

$$
\text{(2.1)} \quad \begin{aligned}
\mathrm{STP}\,(G, \mathcal{N}, c) := \mathrm{conv}\,\{\ (\chi^{S_1}, \ldots, \chi^{S_N}) \in \mathbb{R}^{\mathcal{N} \times E} \quad | & \\
(i) \quad & S_k \text{ is a Steiner tree for } T_k \text{ in } G \\
& \text{for } k = 1, \ldots, N; \\
(ii) \quad & \sum_{k=1}^{N} |S_k \cap \{e\}| \leq c_e, \text{ for all } e \in E\}.
\end{aligned}
$$

$\mathbb{R}^{\mathcal{N} \times E}$ denotes the $N \cdot |E|$ – dimensional vector space $\mathbb{R}^E \times \ldots \times \mathbb{R}^E$, where the components of each vector $x \in \mathbb{R}^{\mathcal{N} \times E}$ are indexed by x_e^k for $k \in \{1, \ldots, N\}$, $e \in E$. Moreover, for a vector $x \in \mathbb{R}^{\mathcal{N} \times E}$ and $k \in \{1, \ldots, N\}$, we denote by $x^k \in \mathbb{R}$ the vector $(x_e^k)_{e \in E}$, and we simply write $x = (x^1, \ldots, x^N)$ instead of $x = ((x^1)^T, \ldots, (x^N)^T)^T$. For an edge set $F \subseteq E$, χ^F denotes the incidence vector of F. It is easy to see that the following relation holds.

$$
\text{(2.2)} \quad \begin{aligned}
\mathrm{STP}\,(G, \mathcal{N}, c) = \mathrm{conv}\,\{x \in \mathbb{R}^{\mathcal{N} \times E} \ | & \\
\text{(i)} \quad & \sum_{e \in \delta(W)} x_e^k \geq 1, \text{ for all } W \subset V,\ W \cap T_k \neq \emptyset, \\
& \qquad (V \setminus W) \cap T_k \neq \emptyset,\ k = 1, \ldots, N; \\
\text{(ii)} \quad & \sum_{k=1}^{N} x_e^k \leq c_e, \quad \text{for all } e \in E; \\
\text{(iii)} \quad & 0 \leq x_e^k \leq 1, \quad \text{for all } e \in E,\ k = 1, \ldots, N; \\
\text{(iv)} \quad & x_e^k \in \{0, 1\}, \quad \text{for all } e \in E,\ k = 1, \ldots, N\}.
\end{aligned}
$$

The constraints (2.2) (ii) are called *capacity constraints* and the inequalities (2.2) (iii) *trivial inequalities*. It is $\mathcal{NP}$-hard to determine the dimension of

STP $(G, \mathcal{N}, c)$. Due to this fact we decided to investigate the facial structure of instances where the underlying graph is complete and the net list is disjoint. By applying the following two lemmas we can (partially) carry over results obtained for that special case to arbitrary instances.

Lemma 2.3 (Deleting an edge)
Let $(G, \mathcal{N}, c)$ be an instance of the Steiner tree packing problem. Let $a^T x \geq \alpha$ be a valid inequality of $STP(G, \mathcal{N}, c)$ and suppose $f \in E$ is deleted from G. Then $\hat{a}^T x \geq \alpha$ is a valid inequality of $STP(G \setminus f, \mathcal{N}, c)$ where $\hat{a}_e^k = a_e^k$ for all $e \in E \setminus \{f\}$, $k \in \{1, \ldots, N\}$ (where $G \setminus f$ denotes the graph that is obtained by deleting edge f).

Lemma 2.4 (Splitting a node)
Let $(G, \mathcal{N}, c)$ be an instance of the Steiner tree packing polyhedron. Let $f \in E$ with $c_f = 1$ and let $\hat{a}^T x \geq \alpha$ be a valid inequality of $STP(G \,/\, f, \mathcal{N}, c)$ ($G \,/\, f$ denotes the graph that is obtained by shrinking edge f). Then, $a^T x \geq \alpha$ defines a valid inequality for $STP(G, \mathcal{N}, c)$ with $a_e^k = \hat{a}_e^k$ for all $e \in E \setminus \{f\}$, $k \in \{1, \ldots, N\}$ and $a_f^k = 0$ for all $k = 1, \ldots, N$.

These two lemmas state that the validity of an inequality is preserved under applying graph operations like deleting an edge or splitting a node. In section 3 we will consider this issue in more detail.

Let us now summarize some results for the case that G is complete and the net list is disjoint.

First, the trivial inequalities $x_e^k \geq 0$ of (2.2) (iii) are facet-defining if and only if $|V| \geq 5$ or $e \notin E(T_k)$, whereas the trivial inequalities $x_e^k \leq 1$ of (2.2) (iii) are facet-defining if and only if $c_e \geq 2$. Moreover, the capacity constraints (2.2) (ii) are facet-defining if and only if $c_e \leq N - 1$.

We have also shown that each nontrivial facet-defining inequality of the Steiner tree polyhedron can be lifted to yield a facet-defining inequality of the Steiner tree packing polyhedron. More precisely, if $\hat{a}^T x \geq \alpha$ defines a facet of the Steiner tree polyhedron STP $(G, \{T_k\}, c)$ for some $k \in \{1, \ldots, N\}$, then $a^T x \geq \alpha$ defines a facet of STP $(G, \mathcal{N}, c)$, where $a_e^l = 0$ for $l \neq k$ and $a_e^k = \hat{a}_e$ for all $e \in E$. This theorem offers the opportunity to apply results for the Steiner tree polyhedron from the literature, see for instance Grötschel and Monma [GM90] and Chopra and Rao [CR88a], [CR88b]. Ball, Liu and Pulleyblank [BLP87] and Fischetti [F91], among others, studied the Steiner tree polyhedron for directed graphs.

We focus here on one class of facet-defining inequalities that was characterized in [GM90]. Let G be a graph and $T \subseteq V$ be a terminal set. We call a partition $V_1, \ldots, V_p$, $p \geq 2$, of V a *Steiner partition with respect to* T, if $V_i \cap T \neq \emptyset$ for $i = 1, \ldots, p$. Grötschel and Monma have shown that if G is connected and contains no Steiner bridge the following system is a non-redundant facet-defining system of inequalities for STP $(G, \{T\}, 1\!\!1)$.

$$x(\delta(V_1,\dots,V_p)) \geq p-1, \quad \begin{array}{l} V_1,\dots,V_p,\ p \geq 2, \text{ is a Steiner partition of } V \\ \text{with respect to } T \text{ such that} \\ \bullet\ (V_i, E(V_i)) \text{ is connected for } i = 1,\dots,p, \\ \bullet\ (V_i, E(V_i)) \text{ contains no Steiner bridge} \\ \quad \text{with respect to } V_i \cap T\ (i = 1,\dots,p), \text{ and} \\ \bullet\ G(V_1,\dots,V_p) \text{ is 2-node connected;} \end{array} \tag{2.5}$$

where $G(V_1,\dots,V_p)$ is the graph obtained from G by contracting each element of the partition to a single node. Each inequality that is induced by a Steiner partition $V_1,\dots,V_p$ is called a *Steiner partition inequality.* If $p=2$, the inequality is also called a *Steiner cut inequality.*

Let us now describe some results concerning joint inequalities, i. e., inequalities that combine two or more nets.

We consider the class of so-called alternating cycle inequalities. Let $G=(V,E)$ be a graph and $\mathcal{N} = \{T_1, T_2\}$ a net list. We call a cycle F an *alternating cycle with respect to* T_1, T_2, if $F \subseteq [T_1 : T_2]$ and $V(F) \cap T_1 \cap T_2 = \emptyset$. Moreover, let $F_1 \subseteq E(T_2)$ and $F_2 \subseteq E(T_1)$ be two sets of diagonals of the alternating cycle F with respect to T_1, T_2. The inequality $(\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})^T x \geq \frac{1}{2}|F| - 1$ is called an *alternating cycle inequality.*

Before stating the next theorem let us recall the definition of *cross free.* We say that two diagonals uv and rs of a cycle F *cross* if they appear on F in the sequence u, r, v, s or u, s, v, r; otherwise uv and rs are called *cross free.* For an alternating cycle F with respect to T_1, T_2, we call two sets of diagonals $F_1 \subseteq E(T_2)$ and $F_2 \subseteq E(T_1)$ *maximal cross free* if F_1 and F_2 are cross free (that is each pair of edges $e_1 \in F_1$ and $e_2 \in F_2$ is cross free), each diagonal $e_1 \in E(T_1) \setminus F_2$ crosses F_1 and each diagonal $e_2 \in E(T_2) \setminus F_1$ crosses F_2.

Theorem 2.6 *Let $G=(V,E)$ be the complete graph with node set V and let $\mathcal{N} = \{T_1, T_2\}$ be a disjoint net list with $T_1 \cup T_2 = V$ and $|T_1| = |T_2| = l$, $l \geq 2$. Furthermore, let F be an alternating cycle with respect to T_1, T_2 with $V(F) = V$ and $F_1 \subseteq E(T_2)$, $F_2 \subseteq E(T_1)$. Then the alternating cycle inequality*

$$(\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})^T x \geq l - 1$$

defines a facet of STP $(G, \mathcal{N}, 1\!\!1)$ if and only if F_1 and F_2 are maximal cross free.

We have also considered some extensions of the alternating cycle inequalities, which will be of interest in the subsequent section. First, let us focus on the case in which the underlying graph may contain parallel edges in addition. Here, each coefficient of an edge that is not parallel to an edge of the alternating cycle F, obtains the value of the "original edge". The coefficients of the edges that are parallel to an edge of F obtain the value 1. This results in the following theorem.

Theorem 2.7 *Let $G = (V, E)$ be a graph that contains the complete graph on node set V as a subgraph and let $\mathcal{N} = \{T_1, T_2\}$ be a disjoint net list with $T_1 \cup T_2 = V$ and $|T_1| = |T_2| = l$, $l \geq 2$. Furthermore let F be an alternating cycle with respect to T_1, T_2 with $V(F) = V$ and $F_1 \subseteq E(T_2)$, $F_2 \subseteq E(T_1)$. Then the alternating cycle inequality*

$$(\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})^T x \geq l - 1$$

defines a facet of $STP(G, \mathcal{N}, 1\!\!1)$ if and only if F_1 and F_2 are maximal cross free.

Another extension of the alternating cycle inequalities is obtained if an additional node z is added to the complete graph in Theorem 2.6. In our companion paper we have pointed out that there exist several alternatives how to choose the coefficients of the edges incident to the extra node z in order to obtain a facet-defining inequality for the corresponding Steiner tree packing polyhedron. In the following we give a selection of those alternatives that are taken into account by our separation algorithm (see the next section).

Suppose we have given a complete graph $G = (V \cup \{z\}, E)$ and a net list $\mathcal{N} = \{T_1, T_2\}$ such that $T_1 \cap V, T_2 \cap V$ is a partition of V with $|T_1 \cap V| = |T_2 \cap V| = \frac{|V|}{2}$. Furthermore, let F be an alternating cycle with respect to T_1, T_2 with $V(F) = V$ and let $F_1 \subseteq E(V \cap T_2)$ and $F_2 \subseteq E(V \cap T_1)$ be maximal cross free. In order to obtain a facet-defining inequality $\hat{a}^T x \geq \alpha$ with $\hat{a}|_{E(V)} = (\chi^{E(V)\setminus(F\cup F_1)}, \chi^{E(V)\setminus(F\cup F_2)})$ for STP $(G, \mathcal{N}, 1\!\!1)$ the remaining coefficients $\hat{a}^k_e$, $e \in \delta(z)$ can be independently chosen from the following list of alternatives for each net k.

Definition 2.8 (Possible choices for the new coefficients by adding an additional node z)
Let $k \in \{1, 2\}$ with $\bar{k} = 1$, if $k = 2$, and $\bar{k} = 2$, if $k = 1$.

(1) If z is a terminal of net k ($z \in T_k$), all coefficients obtain value 1, that is
$\hat{a}^k_e = 1$, *for all $e \in \delta(z)$.*

(2) If z is not a terminal of net k ($z \notin T_k$), there are the following possibilities.

(i) $\hat{a}^k_{zt} = 0$, *for one $t \in T_k$;*
$\hat{a}^k_e = 1$, *for all $e \in \delta(z) \setminus \{zt\}$.*

(ii) $\hat{a}^k_{zt} = 0$, *for one $t \in T_{\bar{k}}$;*
$\hat{a}^k_{zt'} = 0$, *for all $t' \in T_{\bar{k}}$ with $tt' \in F_k$;*
$\hat{a}^k_e = 1$, *for all remaining edges $e \in \delta(z)$.*

The following theorem is presented in a form that is convenient for the explanation of the separation algorithm in section 3. It can easily be checked that this theorem immediately follows from the related theorem in [GMW92].

Theorem 2.9 *Suppose* $Z = \{z_1, \ldots, z_q\}$, $q \geq 1$, *is a set of nodes and* $\hat{G} = (V \cup Z, \hat{E})$ *is the complete graph on node set* $V \cup Z$ *where* $V \cap Z = \emptyset$. *Set* $E := \hat{E} \setminus \hat{E}(Z)$ *and* $G := (V \cup Z, E)$ *and let* $\mathcal{N} = \{T_1, T_2\}$, $T_1, T_2 \subset V \cup Z$ *be a net list such that* $T_1 \cap V$, $T_2 \cap V$ *is a partition of* V *with* $|T_1 \cap V| = |T_2 \cap V| = l$, $l \geq 2$. *Furthermore, let* F *be an alternating cycle with respect to* T_1, T_2 *with* $V(F) = V$ *and let* $F_1 \subseteq E(T_2 \cap V)$, $F_2 \subseteq E(T_1 \cap V)$ *be maximal cross free. Let* $\hat{a} \in \mathbb{R}^{\mathcal{N} \times E}$ *be a vector such that* $\hat{a}|_{\delta(z_i)}$ *satisfies one of the alternatives of 2.8 for* $i = 1, \ldots, q$ *and such that* $\hat{a}|_{E \setminus \delta(Z)} = (\chi^{E(V) \setminus (F \cup F_1)}, \chi^{E(V) \setminus (F \cup F_2)})$. *Finally, set* $\alpha_i^k = |\{z_i\} \cap T_k|$ *for* $k = 1, 2$, $i = 1, \ldots, q$. *Then*

$$\hat{a}^T x \geq l - 1 + \sum_{i=1}^{q} (\alpha_i^1 + \alpha_i^2)$$

defines a facet of $STP(G, \mathcal{N}, \mathbb{1})$.

The next type of inequalities to be considered here are the so-called grid inequalities.

Let $G = (V, E)$ be a graph and $\mathcal{N} = \{T_1, T_2\}$ be a net list. Furthermore, let $\hat{G} = (\hat{V}, \hat{E})$ be a subgraph of G such that $\hat{G}$ is a complete rectangular $h \times 2$ grid graph with $h \geq 3$. Assume that the nodes of V are numbered such that $\hat{V} = \{(i, j) \mid i = 1, \ldots, h,\ j = 1, 2\}$. Moreover, let $(1, 1), (h, 2) \in T_1$ and $(1, 2), (h, 1) \in T_2$. We call the inequality $(\chi^{E \setminus \hat{E}}, \chi^{E \setminus \hat{E}})^T x \geq 1$ an $h \times 2$ *grid inequality.*

For ease of notation we assume that, if we consider a complete rectangular $h \times 2$ grid graph which is a subgraph of a given graph $G = (V, E)$, the node set V is numbered such that the nodes of the grid graph have a numbering as assumed in the above definition.

In [GMW92] we derived (very technical) conditions for an $h \times 2$ grid inequality to define a facet. Since our goal is to solve switchbox routing problems, and since the problem of determing the dimension of the corresponding polytope is $\mathcal{NP}$-hard (cf. [S87]), we mainly focus on the validity of the inequalities. In this case we can neglect most of the technical conditions. This yields the following theorem.

Theorem 2.10 *Let* $\hat{G} = (\hat{V}, \hat{E})$ *be a complete rectangular* $h \times 2$ *grid graph with* $h \geq 3$. *Let* $\mathcal{N} = \{T_1, T_2\}$ *be a net list where* $T_1 = \{(1, 1), (h, 2)\}$ *and* $T_2 = \{(1, 2), (h, 1)\}$. *Furthermore, let* $G = (V, E)$ *be a graph with* $\hat{V} \subseteq V$, $\hat{E} \subseteq E$ *such that* $\{uv \in E \mid u = (i, 1)$ *and* $v = (i, 2)$ *for some* $i \in \{1, \ldots, h\}\}$ *is a cut in* G. *Set* $F := \hat{E}$ *and let* $F_1, F_2 \subset E \setminus F$, *then the inequality*

$$(\chi^{E\setminus(F\cup F_1)}, \chi^{E\setminus(F\cup F_2)})^T x \geq 1$$

is valid for $STP(G, \mathcal{N}, \mathbb{1})$ if and only if F_1 and F_2 satisfy the following properties:

(i) For all $u, v \in V(F)$, $u \neq v$ there does not exist a path from u to v in (V, F_k) for $k = 1, 2$.

(ii) F_1 and F_2 are maximal with respect to property (i).

Note that the graph G in Theorem 2.10 is not necessarily complete. Especially, this type of inequalities is of interest for instances where the underlying graph is a grid graph. This is due to the fact that the corresponding subgraph $\hat{G}$ is automatically a grid graph in this case.

Finally, let us recall the so-called critical cut inequalities. Let $G = (V, E)$ be a graph with edge capacities $c_e \in \mathbb{N}$, $e \in E$. Moreover, let $\mathcal{N} = \{T_1, \ldots, T_N\}$ be a net list. For a node set $W \subseteq V$ we define $S(W) := \{k \in \{1, \ldots, N\} \mid T_k \cap W \neq \emptyset,\ T_k \cap (V \setminus W) \neq \emptyset\}$. We call a cut induced by a node set W *critical for* $(G, \mathcal{N}, c)$ if $s(W) := c(\delta(W)) - |S(W)| \leq 1$.

Suppose that V_1, V_2, V_3 is a partition of V such that $\delta(V_1)$ is a critical cut. Moreover, assume that $T_1 \cap V_1 = \emptyset$ and $T_1 \cap V_i \neq \emptyset$ for $i = 2, 3$. Then, the inequality $x^1([V_2 : V_3]) \geq 1$ is called a *critical cut inequality with respect to* T_1.

In [GMW92] two types of critical cut inequalities were exposed. The graph of the first type includes solely edges whose capacities are equal to one. The graph of the second type contains as few parallel edges as possible at the expense of higher capacities. Since we want to solve problem instances arising in switchbox routing, where $c = \mathbb{1}$ is given, we decided to concentrate on the first type.

Theorem 2.11 *Let $G = (V, E)$ be a graph with $V = \{u, v, w\}$ and let $\mathcal{N} = \{T_1, \ldots, T_N\}$ be a net list with $T_1 = \{u, v\}$. Set $E_{ij} := \{e \in E \mid e$ is incident to i and $j\}$ for $i, j \in V$ and $N_i := \{k \in \{1, \ldots, N\} \mid i \in T_k\}$ for $i \in V$. Assume that $|E_{uv}| \geq 2$, $|N_w| = N - 1$, $|N_u| + |N_v| = N + 1$, $|E_{uw}| \geq |N_u| - 1$, $|E_{vw}| \geq |N_v| - 1$ and $|E_{uw}| + |E_{vw}| = |N_u| + |N_v| - 1$. Then, the inequalilty*

$$x^1(E_{uv}) \geq 1$$

defines a facet of $STP(G, \mathcal{N}, \mathbb{1})$.

3 The Separation Problem for Several Classes of Inequalities

In this section we will briefly summarize the main ideas for separating the classes of inequalities presented in section 2. The separation algorithms and the associated correctness proofs are quite complicated. In order not to be beyond the

scope of this paper, we despense with the proofs and refer the reader to [M92] and [GMW93] for a detailled discussion of this issue. Formally, the *separation problem* for a given class of inequalities can be stated as follows.
Given an instance $(G, \mathcal{N}, c)$ of the Steiner tree packing problem, a vector $y \in \mathbb{R}^{\mathcal{N} \times E}$, $y \geq 0$, and a class of valid inequalities for STP $(G, \mathcal{N}, c)$. Decide, whether y satisfies all inequalities of the given class and, if not, find an inequality of this class violated by y.

Separation of the Steiner Partition Inequalities

Let us consider the class of Steiner partition inequalities introduced in (2.5). Unfortunately, the corresponding separation problem is $\mathcal{NP}$-complete in general. This result is due to [GMS90]. However, there exist special cases for which it can be solved in polynomial time. One of these special cases is obtained if we restrict the graph G to be planar and the set of terminal nodes T to lie on the outer face of G. This special case is of particular practical interest, because it includes the switchbox routing problem. In the following we describe the main idea of the algorithm for solving the separation problem in this case.

First of all, we know from [DW71] that a minimal (with respect to a weighting of the edges $w : E \mapsto \mathbb{R}$) Steiner tree for T can be calculated via a dynamic program. The idea of this procedure is based on the observation that for every minimal Steiner tree S there exists a node v and a subset $J \subseteq T$ such that S can be split up into two subtrees S_1 and S_2. Here, S_1 is an optimal Steiner tree with respect to $J \cup \{v\}$ and S_2 is an optimal Steiner tree with respect to $(T \setminus J) \cup \{v\}$. This observation leads to a recursion formula. For arbitrary graphs G and terminal sets T the running time of the algorithm is exponential in the number of terminals. However, in the particular case that G is planar and all terminals lie on the outer face of G Erickson, Monma and Veinott (cf. [EMV87]) showed that it is sufficient to consider subsets $J \subseteq T$ where the terminals of J are located consecutively on the outer face. Since there are no more than quadratic many of these subsets, a minimal Steiner tree can be computed in polynomial time using this recursion.

We modify this algorithm to solve the separation problem for the Steiner partition inequalities provided G is planar and all terminals lie on the outer face of G. Without loss of generality we can assume that G is 2-edge connected. Thus, the edge set that encloses the outer face of G is a cycle. Suppose the terminal set $T = \{t_1, \ldots, t_z\}$ is numbered in a clockwise fashion along this cycle. Now, consider the dual graph $G^* = (V^*, E)$ of G and subdivide the node representing the outer face in z nodes $d_1, \ldots, d_z$ such that every edge belonging to a path in G from t_i to t_{i+1} on the outer face is now incident to d_{i+1} for $i = 1, \ldots, z$. Let $G_D = (V_D, E)$ denote the resulting graph and set $D = \{d_1, \ldots, d_z\}$ (cf. Figure 4 (a) and (b)).

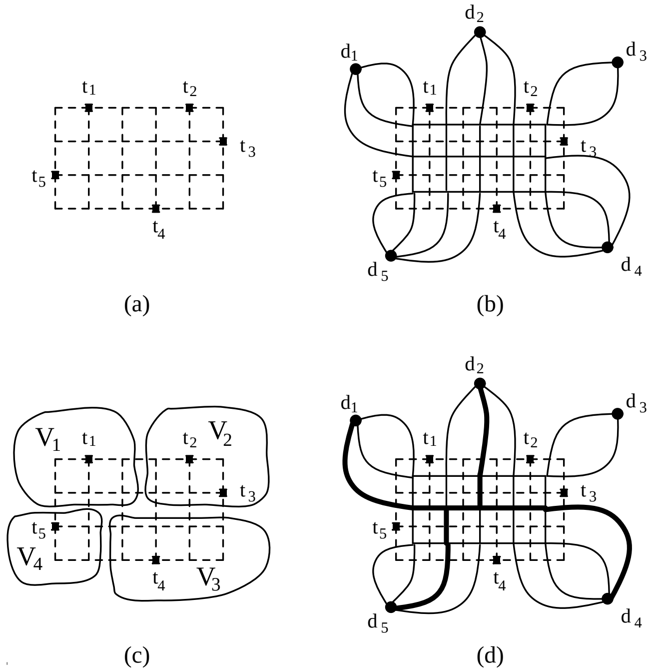

Figure 4:

It turns out that for every Steiner partition $V_1, \ldots, V_i$ such that $(V_k, E(V_k))$ is connected $(k = 1, \ldots, i)$ and such that the graph obtained by contracting every element of the partition to a single node is 2-node connected, the following holds. $S = \delta(V_1, \ldots, V_i)$ is an edge-minimal Steiner tree in G_D with respect to some subset $J \subseteq D$ with $|J| = i$ such that $d_S(j) = 1$ for all $j \in J$ and $d_S(t) = 0$ for all $t \in D \setminus J$.

Also conversely, every edge-minimal Steiner tree in G_D with these properties corresponds to a Steiner partition satisfying the above conditions (see Figure 4 (c) and (d)).

This equivalence yields, that the problem of separating the class of Steiner partition inequalities reduces to the problem of finding a subset J of D and an edge-minimal Steiner tree in G_D with respect to J. Clearly, given J, we can determine an optimal Steiner tree in G_D with respect to J by applying the dynamic programming approach discussed before. Thus, the crucial point is to find the subset $J \subseteq D$. Here, we could show that we can locally decide which terminal belongs to an optimal solution. This observation can be taken into account by modifying the recursion formula appropriately.

The algorithm for separating the Steiner partition inequalities gives rise to several heuristic procedures. Instead of calculating the optimal Steiner tree in G_D we heuristically determine Steiner trees. For more details, we again refer to our paper [GMW93].

Separation of the Alternating Cycle Inequalities

Given an instance $(G, \mathcal{N}, c)$ of a Steiner tree packing problem with $\mathcal{N} = \{T_1, T_2\}$ and a vector $y \in \mathbb{R}^{\mathcal{N} \times E}$, $y \geq 0$, decide, whether y satisfies all alternating cycle inequalities. If this is not the case, find an alternating cycle inequality that is violated by y.

Yet not proved, we strongly conjecture that, in general, this problem is not solvable in polynomial time. Instead, we restrict our attention to the case where G is planar and all terminals lie on the outer face of G. Here, in order to find an alternating cycle inequality, we proceed as follows.

Find a partition $V_1, \ldots, V_k$ of the node set V such that $G(V_1, \ldots, V_k)$, (the graph obtained by contracting every element of the partition to a single node) is, up to parallel edges, a subgraph of the one described in Theorem 2.9 or Theorem 2.6, respectively. Suppose for a moment that $G(V_1, \ldots, V_k)$ does not contain parallel edges. Due to Lemma 2.3 and Lemma 2.4 the deletion of edges and the splitting of nodes preserves validity of an inequality. Thus, by choosing the coefficients of the edges in $G(V_1, \ldots, V_k)$ according to Theorem 2.9 (resp. Theorem 2.6), we obtain an inequality that is valid for STP $(G, \mathcal{N}, c)$. Since $G(V_1, \ldots, V_k)$ is obtained by contracting nodes in G, we can not avoid in general that there are some edges in parallel. In this case, however, we lift these parallel edges by applying Theorem 2.7. We call inequalities obtained by this procedure *extended alternating cycle inequalities*.

The crucial point of this separation algorithm is of course, how to find a contraction minor, such that the resulting extended alternating cycle inequality is violated.

Here, our idea was to apply dynamic programming techniques in a similar way as was done for finding Steiner partition inequalities. Set $T = T_1 \cup T_2 = \{t_1, \ldots, t_z\}$ such that t_i and t_{i+1} are two subsequent terminals on the outer face of G. First, we construct the dual graph of G by splitting up the node representing the outer face of G into z nodes $d_1, \ldots, d_z$ such that every edge belonging to a path in G from t_i to t_{i+1} on the outer face is now incident to d_{i+1} for $i = 1, \ldots, z$. Let G_D denote the resulting graph and set $D = \{d_1, \ldots, d_z\}$. Here, every partition $V_1, \ldots, V_k$ yielding an extended alternating cycle inequality in G corresponds to a Steiner tree in G_D with respect to a subset of the set of terminals $\{d_1, \ldots, d_z\}$. However, this Steiner tree has to satisfy many technical conditions in this case.

In particular, these technical conditions cause that some edges are evaluated differently for the nets T_1 and T_2. This is due to the fact that for the alternating cycle inequality, edge sets F (edges which have a zero coefficient for both nets), F_1 (edges which have a zero coefficient just for net 1) and F_2 (edges which have a zero coefficient just for net 2) are involved as well (cf. Theorem 2.6 and Theorem 2.9). Unfortunately, taking all these constraints into account we obtain a recursion formula, which does not necessarily correspond to the most violated extended alternating cycle inequality. Rather, the value produced by the recursion provides just a lower bound for the most violated extended alternating cycle inequality. If this value is nonnegative, we can guarantee that there does not exist a violated inequality of this type. Otherwise, there may exist a violated extended alternating cycle inequality, but the algorithm terminates with an edge set that does not correspond to an extended alternating cycle inequality (see [GMW93]).

Beyond that the relationship between extended alternating cycle inequalities and certain Steiner trees in G_D that have to satisfy some technical conditions gives rise to many heuristics. Indeed, we have implemented an algorithm that determines heuristically such Steiner trees and checks whether the corresponding extended alternating cycle inequalities are violated.

Separation of the Grid Inequalities

For the separation problem for the grid inequalities described in Theorem 2.10, we could neither prove that this problem is $\mathcal{NP}$-complete nor that it is polynomially solvable. Therefore, we concentrate on heuristic algorithms. Suppose we are given a graph $G = (V, E)$ and a net list $\mathcal{N} = \{T_1, T_2\}$. Though we restrict our attention to valid (not necessarily facet-defining) inequalities the conditions in Theorem 2.10 are still quite restrictive. Especially the conditions that we have to find a subgraph $\hat{G} = (\hat{V}, \hat{E})$ that is a complete rectangular $h \times 2$ grid graph and that $T_1 = \{(1,1), (h,2)\}$ and $T_2 = \{(1,2), (h,1)\}$ are usually not satisfied by practical problem instances. Our idea was to relax these conditions such that the corresponding inequality $(\chi^{E\setminus(\hat{E}\cup F_1)}, \chi^{E\setminus(\hat{E}\cup F_2)})^T x \geq 1$ (where $F_1, F_2 \subset E \setminus \hat{E}$ are chosen apropriately) remains valid. The heuristic for separating this (new) class of inequalities works in a greedy like fashion. Again, for a detailled description of this algorithm we refer the interested reader to [GMW93].

Finding Critical Cuts

We consider now the problem of finding critical cuts. Remember that a cut induced by a set of nodes W is critical, if $s(W) = c(\delta(W)) - |S(W)| \leq 1$, where $S(W) = \{k \in \{1, \ldots, N\} \mid T_k \cap W \neq \emptyset,\ T_k \cap (V \setminus W) \neq \emptyset\}$. In the following we briefly explain why we concentrate on the problem of finding critical cuts rather than on the separation problem for the critical cut inequalities itself.

First, let us point out that, from a practical point of view, we are interested in Steiner tree packings where each of the single Steiner trees is edge-minimal. Since a positive objective function is minimized, we know in advance that the weight-minimal Steiner trees are also edge-minimal, and we exploit this property to reduce the problem size.

Suppose $W \subseteq V$ is a node set and T_k is a set of terminals with $T_k \subseteq W$ or $T_k \subseteq V \setminus W$. Then any edge-minimal Steiner tree for T_k that uses one edge of

$\delta(W)$ has to contain at least two of these edges. But, if $\delta(W)$ is a critical cut then at most one edge of $\delta(W)$ can be used by the Steiner tree for T_k. Hence, the following variables can be fixed accordingly, i. e.,

$$\begin{aligned} x_e^k = 0, &\quad \text{for all } k \in \{1,\ldots,N\} \setminus S(W),\, T_k \subseteq W,\, e \in E(V \setminus W) \cup \delta(W); \\ x_e^k = 0, &\quad \text{for all } k \in \{1,\ldots,N\} \setminus S(W),\, T_k \subseteq V \setminus W,\, e \in E(W) \cup \delta(W). \end{aligned}$$

Of course, instead of fixing these variables explicitly, we remove them from the linear program.

Let us now point out the relationship to the critical cut inequality. In Theorem 2.11 we made several assumptions. Given a graph $G = (V, E)$ where $V = \{u, v, w\}$ and a net list $\mathcal{N} = \{T_1, \ldots, T_N\}$ such that $T_1 = \{u, v\}$. Moreover, we require that $|E_{uv}| \geq 2$, $|N_w| = N - 1$, $|N_u| + |N_v| = N + 1$, $|E_{uw}| \geq |N_u| - 1$, $|E_{vw}| \geq |N_v| - 1$ and $|E_{uw}| + |E_{vw}| = |N_u| + |N_v| - 1$ (cf. Theorem 2.11 for the definition of E_{ij} for $i, j \in V$). These assumptions imply that $\delta(w)$ defines a cut satisfying $|S(\{w\})| = N - 1$ and $|\delta(w)| = |N_u| + |N_v| - 1 = N$. Since $c = \mathbb{1}$, we obtain $s(\{w\}) = c(\delta(w)) - |S(\{w\})| = |\delta(w)| - |S(\{w\})| = (N+1) - N = 1$. Therefore, the cut induced by $W = \{w\}$ is critical. Due to the above discussions, edge-minimal solutions satisfy

$$x_e^1 = 0, \quad \text{for all } e \in \delta(W).$$

Thus, by fixing these variables we can separate the critical cut inequalities implicitly by separating the Steiner cut inequalities. For example, a Steiner cut inequality for T_1 of the instance described in Theorem 2.11 is $x^1(\delta(u)) = x^1(E_{uw}) + x^1(E_{uv}) \geq 1$. By taking the fixed variables into account we obtain the critical cut inequality $x^1(E_{uv}) \geq 1$.

In the remainder of this subsection we briefly sketch the ideas, how to find critical cuts. We restrict ourselves to instances $(G, \mathcal{N}, \mathbb{1})$, where G is a complete rectangular grid graph and all terminal sets of the net list $\mathcal{N}$ lie on the outer face of G.

Theorem 3.1 *Let $G = (V, E)$ be a complete rectangular $h \times b$ grid graph and $\mathcal{N}$ a net list such that all terminal sets of $\mathcal{N}$ lie on the outer face of G. Let $W \subset V$, $\emptyset \neq W$, be a set of nodes and let the cut induced by W be critical with respect to the given instance $(G, \mathcal{N}, \mathbb{1})$. Then, one of the following statements is true.*

(i) There exists a node $w \in V$ such that $\delta(w)$ is a critical cut with respect to $(G, \mathcal{N}, \mathbb{1})$.

(ii) There exists a horizontal or vertical critical cut with respect to $(G, \mathcal{N}, \mathbb{1})$. (A cut F is called horizontal *if there exists some $i \in \{1, \ldots, h-1\}$ such that $F = \{uv \in E \mid u = (i, j)$ and $v = (i+1, j)$ for some $j \in \{1, \ldots, b\}\}$; a vertical cut is defined accordingly).*

Based on this lemma we can now develop an algorithm for finding critical cuts. We check, for all nodes $v \in V$, whether $\delta(v)$ is a critical cut. In addition, we also check this for the set of vertical and horizontal cuts. If we do not succeed in finding a critical cut, we can conclude that none exists. This is due to Lemma 3.1. Otherwise, we fix the corresponding variables. In order to find further critical cuts, we inductively enlarge the node set $W = \{v\}$ in all four possible directions of the grid in a greedy like fashion. The variables of the critical cuts found this way are fixed accordingly.

We want to point out that this algorithm is not really a separation algorithm. The input depends on the problem instance only and not on a given vector y. Thus, this algorithm is applied only once. However, critical cut inequalities are separated implicitly through the separation of the Steiner cut inequalities.

4 Implementation of the Cutting Plane Algorithm

In this section we develop a cutting plane algorithm for the (weighted) Steiner tree packing problem. After giving a short outline of the general procedure, we introduce a primal heuristic for the switchbox routing problem. It turns out that this heuristic plays an important role in our algorithm. Thereafter, we discuss some implementation details that are indespensable in solving practical problem instances. In particular, some of the implemetation issues are adapted to switchbox routing problem instances.

4.1 An Outline of the Cutting Plane Procedure

Our goal is to solve the following problem

$$\begin{aligned} \min \quad & \sum_{k=1}^{N} w^T x^k \\ & x \in \mathrm{STP}\,(G, \mathcal{N}, \mathbb{1}), \end{aligned}$$

via a cutting plane algorithm, where $(G, \mathcal{N}, \mathbb{1}, w)$ defines a weighted instance of the switchbox routing problem. The idea of a cutting plane algorithm is the following.

Start with a small set of valid inequalities, for example the trivial and capacity inequalities. These inequalities define a polyhedron P' that contains $\mathrm{STP}\,(G, \mathcal{N}, \mathbb{1})$. Optimize the linear objective function over P' and let y be an optimal solution. Obviously, y yields a lower bound for the optimum value of the weighted Steiner tree packing problem. If y is also feasible, y is an optimal solution for the weighted Steiner tree packing problem. Otherwise, there exists a valid inequality for $\mathrm{STP}\,(G, \mathcal{N}, \mathbb{1})$ that is violated by y. Thus, we must solve the separation problem, i. e., find a valid inequality that is violated by y. If such an inequality is found, we add it to the linear program and solve it again. The procedure of iteratively solving linear programs and adding violated constraints is commonly called a *cutting plane algorithm.*

A cutting plane algorithm ends with an optimal solution or (at least) with a lower bound for the weighted Steiner tree packing problem. The latter case is not avoidable in general, since we do not know a complete description of the Steiner tree packing polyhedron, and exact separation routines are not available for all known classes of facet-defining inequalities. If we intend to find an optimal solution of the problem we must embed the procedure into an enumeration scheme. Here, the overall problem is divided into two subproblems by fixing some variable to zero in the first subproblem and the same variable to one in the other subproblem. This process can be visualized by a binary tree (the so-called *branching tree*), where each subproblem is represented by a node. The whole method is commonly known as a *branch and cut algorithm.*

For the sake of efficiency the branching tree is to be kept as small as possible. In order to achieve this, the following issues are of special importance. First, we need a good description of the Steiner tree packing polyhedron by means of inequalities. This is the contents of our paper [GMW92]. Second, exact separation algorithms or at least good separation heuristics are necessary. This issue was considered in the preceeding section. Third, for the practical success of the total algorithm a procedure for finding a "good" feasible solution is of particular importance. This is due to the following reasons. A subproblem of the branching tree is solved if the value of the lower bound equals (up to rounding) that of the best feasible solution. In addition, for practical purposes it often suffices to find a provably good solution, i. e., a solution that differs from the optimal solution value only by a given percentage. Unfortunately, in our case it is $\mathcal{NP}$-complete to find a feasible solution for the Steiner tree packing problem, even if the instance defines a switchbox routing problem (see [S87]). So, we concentrated on developing a heuristic, which we introduce in the next section.

A Primal Heuristic

This section is devoted to describing our primal heuristic. The idea of our heuristic is to make use of the information given by the actual solution of the cutting plane phase.

We have developed a sequential algorithm. We consider each terminal of a net to be an (isolated) component. We iteratively connect two components of a net according to an a-priori determined sequence. However, we do not apply this scheme by routing one net completely after another, but we connect only two components in each iteration. The success of such a procedure strongly depends on the predefined sequence. In our algorithm this sequence is mainly determined by the solution y of the actual linear program. More precisely, we define a function f depending on y according to which the subsequent two components are selected. (A detailed explanation of the function f is given after the algorithmic description of the heuristic.) We try to connect the two selected components via a shortest path. Since in a complete rectangular grid graph a shortest path is not unique in general, we have implemented further criteria according to which the choice is made. Besides others, these criteria depend on the location of the terminals of the other nets, the position of the not yet connected terminals of the same net and, again, on the solution y. For a detailed description of these criteria we refer the reader to [M92]. If it is possible to connect the two components on a shortest path by taking the mentioned criterion into account, we connect these two components and choose the next pair of components. Otherwise, we recompute the function f and the sequence by taking the already connected components into account. This iterative procedure is continued until all nets are connected or no further components can be connected. In detail, the algorithm can be described as follows.

Algorithm 4.1 (A Primal Heuristic)

Input:

A complete rectangular $h \times b$ grid graph $G = (V, E)$ with edge capacities $c_e = 1$ and edge weights $w_e \in \mathbb{R}_+$, $e \in E$. Furthermore, a net list $\mathcal{N} = \{T_1, \ldots, T_N\}$ and a vector $y \in \mathbb{R}^{\mathcal{N} \times E}$, $y \geq 0$.

Output:

A feasible solution of the weighted Steiner tree packing problem $(G, \mathcal{N}, \mathbb{1}, w)$ or the message "No feasible solution found".

(1) Set $S_k := \emptyset$ for $k = 1, \ldots, N$.
(2) Determine the graph $\hat{G} = (V, \hat{E})$ with $\hat{E} := \{e \in E \mid c_e > 0\}$.
(3) Compute shortest paths for all pairs of nodes in $\hat{G}$.
(4) For $k = 1, \ldots, N$ perform the following steps:
(5) If $S_k = \emptyset$, then
determine $s_k, t_k \in T_k$ such that

$$f_{y^k}(s_k, t_k) = \min_{\substack{s,t \in T_k \\ s \neq t}} f_{y^k}(s, t);$$

set $T'_k := T_k \setminus \{t_k\}$.
(6) Else
determine $s_k \in T'_k, t_k \in V(S_k)$ such that

$$f_{y^k}(s_k, t_k) = \min_{s \in T'_k, t \in V(S_k)} f_{y^k}(s, t).$$

(7) As long as further connections are possible perform the following steps:
(8) Determine $k_0 \in \{1, \ldots, N\}$ with

$f_{y^{k_0}}(s_{k_0}, t_{k_0}) = \min\{f_{y^k}(s_k, t_k) \mid k = 1, \ldots, N\}$.

(9) Try to connect s_{k_0} with t_{k_0} via a shortest path by taking the above criteria into account.

(10) If the connection via a shortest path is possible, then
let W be the chosen path;
set $S_{k_0} := S_{k_0} \cup W$, $T'_{k_0} := T'_{k_0} \setminus \{s_{k_0}\}$ and $c_e := 0$ for all $e \in W$;
if $T'_{k_0} = \emptyset$, set $f_{y^{k_0}}(s_{k_0}, t_{k_0}) := \infty$;
else determine another pair (s_{k_0}, t_{k_0}) similar to (6).

(11) Else goto (2);

(12) If all terminal sets are connected, return the feasible solution $(S_1, \ldots, S_N)$.

(13) Otherwise, print the message "No feasible solution found".

(14) STOP.

Let us now define the function $f_{y^k} : V \times V \mapsto \mathbb{R}_+$ for some $k \in \{1, \ldots, N\}$. We give a formal definition first and explain the underlying heuristic idea afterwards. For the ease of exposition let the nodes be numbered such that $V = \{(i,j) \mid i = 1, \ldots, h,\ j = 1, \ldots, b\}$ and let $V_{l,r,t,d} := \{(i,j) \mid i = l, \ldots, r,\ j = t, \ldots, d\}$ for $l, r \in \{1, \ldots, b\}, l < r$ and $t, d \in \{1, \ldots, h\}, t < d$. Suppose we want to execute step (5) (resp. (6)) in Algorithm 4.1. Let S_k be the edge set that was already determined for connecting T_k, T'_k the set of not yet connected terminals and $\hat{G}$ the underlying graph.

We consider the case $S_k \neq \emptyset$ (in the case $S_k = \emptyset$ the function f_{y^k} is defined similarly) and let $s_k = (i_s, j_s) \in T'_k$ and $t_k = (i_t, j_t) \in V(S_k)$ be given. Determine $l, r \in \{1, \ldots, b\}, l < r$ and $t, d \in \{1, \ldots, h\}, t < d$ such that $s_k, V(S_k) \in V_{l,r,t,d}$ and $|V_{l,r,t,d}|$ is minimal. Set $E_{l,r,t,d} = \{e \in \hat{E}(V_{l,r,t,d}) \mid y^k_e > 0\}$ and suppose (V_s, E_s) is the component in $(V_{l,r,t,d}, E_{l,r,t,d})$ with $s_k \in V_s$. Set

$$f_{y^k}(s_k, t_k) := |w(W(s_k, t_k)) - \sum_{e \in E_s} w_e\, y^k_e|,$$

where $W(s_k, t_k)$ is a shortest path from s_k to t_k in $\hat{G}$ (with respect to w).

The heuristic idea of this function is the following. We determine a graph $(V_{l,r,t,d}, E_{l,r,t,d})$ which is the smallest rectangular grid graph containing both components (often designated as the "minimal enclosing rectangle"). Inside the minimal enclosing rectangle we compute the weighted sum ($= \omega$) of those edges that are in the same component as s_k, where only edges with $y^k_e > 0$ are considered. The value ω is compared to the length ($= \lambda$) of a shortest path between the two nodes. If ω is smaller than λ, we assume that the information from y^k is too poor to decide how to connect the two nodes. The smaller the difference, the less information and the greater the value of f. On the other hand, if ω is greater than λ the two nodes will be probably connected via a detour. The greater the difference, the greater the value of f. Thus, we choose the components with value ω close to λ first.

Obviously all ideas mentioned so far are of heuristic nature and there is no guarantee that we will obtain good results. However, due to many tests we have performed this strategy seems to be reasonable.

Implementation Details

In this section we want to focus on some little "tricks" that enter into our cutting plane algorithm. The underlying ideas might appear easy and not very deep to the reader. However, it turns out that these ideas are very effective and indespensible for solving practical problems. We want to illustrate the effects of the ideas on an example called "difficult switchbox" (for the data of this problem see the next section).

Let us mention here that we use the code CPLEX of R. E. Bixby (Rice University, Houston, Texas) for solving the linear programs that come up. Without such a fast and robust code we would not have been able to solve the given problem instances. The linear programs we encountered appeared to be quite difficult. One of the reasons for this is probably that our linear programs have many alternative optimum solutions and are simultaneously primally and dually highly degenerate.

A frequently used method to overcome such difficulties is to perturbe the right hand side of the linear program. Since we are solving the problems with the dual simplex method we must perturbe the objective function of the weighted Steiner tree packing problem. After many experiments and discussions with R. E. Bixby we decided to proceed as follows. Let $\omega \in \mathbb{R}^{\mathcal{N} \times E}$ with $\omega_e^k = w_e$ for all $e \in E$, $k = 1, \ldots, N$ be the original objective function. For each terminal set T_k, we compute a Steiner tree S_k by applying a heuristic procedure and determine random numbers $\varepsilon_e^k \in [0, 1]$. Then we use the objective function vector $\tilde{w} \in \mathbb{R}^{\mathcal{N} \times E}$ defined by

$$\begin{aligned} \tilde{w}_e^k &:= \omega_e^k - b\,\varepsilon_e^k - \eta, \quad && \text{if } e \in S_k, \text{ for } k = 1, \ldots, N; \\ \tilde{w}_e^k &:= \omega_e^k - b\,\varepsilon_e^k, && \text{if } e \notin S_k, \text{ for } k = 1, \ldots, N. \end{aligned} \tag{4.2}$$

where $\eta = \frac{1}{2(n+1)}$ and $b = \min\{10^{-5}, \frac{1}{2\,(n+1)}\}$ in the actual implementation. It is easy to see that, if the given objective function is integer, an optimal Steiner tree packing with respect to $\tilde{w}$ is also optimal with respect to ω and vice versa.

Table 1 demonstrates the success of the perturbation trick for the "difficult switchbox" routing problem. Column 1 gives the number of cutting plane iterations, column 2, 3 and 4 (resp. 5, 6 and 7) contain the LP objective value, the number of pivots and the accumulated CPU-time by using the perturbed (resp. original) objective function. The numbers are very impressive, in particular if one considers the last rows. The running time is reduced to less than one tenth of the original time.

iter.	perturbed objective function			original objective function		
	LP value	pivots	CPU-time	LP value	pivots	CPU-time
1	0.000	0	0:53	0.000	0	0:52
2	456.562	1163	3:25	456.850	2646	5:35
3	457.571	420	5:07	457.100	4656	19:49
4	457.589	548	7:28	457.350	5995	40:39
5	457.746	800	10:25	457.417	7001	67:37
6	457.793	1224	14:36	457.417	7657	97:55
7	458.014	3175	24:03	457.515	11367	149:21
8	458.314	2007	31:07	457.748	23718	244:11
9	458.625	2554	40:19	458.149	49393	456:26

Table 1:

Another (poyhedral) preprocessing trick helped to increase the lower bounds and to decrease the running time considerably. After "solving" the trivial initial linear program by setting all variables to zero we do not call our general separation routines; rather, we generate a particular class of Steiner cut and Steiner partition inequalities for which we have heuristic reasons to believe that they form a sensible set of "good" initial cutting planes.

Since the underlying graph is a complete rectangular grid graph, we add all Steiner cut inequalities that are induced by a horizontal or vertical cut. The advantage is that these inequalities have pairwise different support. In addition, for multiterminal nets we extend each Steiner cut inequality to a Steiner partition inequality with right hand side greater than two. For example, let $|T_k| = p \geq 3$, $F = \delta(W)$, $W \subset V$ be a vertical cut that induces a Steiner cut inequality. First, we determine a Steiner partition $W_1, \ldots, W_q$ of W such that $[W_i : W_{i+1}]$ is a horizontal cut in $(W, E(W))$ for $i = 1, \ldots, q-1$ and q is maximal. The only node sets of $W_1, \ldots, W_q$ that possibly contain more than one terminal are W_1 and W_q. For these two node sets we again determine a Steiner partition $W_r^1, \ldots, W_r^{l_r}$ for $r = 1$ and $r = q$ such that $[W_r^i : W_r^{i+1}]$ is a vertical cut in $(W_r, E(W_r))$ and l_r is maximal. The same procedure is applied to the node set $V \setminus W$. Taking both together we obtain (after renumbering) a Steiner partition $W_1, \ldots, W_s$ with $s = p$, and $x(\delta(W_1, \ldots, W_s)) \geq p - 1$ defines a Steiner partition inequality. We extend each horizontal and vertical cut that defines a Steiner cut inequality in this way. Obviously, the resulting inequalities do not necessarily have different support, but the right hand side is quite large. Let us denote all inequalities constructed this way and the Steiner cut inequalities induced by a horizontal or

iter.	with special Steiner part. inequ.		without special Steiner part. inequ.	
	LP-value	CPU-time	LP-value	CPU-time
1	0.000	0:52	0.000	0:52
2	456.562	3:24	394.725	3:29
3	457.574	5:38	397.335	6:23
4	457.741	10:21	401.075	17:05
5	457.862	22:39	407.586	43:05
6	458.070	45:53	416.256	67:43
7	458.551	76:13	423.642	103:58
8	458.983	107:28	428.051	158:42
9	459.615	142:39	431.740	203:22

Table 2:

vertical cut by *special Steiner partition inequalities*.

Table 2 illustrates the progress we obtain by using the special Steiner partition inequalities after solving the initial linear program. Column 1 presents the number of cutting plane iterations. Column 2 and 3 (resp. 4 and 5) give the LP objective value and the accumulated CPU-time by using (resp. not using) the special Steiner partition inequalities after the first iteration. The results are impressive. The lower bound we obtain within three minutes after the second iteration by adding the special Steiner partition inequalities is much better than after running the algorithm with the separation algorithms for the Steiner partition inequalities discussed in section 3 for more than three hours.

Next, we want to deal with the separation of the extended alternating cycle inequalities. The separation algorithms we have outlined in section 3 (the dynamic program as well as the heuristics) need a pair of nets as input. The problem we are concerned with is to choose one (or several) "good" pairs of terminal sets for which we want to execute the separation algorithms. If we would call one of these algorithms for all net pairs, we would obtain a non-acceptable running time, because the number of calls is quadratic in the number of nets.

In order to overcome this problem, we try to exploit the information given by the primal heuristic 4.1. Remember that two components are gradually connected in this heuristic. More precisely, in step (9) it is tried to connect two components via a shortest path. If this is not possible, another net must block this path. Obviously the two nets concurrently prefer certain edges in this case. Moreover, this situation indicates that the information provided by the linear programming solution is too poor to decide which of the nets is forced to make a detour. Hence, we conclude that more inequalities combining these nets are necessary. Thus, we call the separation algorithms for the extended alternating cycle inequalities for nets that are in conflict due to the information of the primal heuristic. Practical experiments have shown that the number of such conflicts is sublinear in the number of nets and that strongly violated extended alternating cycle inequalities can be obtained for such conflicting net pairs.

We want to point out that not only the linear program solution supplies important information for the primal heuristic. But also conversely, the primal heuristic indicates which type of inequalities are promissing for a further execution of the cutting plane algorithm. In our opinion this interplay of the methods for determing the lower and upper bound is essential in order to solve large scaled problems.

Let us now summarize the overall algorithm.

Algorithm 4.3 (Branch and Cut Algorithm for the Switchbox Routing Problem)

Input:

A complete rectangular grid graph $G = (V, E)$ with edge capacities $c_e = 1$ and edge weights $w_e \in \mathbb{N}_0$, $e \in E$; a net list $\mathcal{N} = \{T_1, \dots, T_N\}$ where the terminal sets are on the outer face of G.

Output:

An optimal solution of the weighted Steiner tree packing problem.

Initialization

(1) Determine the perturbed objective function vector $\tilde{w}$ according to (4.2).

(2) Determine critical cuts by applying the algorithm based on Lemma 3.1 (see section 3) and fix the corresponding variables.

(3) Initialize the branching tree with the whole problem.

(4) Solve the following (trivial) linear program

$$\begin{aligned} \min \quad & \tilde{w}^T x \\ & \sum_{k=1}^{N} x_e^k \le c_e, \quad && \text{for all } e \in E. \\ & x_e^k \ge 0, \quad && \text{for all } e \in E,\ k = 1, \dots, N. \end{aligned}$$

(5) Try to determine a feasible solution by applying primal heuristic 4.1.

(6) If a feasible solution was found

set b to the objective function value of the solution.

Else

set $b = \infty$.

(7) Add the special Steiner partition inequalities to the linear program.

Solution and evaluation of the linear program

(8) Determine an optimal solution y of the actual linear program.

(9) If y is the incidence vector of a Steiner tree packing, then
set $b = \tilde{w}^T y$.

(10) Else
try to improve the upper bound b by applying primal heuristic 4.1.

(11) If $\lceil \tilde{w}^T y \rceil = \lceil b \rceil$, then perform the following step:
If there still exists an unsolved subproblem in the branching tree, choose one and goto (8).
Else print the optimal solution corresponding to b, STOP.

(12) Eliminate all inequalities $a^T x \geq \alpha$ with $a^T y > \alpha$ from the actual linear program.

Separation

(13) Determine violated constraints from the "pool" (for an explanation of the *pool* see below) as well as by applying the separation heuristics mentioned in section 3.

(14) If violated constraints are found, add them to the linear program and goto (8).

(15) Try to find violated Steiner partition inequalities and extended alternating cycle inequalities by using the dynamic programs.

(16) If violated constraints are found, add them to the linear program and goto (8).

Branching

(17) Determine a variable index (e, k) with $y_e^k \notin \{0, 1\}$.

(18) Generate two subproblems, one by adding the constraint $x_e^k = 0$ and the other by adding the constraint $x_e^k = 1$.

(19) Add both subproblems to the branching tree.

(20) Choose a subproblem from the branching tree and goto (8).

The cutting plane algorithm itself encloses (up to the initialization) steps (8) to (16). We have embedded this method into a general branch and cut framework developed by M. Jünger (Universität zu Köln). The enumeration scheme is only sketched in steps (4) and (17) to (20). In fact, an efficient implementation of such a scheme is a very difficult and complex task. For more details concerning the branch and cut framework the interested reader is refered to the software package of M. Jünger.

In step (12) we delete all inequalities (up to the capacity and trivial inequalities) that are not satisfied with equality from the linear program, in order to keep the size of the linear program small. The eliminated constraints are stored in a so-called "pool", which is checked during the separation phase.

name	h	b	N	distribution of the nets					ref.
				2	3	4	5	6	
difficult switchbox	15	23	24	15	3	4	1	1	[BP83]
more difficult switchbox	15	22	24	15	3	5		1	[CH88]
terminal intensive switchbox	16	23	24	8	7	5	4		[Lu85]
dense switchbox	17	15	19	3	11	5			[Lu85]
augmented dense switchbox	18	16	19	3	11	5			[Lu85]
modified dense switchbox	17	16	19	3	11	5			[CH88]
pedagogical switchbox	16	15	22	14	4	4			[CH88]

Table 3:

If step (17) is executed, we are sure that there exists an index such that $0 < y_e^k < 1$. This is true, because y is not the incidence vector of a Steiner tree packing and in step (13) the Steiner cut inequalities are exactly separated by our separation algorithms. According to (2.2) the existence of such an index is guaranteed.

Algorithm 4.3 can be used, in principle, to determine an optimal solution of a given switchbox routing problem or to detect that no feasible solution exists. However, it may not be possible to guarantee this in acceptable time. For that reason we povide an option in our algorithm to limit the running time. If this limit is exceeded, the algorithm stops and prints the best lower and upper bound.

5 Computational Results

In this section we report on our computational experiences with the algorithm introduced in section 4. We have tested our algorithm on switchbox routing problems that are discussed in literature. Table 3 summarizes the data. Column 1 presents the name used in literature. In column 2 and 3 the height and width of the underlying grid graph is given. Column 4 contains the number of nets. Columns 5 to 9 provide information about the distribution of the nets; more precisely, column 5 gives the number of 2-terminal nets, column 6 gives the number of 3-terminal nets and so on. Finally, the last column states the reference to the paper the example is taken from.

In all examples as they were originally introduced in literature, the underlying graph is given as follows. The graph is obtained from a complete rectangular grid graph by removing the outer cycle, see Figure 5 (a). Hence, every terminal is incident to a unique edge, and obviously, every Steiner tree must contain this edge. It is easy to see that by contracting all pending edges an equivalent problem is obtained, see Figure 5 (b). The graph resulting this way is a complete rectangular grid graph with terminals on the outer face. This instance is the input to our problem.

The first example "difficult switchbox" was introduced by Burstein and Pelavin. The second one "more difficult switchbox" is derived from the first one by deleting the last column. (More precisely, the edges $[(i, 23), (i, 24)]$ of the first grid graph are contracted for $i = 1, \ldots, 15$ and parallel edges are deleted.) The net list is the same. The difference in the distribution occurs (see column 7 and 8), because an edge whose endpoints belong to the same net is contracted. The third problem instance was introduced by Luk, here each outer face node is occupied by a terminal. The fourth switchbox routing problem is again due to Luk. Up to now it is not known whether a solution for this example exists, if the Manhattan or 2-layer model is used. Based on this example two variants can be obtained. One, called "augmented dense switchbox", has an additional column on the right, the other, called "modified dense switchbox", has an additional column near the middle and an additional row on the buttom. The last example was introduced by Cohoon and Heck. They illustrated their algorithm on this problem.

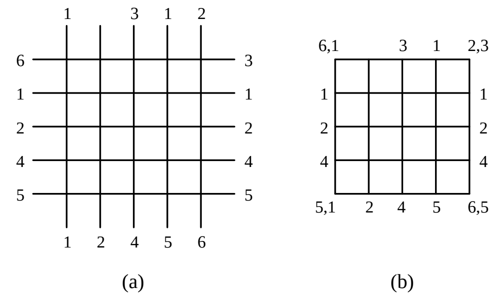

Figure 5:

In all examples the edge weights as well as the edge capacities are equal to one. Unfortunately, the problem instances do not fix the routing model (Manhattan, knock-knee or multiple layer model). To our knowledge all methods from the literature use the Manhattan model or the 2-layer model. The choice of the underlying model strongly influences the solvability of the problems. For example, there may exist a solution in the 2-layer model, whereas it does not in the knock-knee model. Figure 5 illustrates such an example (this example is taken from [CH88]). Moreover, there exist problem instances where shorter connections are possible in the 2-layer model than in the knock-knee model. The same is true

example	variables	fixed variables	remaining variables
difficult switchbox	15648	2224	13424
more difficult switchbox	14952	2450	12502
terminal intensive switchbox	16728	4913	11815
dense switchbox	9082	4831	4251
augmented dense switchbox	10298	2678	7620
modified dense switchbox	9709	4057	5652
pedagogical switchbox	9878	2039	7839

Table 4:

for a comparison of the knock-knee model with the Manhattan model. Thus, a comparison of algorithms for the different models is not possible. So we confine ourselves to report on the results we have obtained by applying our algorithm.

Table 4 informs about the size of the problems and about the success of fixing variables with the algorithm discussed in the last subsection of section 3. Column 2 states the total number of 0/1 variables, column 3 gives the number of fixed variables and the last column contains the number of remaining variables. Table 4 illustrates that many variables can be fixed, for example more than one half of the variables for problem "dense switchbox". Nevertheless, the number of remaining variables is still large (see the last column).

In Table 5 the results we have obtained with our branch and cut algorithm are summarized. Column 2 gives the best feasible solution. The values are not integer due to the perturbed objective function. To obtain the real value with respect to the original objective function the entries must be rounded up. The entries in column 3 are the objective function values of the linear program when no further violated constraints are found, i. e., when branching (steps (17) to (20) in Algorithm 4.3) is performed for the first time. This values are obviously lower bounds for the whole problem. In column 4 the percental derivation of the best solution from the lower bound is given; more precisely, column 4 contains the value $\frac{\lceil \tilde{w}_2 \rceil - \lceil \tilde{w}_3 \rceil}{\lceil \tilde{w}_3 \rceil}$, where $\tilde{w}_2$ (resp. $\tilde{w}_3$) is the corresponding value of column 2 (resp. 3). Column 5 (resp. 6) gives the number of cutting plane iterations (resp. the number of nodes in the branching tree). Finally, the last column reports

example	best sol.	LP value	gap	iter.	B&C	CPU-time
difficult switchbox	463.711	463.709	0.0%	69	3	1564:15
more difficult switchbox	451.712	451.708	0.0%	53	1	983:23
terminal intensive switchbox	536.694	535.196	0.2%	163	13	3755:44
dense switchbox*	440.601	437.579	0.7%	119	4	1017:43
augmented dense switchbox*	468.600	466.006	0.4%	105	1	4561:41
modified dense switchbox	451.585	451.009	0.0%	51	1	387:03
pedagogical switchbox	330.770	330.760	0.0%	77	5	251:58

Table 5:

on the running times. The values are stated in minutes obtained on a SUN 4/50. The two examples "dense switchbox" and "augmented dense switchbox" marked with a footnote are stopped after the time given in the last column, because no further progress could be achieved. We claim that the values given in column 2 are optimal, but we are yet not able to prove this with the cutting plane algorithm. All other problem instances are solved to optimality.

The numbers in Table 5 are quite encouraging. For all problem instances the lower bound in column 3 guarantees that the best feasible solution deviates at most 0.7% from the optimal solution. In our opinion the main advantage of our algorithm is that the quality of an heuristically determined solution can be evaluated with the lower bound. Especially, for problem instances arising in VLSI-Design, where in general only heuristics are at hand, a cutting plane algorithm helps in analyzing the heuristics and simultanously delivers a lot of knowledge about the problem itself.

Nevertheless, one major problem with our algorithm is its running time. The numbers in the last column of Table 5 are very high. One reason is that we are interested in an optimal solution or at least in the best lower and upper bound for each of the problems that we can achieve with our approach. This is time consuming. In practice, it often suffices to find a solution of a predefined quality guarantee. From this point of view, we have analyzed our results also. Table 6 presents the time (measured in minutes), after which the lower bound deviates

example	5%	2%	1%	0%
difficult switchbox	3:24	3:24	90:12	688:49
more difficult switchbox	3:20	3:20	38:19	530:11
terminal intensive switchbox	5:44	83:24	239:10	–
dense switchbox*	2:00	2:00	103:07	–
augmented dense switchbox*	2:04	2:04	269:20	–
modified dense switchbox	2:04	2:04	2:04	387:03
pedagogical switchbox	1:46	2:27	15:04	117:55

Table 6:

at most 5, 2, 1 and, if obtained, 0 percent from the best feasible solution.

It can be seen from column 2 that, for all problem instances, the lower bound deviates from the best feasible solution by at most 5% percent after no more than 6 minutes. Table 6 illustrates in addition that the amount of time increases strongly to obtain a quality below one percent.

In our opinion the times in column 1 of Table 6 are acceptable. However, we would like to point out that these examples are quite small in comparison to problem sizes arising in other practical applications for the design of electronic circuits. Our long-term goal is to apply the branch and cut algorithm to large scale problem instances, too. In order to achieve this, we surely must reduce the running times. We have analyzed our algorithm concerning the question where most of the time is spent. It turns out that about 90% percent of the time is used to solve the linear programs. To our present knowledge two possibilities arise to overcome this problem.

1. Reducing the number of variables.

   We consider the problem only on a subset of the set of variables, solve the problem on this subset and check whether this solution is also optimal for the whole problem. If not, we add some variables and solve the extended problem again. This method is commonly used to solve large scaled practical problems by a cutting plane algorithm (see, for instance, [GH91], [PR91]).

2. Decompose the linear programs.

   The constraint matrices of our problems are of very special structure. Due to this structure it seems to be promising to decompose the linear program. Methods for deomposing linear programs were suggested by Dantzig and Wolfe [DaW60] or by Benders [B62]. Up to now these methods are not used in practice, because the problems can be solved faster directly. However, with the help of parallel computers these methods may get competitive, especially for our problem instances.

Conclusion

In this paper we have developed a cutting plane algorithm for the Steiner tree packing problem. We have introduced some separation methods for special problem instances where the underlying graph is planar and all terminal sets lie on the outer face of the graph. This special instances include an important subproblem in VLSI-Design, the so-called switchbox routing problem. We have reported on computational results we have obtained with our branch and cut algorithm for this type of problems. The results are encouraging. Most of the problems discussed in literature are solved to optimality. Thus, we have good hopes that this approach may also be applicable to large scale problem instances as they occur in practice. To achieve this long-term goal there surely remain a lot of problems to be solved.

References

[B62] J. F. Benders: *Partitioning procedures for solving mixed-variables programming problems*, Numerische Mathematik 4, 1962, 238 – 252.

[BB84] M. L. Brady, D. J. Brown: *VLSI routing: Four layers suffice*, in: F. P. Preparata (ed.): "Advances in Computing Research", Bd. 2: VLSI theory, Jai Press, London, 1984, 245 – 258.

[BP83] M. Burstein, R. Pelavin: *Hierarchical wire routing*, IEEE Transactions on Computer-Aided-Design CAD-2, 1983, 223 – 234.

[BLP87] M. O. Ball, W. Liu, W. R. Pulleyblank: *Two terminal Steiner tree polyhedra*, Report No. 87466-OR, Institut für Ökonometrie und Operations Research, Universität Bonn, Bonn, 1987.

[CH88] J. P. Cohoon, P. L. Heck: *BEAVER: A computational-geometry-based tool for switchbox routing*, IEEE Transactions on Computer-Aided-Design CAD-7, 1988, 684 – 697.

[CR88a] S. Chopra, M. R. Rao: *The Steiner tree problem I: Formulations, compositions and extension of facets*, Technical Report No. 88-82, Graduate School of Business Administration, New York University, New York, 1988.

[CR88b] S. Chopra, M. R. Rao: *The Steiner tree problem II: Properties and classes of facets*, Technical Report No. 88-83, Graduate School of Business Administration, New York University, New York, 1988.

[DW71] S. E. Dreyfus, R. A. Wagner: *The Steiner problem in graphs*, Networks 1, 1971, 195 – 207.

[DaW60] G. B. Dantzig, P. Wolfe: *Decomposition principle for linear programs*, Operations Research 8, 1960, 101 – 111.

[EMV87] R. E. Erickson, C. L. Monma, A. F. Veinott: *Send-and-split method for minimum concave-cost network flows*, Mathematics of Operations Research 12, 1987, 634 – 664.

[F91] M. Fischetti: *Facets of two Steiner arborescence polyhedra*, Mathematical Programming 51, 1991, 401 – 419.

[GH91] M. Grötschel, O. Holland: *Solution of large-scale symmetric travelling salesman problems*, Mathematical Programming 51, 1991, 141 – 202.

[GJ77] M.R. Garey, D.S. Johnson: *The rectilinear Steiner tree problem is $\mathcal{NP}$-complete*, SIAM J. Appl. Math. 32, 1977, 826 – 834.

[GM90] M. Grötschel, C. L. Monma: *Integer polyhedra associated with certain network design problems with connectivity constraints*, SIAM Journal on Discrete Mathematics 3, 1990, 502 – 523.

[GMS90] M. Grötschel, C. L. Monma, M. Stoer: *Computational results with a cutting plane algorithm for designing communication networks with low-connectivity constraints*, Operations Research 40, 1992, 309 – 330.

[GMW92] M. Grötschel, A. Martin, R. Weismantel: *Packing Steiner trees: polyhedral investigations*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 92-8, 1992.

[GMW93] M. Grötschel, A. Martin, R. Weismantel: *Packing Steiner trees: separation algorithms*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 93-2, 1993.

[K72] R.M. Karp: *Reducibility among combinatorial problems*, in: R.E. Miller, J.W. Thatcher (eds.), "Complexity of Computer Computations", Plenum Press, New York, 1972, 85 – 103.

[KL84] M.R. Kramer, J. van Leeuwen: *The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits*, F.P. Preparata (ed.), "Advances in Computing Research", Bd. 2: VLSI theory, Jai Press, London, 1984, 129 – 146.

[Lu85] W. K. Luk: *A greedy switch-box router*, Integration 3, 1985, 129 – 149.

[Li84] W. Lipski: *On the structure of three-layer wireable layouts*, F. P. Preparata (ed.): "Advances in Computing Research", Bd. 2: VLSI theory, Jai Press, London, 1984, 231 – 244.

[M92] A. Martin: *Packen von Steinerbäumen: Polyedrische Studien und Anwendung*, Ph.D. Thesis, Technische Universität Berlin, 1992.

[PR91] M. Padberg, G. Rinaldi: *A branch and cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Review 33, 1991, 60 – 100.

[S87] M. Sarrafzadeh: *Channel-routing problem in the knock-knee mode is $\mathcal{NP}$-complete*, IEEE Transactions on Computer-Aided-Design CAD-6, 1987, 503 – 506.