Packing Paths and Steiner Trees: Routing of Electronic Circuits

A. Martin and R. Weismantel
Konrad-Zuse-Zentrum Berlin
Heilbaneer Staße 10
D-10711 Berlin-Wilmersdorf
Germany
email: martin@zib-berlin.de
email: weismantel@zib-berlin.de

One of the challenging problems in the design of electronic circuits is the so-called routing problem. Roughly speaking, the task is to connect so-called terminal sets via wires on a predefined area. In addition, certain design rules are to be taken into account and an objective function such as the wiring length must be minimized. The routing problem in general is too complex to be solved in one step. Depending on the user's choice of decomposing the chip design problem into a hierarchy of stages, on the underlying technology, and on the given design rules, various subproblems arise. We discuss several variants of practically relevant routing problems and give a short overview on the underlying technologies and design rules. Many of the routing problems that come up this way can be formulated as the problem of packing so-called Steiner trees in certain graphs. We consider the Steiner tree packing problem from a polyhedral point of view and present three possibilities to define an appropriate polyhedron. Weighing their pros and cons we decide for one of these polytopes and sketch some of our investigations.

1. Introduction

Electronic circuits are – not least due to the incredible improvements in the last decades – one of the backbones of today's technology. For example, modern automatic control technology, manufacture or communication systems are simply inconceivable without electronic control. An electronic circuit is a complex connexion of semi-conductor elements (so-called transistors). This connexion is the physical realization of a logic function. Today it is possible to integrate hundreds of thousands or even millions of transistors on a few square centimeters (*V*ery *L*arge *S*cale *I*ntegration). The complexity and the large scale of the problems arising in the design of such circuits provide a great challenge to those interested in integrated system design. In fact, the involved problems touch the

fields of computer science, engineering and mathematics. A number of these problems can be modelled as combinatorial optimization problems, and thus, solution methods of this field are applicable. Several versions of the so-called *routing problem* belong hereto. Roughly speaking, the routing problem can be formulated as follows:

> Let a certain area (typically a rectangle with some "forbidden zones") and a list of sets of points be given. The routing problem is to connect each set of points by wires on the area such that certain technical side constraints are met and some objective function is optimized.

Each set of points is called a *net* and every single point a *terminal*. The routing problem strongly depends on the chosen technology, the design rules and the customer requirements. For example, the design rules restrict the *routing area* (i.e. the area that is available for connecting the nets) or prescribe the distance two different wires must stay apart. We will discuss these issues in more detail in Section 2. It turns out that the routing problem is enourmously complicated. At least at present, it seems impossible to solve it in one step for realistic problem instances. In practice, the routing problem is usually decomposed into two subproblems. In a first step, one determines how the wires of each net maneuver around the obstacles in the routing area (*global routing*). Here, the design rules are taken into account only to some extent. The second phase (*detailed routing*) consists in finding the detailed routes for each net that comply with the global routes and that obey the design rules exactly.

Many routing problems that arise in this decomposition can be formulated using graph theory. One way of introducing a graph $G = (V, E)$ is to define nodes for subareas of the whole routing area, and to link nodes that represent adjacent subareas by an edge. In addition, we assign capacities to the edges or nodes, respectively. The nets are represented in this graph by subsets of the node set. In graph-theoretic terms each route of a net is called a Steiner tree. The problem of routing N nets reduces to the problem of finding (packing) N Steiner trees in G that meet the capacity constraints. We call this problem the Steiner tree packing problem. We will discuss various types of Steiner tree packing problems that arise in VLSI- design in Section 3.

Our approach to the Steiner tree packing problem is to consider it from a polyhedral point of view and to use linear programming techniques. We define a polyhedron P whose vertices correspond uniquely to the solutions (Steiner tree packings) in the graph. In Section 4 we discuss several ways to define an appropriate polyhedron P and weigh the pros and cons. What we need for the application of linear programming techniques is a complete or at least "good" description of the polyhedron P by means of inequalities. We will demonstrate the inherent complexity of this task on some small examples. The inequalities we found form the basis for the development of a cutting plane algorithm. We have implemented a cutting plane algorithm for special instances of the routing

problem, so-called switchbox routing problems, and achieved quite good results for many benchmark examples discussed in the literature.

2. The layout of electronic circuits

The design of electronic circuits is typically a two-phase process. At the beginning, a description of a task the circuit to be designed must perform is given. Such a task is a complex logical function which consists of many elementary logic operations (for example, the logic "and"-operation). Usually several of these elementary logic operations are combined into a logical unit (for example an adder). In the *logical design phase* it is specified which of these predefined logical units are to be used, and it is determined which of the chosen logical units must be connected by wires so that the chip performs in the way it should. The logical units are also called *cells*. Each cell is characterized by its width, its height, its contact points (so-called *terminals*) and its electric properties. A *net* is a set of terminals that must be connected by a wire (as specified in the logical design phase). The list of cells and the list of nets are the input of the second phase, the *physical design*. Here, the task is to assign the cells to a certain rectangular area (silicon) and connect (route) the nets by wires. The rectangular area is usually subdivided into an inner part (called *master*) and an outer part. The set of cells consists of *logic cells* and *input/output cells*. Logic cells must be assigned to locations on the master, whereas input/output cells are to be placed on the outer part. In fact, the physical design problem is more complicated, since certain design rules have to be taken into account and an objective function is to be minimized. The design rules strongly depend on the given layout style and specify, for instance, the distance two nets must stay apart, whether certain cells are preassigned to certain locations and so on. This applies especially to the objective function. In practice, the following layout styles are of particular interest.

1. General cells
   In this layout style the cells are of arbitrary rectangular shape with a few exceptions such as L-shapes. A cell can be placed anywhere on the master (see Figure 1). The goal here is to place the cells and route the nets such that the resulting area is minimized.

2. Standard cells
   Here, the master is subdivided into a placement and routing area. The placement area consists of a set of (parallel) rows of equal height. The cells are rectangular of identical height, but they may differ in their width (see Figure 2). The cells must be assigned to the rows such that the longest row is minimized or the overall length of the wires is minimized. The nets are routed in the channels lying between the rows.

3. Gate arrays
   In contrast to the above layout styles the size of the master is fixed. Again, a subdivision into a placement and routing area is given a priori.

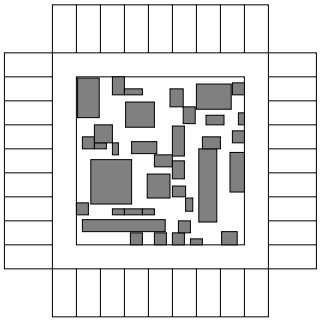


FIGURE 1.

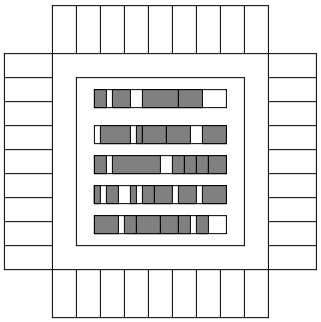


FIGURE 2.

The placement area consists of so-called *base cells* arranged in form of a matrix. The cells are rectangular and the width (height) of a cell is a multiple of the width (height) of a base cell. The routing takes place on the routing area which is given in advance (see Figure 3).

4. Sea-of-cells
   The only difference to the gate array layout style is that the master does not contain a subdivision into a placement and routing area. The whole master is subdivided into base cells in form of a matrix (see Figure 4). The routing area is composed of those base cells that are not occupied by the placed cells.

For the first two layout styles the primary goal is to minimize the whole area of the master, whereas for the other two styles the routability, i.e., the problem of placing the cells such that there exists a feasible solution to the routing problem, is the center of interest. However, routability can hardly be measured and expressed in form of an objective function. Thus, minimizing the total length of all routes is very often used instead. Another heuristic reason for minimizing the routing length (also in case of the first two layout styles) is

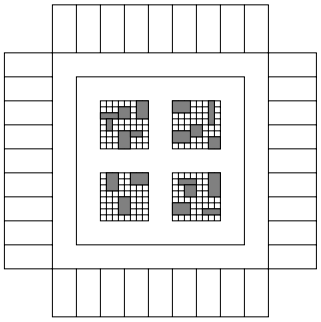


FIGURE 3.

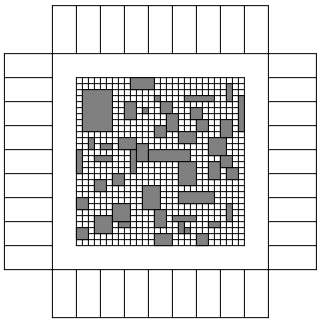


FIGURE 4.

that an electronic circuit with small routing length usually needs little area on the whole. Thus, minimizing the overall area is (somehow) implicitly taken into account by minimizing the routing length.

Any reasonably precise version of the physical design problem is $\mathcal{NP}$-hard, even very simple ones. Moreover, most real world problem instances involve several thousands of cells and nets, so that today's algorithmic knowledge makes it very improbable that they can be solved to optimality. Therefore, usually heuristic decomposition into subproblems is applied. The first subproblem consists of finding appropriate locations for the cells (*placement problem*). Subsequently, the nets must be realized by wiring the appropriate terminals (*routing problem*) and finally, a compaction step is performed if required. This process is iterated with different parameters if the final result is not satisfactory.

For the remainder of this paper we will focus on the routing problem in more detail.

3. The routing problem

There is given a list of nets. Each net consists of a set of terminals. The terminals specify the points at which wires have to contact the cells. The routing problem is to connect the nets by wires on the routing area subject to technical side constraints which depend on the given layout style. Most frequently, the objective is to minimize the overall wiring length or to minimize the length of the longest wire.

We say a net is *routed* if its terminals are connected by (electric) wires. We speak of a *k-terminal net*, if k is the number of terminals of the net. If $k > 2$, the term *multiterminal net* is often used. In the following we will not distinguish between a net and the route of a net, if this does not lead to confusions.

The routing itself takes place on so-called *layers*. If some net changes a layer, a hole, called *via*, must be drilled. Usually, each layer is subdivided into horizontal and vertical lines, so-called *tracks* to which the wires of the nets must be assigned. If there does not exist such a division into tracks we speak of a *free* or *grid- free routing*. Further side constraints include, for instance, the distance to nets must stay apart from each other, how long two different nets may run on top of each other on two different layers or that some wires must not exceed a certain length.

In practice, the routing problem itself is decomposed because of its inherent complexity and large scale. In the global routing phase the homotopy of the nets is determined, i.e., it is determined how the wires "maneuver around the cells". Thereafter, in the detailed routing phase the wires are assigned to the layers and tracks according to the homotopy specified in the global routing step. We consider both routing phases in more detail now. Before doing so, let us fix some graph-theoretic notation.

We denote graphs by $G = (V, E)$, where V is the node set and E the edge set. All graphs we consider are undirected and finite. For a given edge set $F \subseteq E$, we denote by $V(F)$ all nodes that are incident to an edge in F. We call a sequence of nodes and edges $K = (v_0, e_1, v_1, e_2, \ldots, v_{l-1}, e_l, v_l)$, where each edge e_i is incident with the nodes v_{i-1} and v_i for $i = 1, \ldots, l$, and where the edges are pairwise disjoint and the nodes distinct (except possibly v_0 and v_l), a *path from v_0 to v_l*, if $v_0 \neq v_l$, and a *cycle*, if $v_0 = v_l$ and $l \geq 2$. We call a graph G a *complete rectangular $h \times b$ grid graph*, if it can be embedded in the plane by h horizontal lines and b vertical lines such that the nodes of V are represented by the intersections of the lines and the edges are represented by the connections of the intersections. A *grid graph* is a graph that is obtained from a complete rectangular grid graph by deleting some edges and removing isolated nodes (i.e. nodes that are not incident to any edge).

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 in G*, if the subgraph $(V(S), S)$ contains a path from s to t for all pairs of nodes $s, t \in T, s \neq t$. Following the notation in VLSI-design we call T a *terminal set* or a *net* and each $t \in T$ a *terminal*. "Routing some net T in a graph G" means in graph-theoretic terms, "finding a Steiner tree for T in

G". We will use both manners of speaking in the following.

3.1. Global routing

The global routing problem is usually modelled as a graph- theoretic problem. Hereto, the routing area is subdivided into subareas and these are represented by nodes or edges in a graph. Of course, there are many ways to do this. One possible way of subdividing the routing area is illustrated in Figure 5. 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 5). 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 $c_{uv} \in \mathbb{N}$ 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. Every terminal of a net is assigned to that node, whose corresponding subarea contains the terminal or is as close as possible to the position of the terminal. The global routing problem consists in routing all nets in G (or in graph-theoretic terms, finding a Steiner tree for each terminal set) such that the capacity constraints are satisfied and the total wiring length (that is the sum of the weights of the Steiner trees) is as small as possible.

After having solved the global routing problem every subarea that corresponds

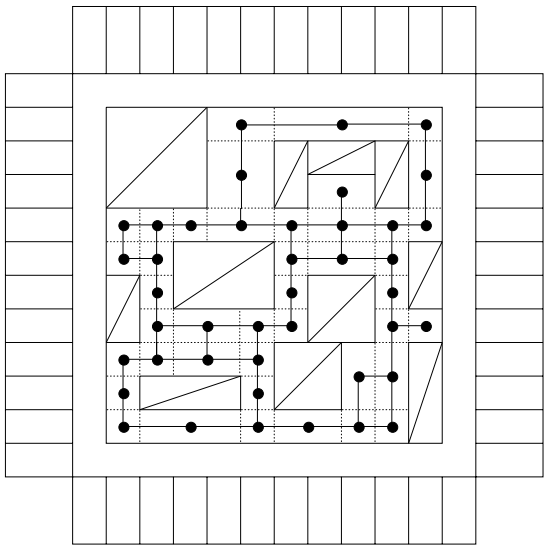


FIGURE 5.

to a node in the global routing graph must now be routed in detail. This is the topic of the next subsection.

3.2. Detailed routing

The number of different detailed routing models which are studied in the literature or which are used in practice is tremendous. Usually, the problems coming up are formulated in a grid graph. We restrict ourselves to this case, too. The reader interested in grid-free routing models is refered to [11] where an excellent overview on all different kinds of models is given.

The detailed routing problems can be classified according to two criteria which are independent from each other. We introduce these classifications now without claiming to be complete. Again, for more details we refer to [11].

1. The detailed routing problems are distinguished according to the shape of the routing area and the locations of the terminals. As mentioned before the nodes in the global routing graph represent subareas of the whole routing area. Depending on the subdivision different shapes of detailed routing areas arise. At the end of the global routing phase it is known which nets go across which subareas. Suppose, some net crosses the border of two adjacent subareas (depicted by dotted lines in Figure 5). Of course, from the information of the global routing solution it is not clear at which point the net meets the border. Each such crossing point is interpreted as a "pseudo"-terminal. In order to solve the routing problems for each of these subareas independently locations for the pseudo-terminals must be determined. This usually is done by applying heuristics. Concerning the shape of the routing area and the locations of the terminals the following detailed routing models are of particular interest in practice.

   (a) (Channel routing) Here, we are given a complete rectangular grid graph. The terminals of the nets are exclusively located on the lower and upper border (see Figure 6). It is possible to vary the height (= number of horizontal tracks) of the channel. Hence, the size of the routing area is not fixed in advance.

   (b) (Switchbox routing) Again, we are given a complete rectangular grid graph. The terminals may be located on all four sides of the grid graph (see Figure 7). Thus, the size of the routing area is fixed.

   (c) (General routing) In this case, an arbitrary grid graph is considered. The terminals are located at any hole of the grid (see Figure 8). In contrast to the first two models, the homotopy of the nets is no longer trivial and has to be taken into account.

2. The detailed routing problems are distinguished to which extent the layers are taken into account when the wires of the nets are assigned to the tracks.

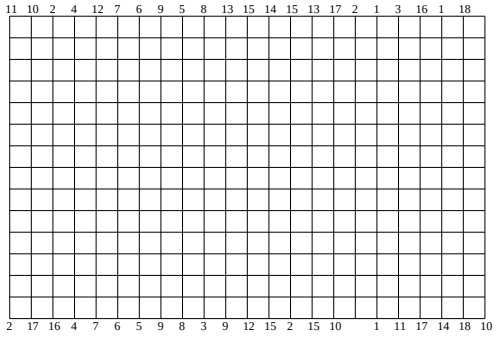


FIGURE 6.

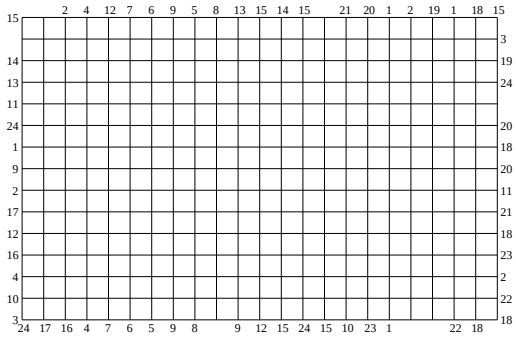


FIGURE 7.

(a) (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.

(b) (Planar model) This is a special case of the multiple layer model where $k = 1$, that is we are given a (planar) grid graph and we are looking for node disjoint connections of the nets. This model is very restrictive, since only one layer is available. Thus, only few practically relevant routing problems can be modelled this way.

(c) (Manhattan model) Given some (planar) 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 9) are not allowed. This restriction guarantees that the resulting routing can be laid out on

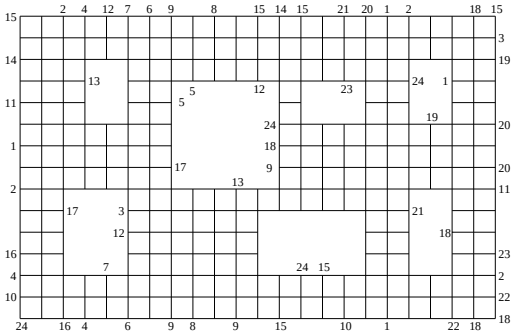


FIGURE 8.

two layers at the possible expense of causing long detours.

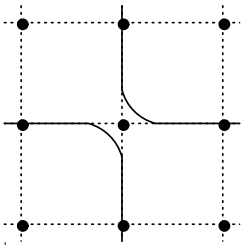


FIGURE 9.

(d) (Knock-knee model) Again, some (planar) 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 [1] have designed an algorithm that guarantees that any solution in this model can be routed on four layers. It was shown in [12] that it is $\mathcal{NP}$-complete to decide whether a realization on three layers is possible.

The models coming out of these two kinds of classifications can be combined in all possible ways. For example, combining 1 (b) and 2 (d) we obtain a switchbox routing problem in the knock-knee model, or in graph-theoretic terms, the problem of finding edge disjoint Steiner trees in a complete rectangular grid graph, where all terminals are located on the outer face. Moreover, depending on the model different objective functions are considered. Possible objective functions are

- minimizing the routing area,

- minimizing the routing length,
- minimizing the number of vias.

Minimizing the routing area is typically the objective in channel routing problems, whereas the routing length is usually minimized, if the routing area is fixed in advance. Optimizing the number of vias is rarely considered in detailed routing algorithms, but most frequently addressed in a postprocessing step.

It is not surprising that most of these routing problems are $\mathcal{NP}$-hard. Even the problem of finding a (with respect to some weighting of the edges) minimum Steiner tree in a graph G for some terminal set T is $\mathcal{NP}$-hard (see [8, 3]). These tremendous difficulties lead to further specializations of the routing problem. For example, routing problems are frequently studied with the additional restriction that all terminal sets have cardinality two, i.e., multiterminal nets are not allowed. In graph-theoretic terms, this means we are looking for disjoint paths in a graph (possibly of minimal length). Though the problem remains $\mathcal{NP}$-hard in general, it is – at least in some special cases – tractable more easily. Investigations for the disjoint path problem have an impact on the solution of practically relevant cases, because most of the nets (about up to 60%) are 2-terminal nets in real world applications.

Summing up, our attention in this section was to give an impression on the huge variety of routing problems that are worth being studied. We have indicated that, at least at the present state of knowledge, it seems impossible to handle the whole routing problem in one step. In the next section we present a model that is applicable to the global routing problem and the switchbox routing problem in the knock-knee model and attack it from a polyhedral point of view.

4. A POLYHEDRAL APPROACH

We are given an undirected graph $G = (V, E)$ with edge capacities $c_e \in \mathbb{N}$ for all $e \in E$ and a net list $\mathcal{N} = \{T_1, \ldots, T_N\}$, $N \in \mathbb{N}$. The *Steiner tree packing problem* consists in finding 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*. We refer to an instance of the Steiner tree packing problem by the tripel $(G, \mathcal{N}, c)$. The idea of a polyhedral approach for the (weighted) Steiner tree packing problem is the following. We define a polyhedron P_I whose vertices are in one-to-one correspondence to the Steiner tree packings in the graph. In order to apply linear programming techniques we need a description of this polyhedron by means of equations and inequalities. The number of such inequalities is usually exponential in the size of the input. A general approach to overcome this difficulty is to apply a cutting plane algorithm: Start with a small set of valid inequalities. These inequalities

define a polyhedron P' that contains P_I. 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 P_I 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 P_I and exact separation routines are not always available. If we intend to find an optimal solution of the problem we must embed the procedure into an enumeration scheme. This whole method is commonly known as a *branch and cut algorithm.* In this section we want to define an appropriate polyhedron for developing a polyhedral approach to the (weighted) Steiner tree packing problem. Indeed, there are many possible ways to define such a polyhedron. Here, we will present three of these possiblities and discuss some of their properties. Before going into detail, let us briefly introduce some notation that will be used throughout this section. We denote by $\mathbb{R}^E$ the vector space where the components of each vector are indexed by the elements of E, i.e., $x = (x_e)_{e \in E}$ for $x \in \mathbb{R}^E$. For an edge set $F \subseteq E$, we define the incidence vector $\chi^F \in \mathbb{R}^E$ by setting $\chi^F_e := 1$, if $e \in F$, and $\chi^F_e := 0$, otherwise. Furthermore, we abbreviate $\sum_{e \in F} x_e$ by $x(F)$ for an edge set F and a vector $x \in \mathbb{R}^E$. We denote by $\mathbb{R}^{\mathcal{N} \times E}$ the $N \cdot |E|$ – dimensional vector space $\mathbb{R}^E \times \ldots \times \mathbb{R}^E$. The components of a vector $x \in \mathbb{R}^{\mathcal{N} \times E}$ are indexed by x^k_e for $k \in \{1, \ldots, N\}$, $e \in E$. For a vector $x \in \mathbb{R}^{\mathcal{N} \times E}$ and $k \in \{1, \ldots, N\}$ we denote by $x^k \in \mathbb{R}^E$ the vector $(x^k_e)_{e \in E}$. If it is clear from the context we will abbreviate a vector $x = ((x^1)^T, \ldots, (x^N)^T)^T$ by $(x^1, \ldots, x^N)$. By the *incidence vector of a Steiner tree packing* $S_1, \ldots, S_N$ we mean the vector $(\chi^{S_1}, \ldots, \chi^{S_N})$.

A canonical formulation

A "natural" model for the (weighted) Steiner tree packing problem is obtained by introducing a variable for every edge of the underlying graph and every net. More precisely, we consider the $N \cdot |E|$ – dimensional vector space $\mathbb{R}^{\mathcal{N} \times E}$ and we introduce a variable x^k_e for every $e \in E$ and $k \in \{1, \ldots, N\}$ with the interpretation

$$x^k_e := \begin{cases} 1, & \text{if edge } e \text{ is contained in the Steiner tree for } T_k, \\ 0, & \text{otherwise.} \end{cases}$$

The *Steiner tree packing polyhedron* $\mathrm{STP}(G, \mathcal{N}, c)$ is the convex hull of all incidence vectors of Steiner tree packings. It is easy to see that the following holds.

$$
\begin{aligned}
\mathrm{STP}\,(G,\mathcal{N},c) = \mathrm{conv}\,\{x \in \mathbb{R}^{\mathcal{N}\times E} \mid \\
\text{(i)}\quad & \sum_{e\in\delta(W)} x_e^k \geq 1, \text{ for all } W \subset V,\ W \cap T_k \neq \emptyset, \\
& \qquad\qquad (V \setminus W) \cap T_k \neq \emptyset,\ k = 1,\ldots,N; \\
\text{(ii)}\quad & \sum_{k=1}^{N} x_e^k \leq c_e, \quad \text{for all } e \in E; \\
\text{(iii)}\quad & 0 \leq x_e^k \leq 1, \quad \text{for all } e \in E,\ k = 1,\ldots,N; \\
\text{(iv)}\quad & x_e^k \in \{0,1\}, \quad \text{for all } e \in E,\ k = 1,\ldots,N\};
\end{aligned}
\tag{4.1}
$$

where $\delta(W)$ denotes the set of edges with exactly one endnode in W, for $W \subseteq V, \emptyset \neq W \neq V$. Clearly, every incidence vector of a Steiner tree packing corresponds to a vertex of the polyhedron $\mathrm{STP}\,(G,\mathcal{N},c)$. Conversely, every vertex of (4.1) is the incidence vector of a Steiner tree packing.

The weighted Steiner tree packing problem can be solved via the following linear program

$$
\min_{x\in\mathrm{STP}(G,\mathcal{N},c)} \sum_{k=1}^{N}\sum_{e\in E} w_e x_e^k,
$$

where w_e corresponds to the weight of edge e.

One "nice" property of this polyhedron is that under some mild assumptions every facet-defining inequality of the polytope $\mathrm{STP}\,(G,\{T_k\},c)$ $(k \in \{1,\ldots,N\})$ defines a facet of the polytope $\mathrm{STP}\,(G,\mathcal{N},c)$. This property was shown in [4] and offers the opportunity to apply results for the Steiner tree polyhedron from the literature.

For real world instances as they appear for the design of electronic circuits the number of variables $N\cdot|E|$ used in Formulation (4.1) tends to several millions. This disadvantage made us think about an alternative model which we will discuss now.

4.1. A "packed" formulation

Instead of using the $N\cdot|E|$ variables introduced before, we associate with every edge e of the graph just one variable x_e which counts the number of Steiner trees that use edge e. We set

$$
\begin{aligned}
\mathrm{STP}_p(G,\mathcal{N},c) := \mathrm{conv}\,\{x \in \mathbb{R}^E \mid \\
\text{(i)}\quad & x_e = \sum_{k=1}^{N} \chi_e^{S_k}, \text{ for all } e \in E; \\
\text{(ii)}\quad & S_k \text{ is a Steiner tree for } T_k \text{ in } G \\
& \text{for all } k = 1,\ldots,N; \\
\text{(iii)}\quad & 0 \leq x_e \leq c_e, \text{ for all } e \in E; \\
\text{(iv)}\quad & x_e \in \mathbb{Z}, \text{ for all } e \in E\}.
\end{aligned}
\tag{4.2}
$$

Obviously, the vertices of (4.2) correspond to the feasible solutions of the Steiner tree packing problem and vice versa. Hence, we can solve the weighted Steiner tree packing problem via the following linear program

$$\min_{x \in \mathrm{STP}_p(G,\mathcal{N},c)} \sum_{e \in E} w_e x_e, \tag{4.3}$$

where w_e corresponds to the weight of edge e.

The problem with this model is that it is very indirect. The "packed" polytope $\mathrm{STP}_p(G,\mathcal{N},c)$ is defined via the "unpacked" model by simply aggregating the incidence vectors. In fact, we do not know – at present – an integer program that is equivalent to (4.3) and that does not use this aggregation trick. It would be interesting to find an explicit IP formulation of (4.3) where only the variables x_e are used.
There is a further problem with the polytope $\mathrm{STP}_p(G,\mathcal{N},c)$. Namely, suppose we are given a vertex x^* of this polytope. Then, by definition, there are Steiner trees $S_1,\ldots,S_N$ such that $x^* = \sum_{k=1}^{N} \chi^{S_k}$ holds. How can we find such Steiner trees in polynomial time? There are some subtilities involved in posing this question correctly in the usual framework of complexity theory. But we do not want to go into these details here.

A study of the relationship between the polytopes (4.1) and (4.2) was addressed by MARTIN [14]. In particular, he showed that in case where the capacities on the edges are neglected ($c = \infty$) a complete description of $\mathrm{STP}\,(G,\,\mathcal{N},\,\infty)$ is given by the facets of the single Steiner tree polyhedra $\mathrm{STP}\,(G,\,\{T_k\},\,\infty)$ for $k = 1,\ldots,N$. This situation does not hold for $\mathrm{STP}_p(G,\,\mathcal{N},\,\infty)$. Indeed, there do exist facet-defining inequalities $a^T x \geq \alpha$ for $\mathrm{STP}_p(G,\,\mathcal{N},\,\infty)$, but, for every $k = 1,\ldots,N$, the inequality $a^T x \geq \alpha$ is not even valid for $\mathrm{STP}_p(G,\{T_k\},\infty)$.

Taking all these observations into account, we expect nearly unsurmountable difficulties, if we try to solve the weighted Steiner tree packing problem by first solving the linear program $\min_{x \in \mathrm{STP}_p(G,\mathcal{N},c)} \sum_{e \in E} w_e x_e$, and, subsequently, unpack the optimal point x^* of the linear program. A use of (4.2) seems to be sensible only if, due to particular structures, unpacking is possible in polynomial time.

An explicit formulation

A third possibility to define a polyhedron associated with the (weighted) Steiner tree packing problem is based on the following ideas.

For every edge set that defines a Steiner tree for a set of terminals, we introduce a variable. For $k = 1,\ldots,N$, set $\mathcal{S}_k := \{S \subseteq E \mid S \text{ is a Steiner tree for } T_k \text{ in } G\}$. For ease of notation we number the elements of $\mathcal{S}_k$ such that $\mathcal{S}_k = \{S_k^1,\ldots,S_k^{s_k}\}$ where s_k corresponds to the cardinality of $\mathcal{S}_k$. Every variable $x_{k,i}$, for $k = 1,\ldots,N,\ i = 1,\ldots,s_k$, is interpreted as follows:

$$x_{k,i} := \begin{cases} 1, & \text{if Steiner tree } S_k^i \text{ is chosen,} \\ 0, & \text{otherwise.} \end{cases}$$

Under these assumptions, let us consider the following polyhedron.

$$
\begin{aligned}
\mathrm{STP}_e(G,\mathcal{N},c) := \mathrm{conv}\,\{x \in \mathbb{R}^{\sum_{k=1}^{N} s_k} \mid \\
\text{(i)} \quad & \sum_{i=1}^{s_k} x_{k,i} = 1, \text{ for all } k = 1,\ldots,N; \\
\text{(ii)} \quad & \sum_{k=1}^{N} \sum_{\{i \mid e \in S_k^i\}} x_{k,i} \le c_e, \text{ for all } e \in E; \\
\text{(iii)} \quad & x_{k,i} \in \{0,1\}, \text{ for } i = 1,\ldots,s_k,\ k = 1,\ldots,N\}.
\end{aligned} \tag{4.4}
$$

Obviously, every vertex of $\mathrm{STP}_e(G,\mathcal{N},c)$ corresponds to a Steiner tree packing and vice versa. Hence, the weighted Steiner tree packing problem can be solved via optimizing the linear objective function over the polyhedron (4.4).

The main drawback of this kind of formulation is – of course – the number of variables involved. The numbers s_k are in general exponential in the size of the input (i.e. the encoding length of the graph, the netlist and the capacity vector). Hence, even solving the linear relaxation will probably cause enormous difficulties. In order to solve the linear relaxation column generation methods must be applied. Here, the idea is to start with a small number of variables (i.e. columns) and solve the corresponding linear program. Subsequently, a pricing step is performed and based on the reduced costs columns are added. This scheme is iterated until the optimal solution of the linear relaxation is obtained.

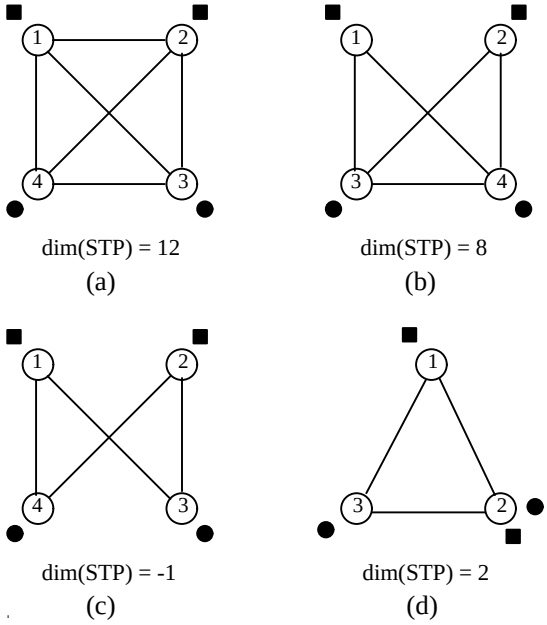
Formulation (4.4) has been considered in several papers (see, for instance, [2, 13, 15]), but, to our knowledge, there have been no serious investigations of the facial structure of this polytope.

Summing up our discussions, the canonical formulation seems – at least from our point of view – best suited for applying a polyhedral approach, in order to solve practical Steiner tree packing problems.

4.2. The Steiner tree packing polyhedron

For the remainder of this paper we will restrict our attention to $\mathrm{STP}\,(G,\mathcal{N},c)$ and give a rough overview on inequalities that are valid for this polytope.

Desireable are inequalities that induce maximal faces of the polytope, i.e., facet-defining inequalities. To decide, whether an inequality defines a facet, the dimension of the polytope must be known. Unfortunately, the problem to decide whether the Steiner tree packing polyhedron is empty or not is already $\mathcal{NP}$-complete (see, for instance, [9, 10, 16]). Hence, there is little hope to study Steiner tree packing polyhedra for general instances $(G,\mathcal{N},c)$. Figure 10 shows some examples and the corresponding dimensions. The affine hull of the polytope of Figure 10 (b) is given by $x^1_{34} = 0,\ x^2_{34} = 1$; that of the polytope of Figure 10 (d) by $x^1_{12} = 1,\ x^2_{12} = 0,\ x^1_{23} = 0,\ x^2_{23} = 1$, for instance. The dimension jumps appear rather erratic.



Figures (a) to (d) show some examples and the dimension of the corresponding polyhedron. The two terminal sets are drawn as rectangles or cycles respectively (T_1={1,2}, T_2={3,4} or T_2={2,3} resp.) and STP abbreviates STP(G,$\mathcal{N}$,$\mathbb{1}$). The polyhedron in (a) is fulldimensional. Deleting edge {1,2} (Figure (b)) decreases the dimension by 4. If additionally edge {3,4} (Figure (c)) is deleted, there even does not exist any feasible solution. Figure (d) shows an example in which the underlying graph is complete but the corresponding polyhedron is not fulldimensional.

FIGURE 10.

We have decided to study the Steiner tree packing polyhedron for special problem instances for which the dimension can be determined easily and to look for facet-defining inequalities for these special instances. Clearly, such an approach is only sensible if the results can be carried over (at least partially) to practically interesting cases.

It has turned out that an instance $(G, \mathcal{N}, c)$, where the graph G is complete, the net list $\mathcal{N} = \{T_1, \ldots, T_N\}$ is disjoint (i.e. $T_i \cap T_j = \emptyset$ for all $i, j \in \{1, \ldots, N\}$, $i \neq j$) and the capacities are equal to one ($c = \mathbb{1}$), is an appropriate case. Under these assumptions, the polytope STP $(G, \mathcal{N}, \mathbb{1})$ is fulldimensional (see [4]).

To illustrate the rich variety of facet-defining inequalities, a complete description of the polytope associated with the example in Figure 10 (a) is shown in Table 1. Many of the inequalities coming up in this example can be generalized to other problem instances. These inequalities include, for instance, the Steiner partition inequalities for single nets and the so-called alternating cycle inequal-

ities which involve two nets. The idea of the Steiner partition inequalities is the following: Let a net $T \in \mathcal{N}$ be given. We partition the node set of the graph into p subsets $V_1, \ldots, V_p$, $p \geq 2$, such that $V_i \cap T \neq \emptyset$ for all $i = 1, \ldots, p$. Obviously, each Steiner tree for T must contain at least $p - 1$ edges whose endnodes are in different elements of the partition. This is expressed in the Steiner partition inequality. In Table 1 the inequalities (11), (12), (15), (16), (18), (22), (35) and (36) are Steiner partition inequalities. For the alternating cycle inequality we are given the following situation. Let $T_1, T_2 \in \mathcal{N}$ be two different nets with $|T_1| = |T_2| = k$. Moreover, we are given a cycle C where the terminals of the two nets appear in an alternating sequence on that cycle (see Figure 11). One can convince oneself that any Steiner tree packing S_1, S_2 such that S_1 and S_2 are edge-disjoint must use at least $k-1$ edges that are not contained in the cycle. In fact, this requirement can be strenghened and leads to the alternating cycle inequalities. In Table 1 the inequalities (13) and (14) are alternating cycle inequalities. Within the scope of this paper we refrain from explaining the details, but refer the interested reader to [4, 6].

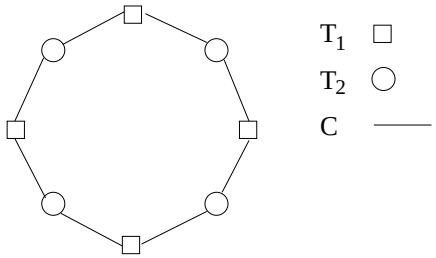


FIGURE 11.

Moreover, we developed exact algorithms and/or heuristics for solving the separation problems for several classes of inequalities. The procedures were integrated into a branch and cut framework and succesfully applied to solve switchbox routing problems discussed in the literature (see [5, 7]).

So far we just focused on the two (extreme) cases where $c = 1\!\!1$ or $c = \infty$. In order to give an impression what may happen if the capacities of the edges are arbitrary integer numbers, consider the example depicted in Figure 12.

The instance is given by a complete graph on four nodes and consists of three nets $T_1 = \{1, 4\}$, $T_2 = \{2, 4\}$, $T_3 = \{2, 3\}$. In case $c_{12} = c_{13} = c_{14} = c_{23} = c_{24} = c_{34} = 1$, a complete discription of the polytope is given by the trivial, the Steiner partition and the so-called critical cut inequalities (cf. [4]). Besides the trivial inequalities $x^k_{uv} \geq 0$, $u, v = 1, \ldots, 4$, $u \neq v$, $k = 1, 2, 3$, the right hand sides of the inequalities are always equal to one and the coefficients in the inequalities are either zero or one. Finally, this polytope is the intersection of 25 half spaces.

If we now raise the capacity of the edge connecting nodes 2 and 4 from one to

(1)	x^1_{12}	$+x^1_{13}$			$+x^1_{24}$		$+x^2_{13}$			$+x^2_{24}$	$+x^2_{34}$	$\geq$	2
(2)	x^1_{12}		$+x^1_{14}$	$+x^1_{23}$				$+x^2_{14}$	$+x^2_{23}$		$+x^2_{34}$	$\geq$	2
(3)	$2x^1_{12}$	$+x^1_{13}$			$+x^1_{24}$		$+2x^2_{13}$				$+x^2_{34}$	$\geq$	2
(4)	$2x^1_{12}$	$+x^1_{13}$			$+x^1_{24}$					$+2x^2_{24}$	$+x^2_{34}$	$\geq$	2
(5)	$2x^1_{12}$		$+x^1_{14}$	$+x^1_{23}$				$+2x^2_{14}$			$+x^2_{34}$	$\geq$	2
(6)	$2x^1_{12}$		$+x^1_{14}$	$+x^1_{23}$					$+2x^2_{23}$		$+x^2_{34}$	$\geq$	2
(7)	x^1_{12}	$+2x^1_{13}$					$+x^2_{13}$			$+x^2_{24}$	$+2x^2_{34}$	$\geq$	2
(8)	x^1_{12}		$+2x^1_{14}$					$+x^2_{14}$	$+x^2_{23}$		$+2x^2_{34}$	$\geq$	2
(9)	x^1_{12}			$+2x^1_{23}$				$+x^2_{14}$	$+x^2_{23}$		$+2x^2_{34}$	$\geq$	2
(10)	x^1_{12}				$+2x^1_{24}$		$+x^2_{13}$			$+x^2_{24}$	$+2x^2_{34}$	$\geq$	2
(11)	x^1_{12}	$+x^1_{13}$	$+x^1_{14}$									$\geq$	1
(12)	x^1_{12}			$+x^1_{23}$	$+x^1_{24}$							$\geq$	1
(13)	x^1_{12}					$+x^1_{34}$					$+x^2_{34}$	$\geq$	1
(14)	x^1_{12}						$+x^2_{12}$				$+x^2_{34}$	$\geq$	1
(15)							x^2_{13}		$+x^2_{23}$		$+x^2_{34}$	$\geq$	1
(16)								x^2_{14}		$+x^2_{24}$	$+x^2_{34}$	$\geq$	1
(17)	x^1_{12}	$+x^1_{13}$		$+x^1_{23}$							$+x^2_{34}$	$\geq$	1
(18)	x^1_{12}	$+x^1_{13}$			$+x^1_{24}$	$+x^1_{34}$						$\geq$	1
(19)	x^1_{12}	$+x^1_{13}$			$+x^1_{24}$		$+x^2_{12}$					$\geq$	1
(20)	x^1_{12}	$+x^1_{13}$			$+x^1_{24}$		$+x^2_{13}$					$\geq$	1
(21)	x^1_{12}	$+x^1_{13}$			$+x^1_{24}$					$+x^2_{24}$		$\geq$	1
(22)	x^1_{12}		$+x^1_{14}$	$+x^1_{23}$		$+x^1_{34}$						$\geq$	1
(23)	x^1_{12}		$+x^1_{14}$	$+x^1_{23}$			$+x^2_{12}$					$\geq$	1
(24)	x^1_{12}		$+x^1_{14}$	$+x^1_{23}$				$+x^2_{14}$				$\geq$	1
(25)	x^1_{12}		$+x^1_{14}$	$+x^1_{23}$					$+x^2_{23}$			$\geq$	1
(26)	x^1_{12}		$+x^1_{14}$		$+x^1_{24}$						$+x^2_{34}$	$\geq$	1
(27)	x^1_{12}						$+x^2_{13}$	$+x^2_{14}$			$+x^2_{34}$	$\geq$	1
(28)	x^1_{12}								$+x^2_{23}$	$+x^2_{24}$	$+x^2_{34}$	$\geq$	1
(29)		x^1_{13}					$+x^2_{13}$			$+x^2_{24}$	$+x^2_{34}$	$\geq$	1
(30)			x^1_{14}					$+x^2_{14}$	$+x^2_{23}$		$+x^2_{34}$	$\geq$	1
(31)				x^1_{23}				$+x^2_{14}$	$+x^2_{23}$		$+x^2_{34}$	$\geq$	1
(32)					x^1_{24}		$+x^2_{13}$			$+x^2_{24}$	$+x^2_{34}$	$\geq$	1
(33)						x^1_{34}	$+x^2_{13}$			$+x^2_{24}$	$+x^2_{34}$	$\geq$	1
(34)						x^1_{34}		$+x^2_{14}$	$+x^2_{23}$		$+x^2_{34}$	$\geq$	1
(35)							x^2_{12} $+x^2_{13}$			$+x^2_{24}$	$+x^2_{34}$	$\geq$	1
(36)							x^2_{12}	$+x^2_{14}$	$+x^2_{23}$		$+x^2_{34}$	$\geq$	1
(37)		x^1_{13}										$\geq$	0
(38)			x^1_{14}									$\geq$	0
(39)				x^1_{23}								$\geq$	0
(40)					x^1_{24}							$\geq$	0
(41)						x^1_{34}						$\geq$	0
(42)							x^2_{12}					$\geq$	0
(43)							x^2_{13}					$\geq$	0
(44)								x^2_{14}				$\geq$	0
(45)									x^2_{23}			$\geq$	0
(46)										x^2_{24}		$\geq$	0
(47)						x^1_{34}					$+x^2_{34}$	$\leq$	1
(48)					x^1_{24}					$+x^2_{24}$		$\leq$	1
(49)				x^1_{23}					$+x^2_{23}$			$\leq$	1
(50)			x^1_{14}					$+x^2_{14}$				$\leq$	1
(51)		x^1_{13}					$+x^2_{13}$					$\leq$	1
(52)	x^1_{12}						$+x^2_{12}$					$\leq$	1

TABLE 1. A complete inequality description of the example in Figure 10 (a)

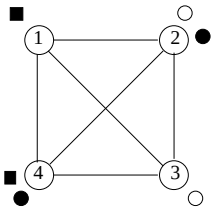


FIGURE 12.

two, the number of facet-defining inequalities increases from 25 to 548. More drastically, in some of the facet-defining inequalities, whose coefficients are in standard coprime form, the numbers 2, 3, 4, 5 or 6 appear and the right hand sides are no longer restricted to be zero or one, but lie in the range between zero and eleven. For instance, one such facet-defining inequality is the following:

$$6x^1_{14} + 2x^1_{23} + 4x^1_{24} + 3x^2_{12} + 3x^2_{14} + 5x^2_{23} + x^2_{24} + 2x^3_{12} + 6x^3_{14} + 2x^3_{24} \geq 11.$$

This small example shows that we are still far from understanding the facial structure of arbitrary Steiner tree packing polyhedra. Indeed, a series of carefull investigations of such polyhedra is indespensable in order to apply a polyhedral approach to VLSI routing problems that are not characterized by all one-capacities. One such challenging example is and remains the global routing problem where, for practically relevant examples, up to several thousands of nets must be wired in a graph with arbitrary capacities.

REFERENCES

1. M. L. BRADY, D. J. BROWN (1984). VLSI routing: Four layers suffice, in: F. P. PREPARATA (ed.): *Advances in Computing Research* Vol. 2: VLSI theory, Jai Press, London, 245 – 258.
2. M. BURSTEIN, R. PELAVIN (1983). *Hierarchical wire routing*, IEEE Transactions on Computer-Aided-Design CAD-2, 223 – 234.
3. M.R. GAREY, D.S. JOHNSON (1977). The rectilinear Steiner tree problem is $\mathcal{NP}$-complete, *SIAM J. Appl. Math.* **32** 826 – 834.
4. M. GRÖTSCHEL, A. MARTIN, R. WEISMANTEL (1992). *Packing Steiner trees: polyhedral investigations*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 92-8.
5. M. GRÖTSCHEL, A. MARTIN, R. WEISMANTEL (1992). *Packing Steiner trees: a cutting plane algorithm and computational results*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 92-9.
6. M. GRÖTSCHEL, A. MARTIN, R. WEISMANTEL (1993). *Packing Steiner trees: further facets*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 93-1.

7. M. GRÖTSCHEL, A. MARTIN, R. WEISMANTEL (1993). *Packing Steiner trees: separation algorithms*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Preprint SC 93-2.
8. R.M. KARP (1972). Reducibility among combinatorial problems, in: R.E. MILLER, J.W. THATCHER (eds.), *Complexity of Computer Computations*, Plenum Press, New York, 85 – 103.
9. M.R. KRAMER, J. VAN LEEUWEN (1984). The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits, in: F.P. PREPARATA (ed.), *Advances in Computing Research*, Vol. 2: VLSI theory, Jai Press, London, 129 – 146.
10. B. KORTE, H.J. PRÖMEL, A. STEGER (1990). Steiner trees in VLSI-Layout, in: B. KORTE, L. LOVÁSZ, H.J. PRÖMEL, A. SCHRIJVER (eds.), *Paths, Flows, and VLSI-Layout*, Springer-Verlag, Berlin Heidelberg, 185 – 214.
11. T. LENGAUER (1990). *Combinatorial algorithms for integrated circuit layout*, Wiley, Chichester.
12. W. LIPSKI (1984). On the structure of three-layer wireable layouts, F. P. PREPARATA (ed.): *Advances in Computing Research*, Vol. 2: VLSI theory, Jai Press, London, 231 – 244.
13. T. LENGAUER, M. LÜGERING (1991). *Integer program formulations of global routing and placement problems*, Reihe Informatik Nr. 95, Universität- Gesamthochschule-Paderborn, Paderborn.
14. A. MARTIN (1992). *Packen von Steinerbäumen: Polyedrische Studien und Anwendung*, Ph.D. Thesis, Technische Universität Berlin
15. A. P.-C. NG, P. RAGHAVAN, C. D. THOMPSON (1987). Experimental results for a linear program global router, *Computers and Artificial Intelligence* **6** 229 – 242.
16. M. SARRAFZADEH (1987). Channel-routing problem in the knock-knee mode is $\mathcal{NP}$-complete, *IEEE Transactions on Computer-Aided-Design CAD-6*, 503 – 506.