Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

NAM-DŨNG HOÀNG[1] THORSTEN KOCH

Steiner Tree Packing Revisited

[1] Faculty of Mathematics, Mechanics, and Informatics, Vietnam National University, 334 Nguyen Trai Str., Hanoi, Vietnam. Work supported by NAFOSTED.

Herausgegeben vom

Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125
e-mail: bibliothek@zib.de
URL: `http://www.zib.de`

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

Steiner Tree Packing Revisited

Nam-Dũng Hoàng* Thorsten Koch

January 27, 2012

Abstract

The Steiner tree packing problem (STPP) in graphs is a long studied problem in combinatorial optimization. In contrast to many other problems, where there have been tremendous advances in practical problem solving, STPP remains very difficult. Most heuristics schemes are ineffective and even finding feasible solutions is already NP-hard. What makes this problem special is that in order to reach the overall optimal solution non-optimal solutions to the underlying NP-hard Steiner tree problems must be used. Any non-global approach to the STPP is likely to fail. Integer programming is currently the best approach for computing optimal solutions. In this paper we review some "classical" STPP instances which model the underlying real world application only in a reduced form. Through improved modelling, including some new cutting planes, and by emplyoing recent advances in solver technology we are for the first time able to solve those instances in the original 3D grid graphs to optimimality.

1 Introduction

The weighted Steiner tree problem in graphs (STP) can be stated as follows:

> *Given a weighted graph $G = (V, E, c)$ and a non-empty set of vertices $T \subseteq V$ called* terminals*, find an edge set S^* such that $(V(S^*), S^*)$ is a tree of minimal weight that spans T.*

An extensive survey on the state-of-the-art of modeling and solving the STP can be found in [25]. Many papers on the STP claim real-world applications, especially in VLSI-design and wire-routing. This usually refers to a generalization of the STP, the weighted Steiner tree packing problem in graphs (STPP). Instead of having one set of terminals, we have N non-empty disjoint sets $T_1, \ldots, T_N$, called *Nets*, that have to be "packed" into the graph simultaneously, i. e., the resulting edge sets $S_1, \ldots, S_N$ have to be disjoint. In these applications, G is usually some sort of 3D grid graph. [13, 21, 14] give detailed explanations of the modeling requirements in VLSI-design. Three routing models for 2D or 3D grid graphs are of particular interest:

*The work of the first author is supported by NAFOSTED.

Channel routing: Here a complete rectangular grid graph is used. The terminals of the nets are exclusively located on two opposing borders. The size of the routing area is not fixed in advance. All nets have only two terminals, i. e., $|T_i| = 2$.

Switchbox routing: We are given a complete rectangular grid graph. The terminals may be located on all four sides of the graph. Thus, the size of the routing area is fixed.

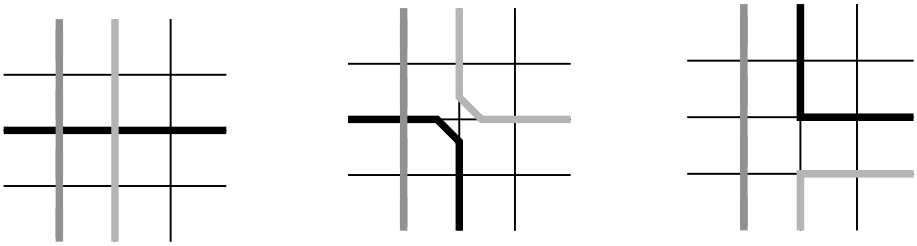
General routing: In this case the grid graph may contain holes or have a non rectangular shape. The size of the routing area is fixed and the terminals may be located arbitrarily.

The intersection of the nets is an important issue in Steiner tree packing. Again three different models are possible:

Manhattan (Fig 1(a)) Consider 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* are not allowed. This restriction guarantees that the resulting routing can be laid out on two layers at the possible expense of causing long detours.

Knock-knee (Fig 1(b)) 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 is smaller than in the Manhattan model. The main drawback is that the assignment to layers is neglected.

Node disjoint (Fig 1(c)) The nets have to be routed in a node disjoint fashion. Since no crossing of nets is possible in a planar grid graph, this requires a multi-layer model, i. e., a 3D grid graph.



(a) Manhattan model (b) Knock-knee model (c) Node disjoint model

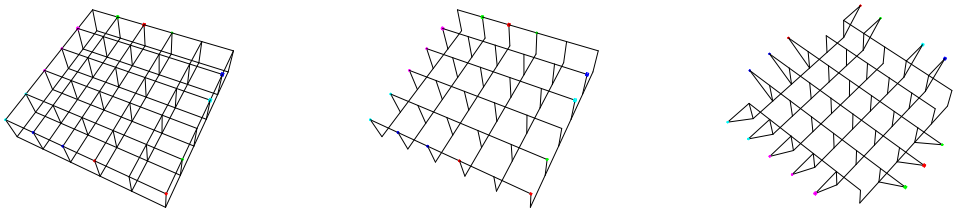
Figure 1: STPP intersection models

While channel routing usually involves only a single layer, switchbox and general routing problems are typically multi-layer problems. Using the Manhattan and Knock-knee intersection is a way to reduce the problems to a single layer. Accordingly, the multi-layer models typically use node disjoint intersection. While the multi-layer model is well suited to reflect reality, the resulting graphs become quite large. We consider two[1] possibilities to model multiple layers:

1 A third possibility is to use a single-layer model with edge capacities greater than one.

k-crossed layers (Fig 2(a)) There is given a k-dimensional grid graph (i. e., k copies of a grid graph are stacked on top of each other and corresponding nodes are connected by perpendicular lines, so-called *vias*), where k denotes the number of layers. This is called the k-layer model in [21].

k-aligned layers (Fig 2(b)) This model is similar to the crossed-layer model, but in each layer there are only connections in one direction, either east-to-west or north-to-south. [21] calls this the *directional* multi-layer model. [20] indicate that for $k = 2$ this model resembles the technology used in VLSI-wiring best. [4] mentions that current technology can use a much higher number of layers (20 and more).



(a) Multi-crossed layers (b) Multi-aligned layers (c) With connectors

Figure 2: STPP modeling taxonomy

Note that for switchbox routing there is a one-to-one mapping between feasible solutions for the Manhattan one-layer model (MOL) and the node disjoint two-aligned-layer model (TAL), assuming that there are never two terminals on top of each other, i. e., connected by a via.

For the general routing model, this mapping might not be possible. If a terminal is within the grid there is no easy way to decide the correct layer for the terminal in the two-layer model.

Unfortunately, in the seven "classic" instances given by [6, 23, 8] two terminals are connected to a single corner in several cases. This stems from the use of *connectors*, i. e., the terminal is outside the grid and connected to it by a dedicated edge. In the multi-layer models there has to be an edge from the terminal to all permissible layers (Fig 2(c)).

The Knock-knee one-layer model can also be seen as an attempt to approximate the node disjoint two-crossed-layer model. But mapping between these two models is not as easy. [5] have designed an algorithm that guarantees that any solution in the Knock-knee one-layer model can be routed in a node disjoint four-crossed-layer model, but deciding whether three layers are enough has been shown to be $\mathcal{NP}$-complete by [22].

2 Integer programming models

A survey of different integer programming models for Steiner tree packing can be found in [7]. We will examine two of the models in more detail.

2.1 Undirected partitioning formulation

This formulation is used in [13]. Given a weighted grid graph $G = (V, E, c)$, and non-empty terminal sets $T_1, \ldots, T_N$, $N > 0$, $|T_i| > 0$, $\mathcal{N} = \{1, \ldots, N\}$, we introduce binary variables x_{ij}^n for all $n \in \mathcal{N}$ and $(i, j) \in E$, where $x_{ij}^n = 1$ if and only if edge $(i, j) \in S_n$. We define $\delta(W) = \{(i, j) \in E | (i \in W, j \notin W) \vee (i \notin W, j \in W)\}$ with $W \subseteq V$. The following formulation models all routing choices for the Knock-knee one-layer model:

$$\min \quad \sum_{n \in \mathcal{N}} \sum_{(i,j) \in E} c_{ij} x_{ij}^n$$

$$\sum_{(i,j) \in \delta(W)} x_{ij}^n \geq 1 \qquad \text{for all } W \subset V, W \cap T_n \neq \emptyset, (V \setminus W) \cap T_n \neq \emptyset, n \in \mathcal{N} \quad (1)$$

$$\sum_{n \in \mathcal{N}} x_{ij}^n \leq 1 \qquad \text{for all } (i, j) \in E \quad (2)$$

$$x_{ij}^n \in \{0, 1\} \quad \text{for all } n \in \mathcal{N}, (i, j) \in E \quad (3)$$

In order to use Manhattan intersection another constraint is needed to prohibit Knock-knees. Let (i, j), (j, k) be two consecutive horizontal (or vertical) edges. Then,

$$\sum_{n \in N_1} x_{ij}^n + \sum_{m \in N_2} x_{jk}^m \leq 1$$
$$\text{for all } j \in V, N_1 \subset \mathcal{N}, N_2 \subset \mathcal{N}, N_1 \cap N_2 = \emptyset, N_1 \cup N_2 = \mathcal{N} \quad (4)$$

is called *Manhattan inequality*.[2] The model can be further strengthened with several valid inequalities as described in [10, 11, 13].

2.2 Multicommodity flow formulation

For our computational investigations we will use a multicommodity flow formulation. For the STP this was formulated in [27].

Given a weighted bidirectional grid digraph $G = (V, A, c)$ and sets $T_1, \ldots, T_N$, $N > 0$, $|T_i| > 0$ of terminals, we arbitrarily choose a root $r_n \in T_n$ for each $n \in \mathcal{N} := \{1, \ldots, N\}$. Let $R = \{r_n | n \in \mathcal{N}\}$ be the set of all roots and $T = \bigcup_{n \in \mathcal{N}} T_n$ be the union of all terminals. We introduce binary variables $\bar{x}_{ij}^n$ for all $n \in \mathcal{N}$ and $(i, j) \in A$, where $\bar{x}_{ij}^n = 1$ if and only if arc $(i, j) \in S_n$. Additionally, we introduce binary variables y_{ij}^t, for all $t \in T \setminus R$. For all $i \in V$, we define $\delta_i^+ := \{(i, j) \in A\}$ and $\delta_i^- := \{(j, i) \in A\}$. For all $t \in T_n$, $n \in \mathcal{N}$, we define $\sigma(t) := n$. The following formulation models all routing choices for any number of layers, crossed and aligned, with Knock-knee intersection:

2 Another way to enforce the Manhattan constraints is given in [15]. Every node v in the grid graph is split into two nodes v_h and v_v. v_h is connected to the horizontal edges and v_v to the vertical edges incident to v. An additional edge is used to connect v_h and v_v. This makes it impossible for more than one net to use both vertical and horizontal edges incident to v. Note that this is equivalent to converting a one-layer model into a two-aligned-layer model.

$$\min \sum_{n\in\mathcal{N}} \sum_{(i,j)\in A} c_{ij}^n \bar{x}_{ij}^n \tag{5}$$

$$\sum_{(i,j)\in\delta_j^-} y_{ij}^t - \sum_{(j,k)\in\delta_j^+} y_{jk}^t = \left\{ \begin{array}{cl} 1 & \text{if } j = t \\ -1 & \text{if } j = r_{\sigma(t)} \\ 0 & \text{otherwise} \end{array} \right\} \quad \text{for all } j \in V, t \in T \setminus R \tag{6}$$

$$0 \le y_{ij}^t \le \bar{x}_{ij}^{\sigma(t)} \qquad \text{for all } (i,j) \in A, t \in T \setminus R \tag{7}$$

$$\sum_{n\in\mathcal{N}} (\bar{x}_{ij}^n + \bar{x}_{ji}^n) \le 1 \qquad \text{for all } (i,j) \in A \tag{8}$$

$$\bar{x}_{ij}^n \in \{0,1\} \qquad \text{for all } n \in \mathcal{N}, (i,j) \in A \tag{9}$$

$$y_{ij}^t \in \{0,1\} \qquad \text{for all } t \in T \setminus R, (i,j) \in A \tag{10}$$

To use node disjoint intersection we have to add:

$$\sum_{n\in\mathcal{N}} \sum_{(i,j)\in\delta_j^-} \bar{x}_{ij}^n \le \left\{ \begin{array}{ll} 0 & \text{if } j \in R \\ 1 & \text{otherwise} \end{array} \right. \qquad \text{for all } j \in V \tag{11}$$

The above system (especially (7)) has the following implications which hold also for the linear relaxation, i. e., $\bar{x}_{ij}^n \in [0,1]$ instead of (9):

$$\bar{x}_{ij}^n \ge \max_{t\in T_n\setminus R} y_{ij}^t \quad \text{for all } (i,j) \in A, n \in \mathcal{N} \tag{12}$$

Assuming $c_{ij}^n > 0$, inequality (12) is even met with equality. This leads to

$$\bar{x}_{jk}^n \le \sum_{(i,j)\in\delta_j^-} \bar{x}_{ij}^n \quad \text{for all } j \in V \setminus R, (j,k) \in \delta_j^+, n \in \mathcal{N} \ , \tag{13}$$

i. e., for each net the flow on each outgoing arc is less than or equal to the total flow into the node. *Proof:* For each $t \in T \setminus R$ and each $j \in V \setminus T$ equation (6) states that $\sum_{(i,j)\in\delta_j^-} y_{ij}^t = \sum_{(j,k)\in\delta_j^+} y_{jk}^t$. It follows that $\sum_{(i,j)\in\delta_j^-} y_{ij}^t \ge y_{jk}^t$ for any $(j,k) \in \delta_j^+$ and further $\sum_{(i,j)\in\delta_j^-} \max_{t\in T_n\setminus R} y_{ij}^t \ge \max_{t\in T_n\setminus R} y_{jk}^t$ for any $(j,k) \in \delta_j^+$. Substituting (12) we arrive at (13). This holds also if $j \in T \setminus R$, because (6) only limits the flow on outgoing arcs in this case. □

Equation (13) can be strengthened by subtracting the incoming arc anti-parallel to the outgoing arc in question, giving the following valid inequality:

$$\bar{x}_{jk}^n + \bar{x}_{kj}^n \le \sum_{(i,j)\in\delta_j^-} \bar{x}_{ij}^n \quad \text{for all } j \in V \setminus R, (j,k) \in \delta_j^+, n \in \mathcal{N} \ , \tag{14}$$

Proof: If in any optimal solution $\bar{x}_{kj}^n$ is one, $\bar{x}_{jk}^n$ has to be zero due to (8). In this case (14) is trivially satisfied. In case $\bar{x}_{kj}^n$ is zero, (14) is equal to (13). □

2.3 Comparison of formulations

Theorem 1 *Any feasible solution of the* LP *relaxation of the multicommodity flow formulation of the node disjoint two-aligned-layer model satisfying inequality* (14) *defines a valid solution for the* LP *relaxation of the partitioning formulation of the Manhattan one-layer model by setting* $x^n_{ij} = \bar{x}^n_{ij} + \bar{x}^n_{ji}$ for all $(i,j) \in E$.

Proof: For any given $n \in \mathcal{N}$ and any given partition $W \subset V$, $W \cap T_n \neq \emptyset$, and $U = (V \setminus W) \cap T_n \neq \emptyset$, we can assume without loss of generality that $r_n \in R \cap U$ and that there exists a terminal $t \in (T_n \setminus R) \cap W$. Due to (6) $\{(i,j) \in A \mid y^t_{ij}\}$ form a path from r to t, i. e., any feasible solution to (6) will constitute a flow of one unit within the y^t variables from r_n to t. It follows that the sum of the y^t in the cut between U and W is at least one. Due to (12) the same holds for the $\bar{x}^n$, i. e., $\sum_{(i,j)\in A, i\in U, j\in W} \bar{x}^n_{ij} \geq 1$. Consequently (1) holds. (2) and (3) hold because of (8) and (9).

In the two-aligned-layer model, each node j in the graph has at most three neighbors i, k, and l, with l being the node on the other layer.

Due to (2) for (4) to hold it suffices to show that $x^n_{ij} + x^m_{jk} \leq 1$ holds for any $j \in V$ and $m \neq n$. For the two-aligned-layer model we can rewrite this as:

$$x^n_{ij} + x^m_{jk} = \bar{x}^n_{ij} + \bar{x}^n_{ji} + \bar{x}^m_{jk} + \bar{x}^m_{kj} \leq 1 \tag{15}$$

(i) For $j \in V \setminus T_n$ this holds because adding up

$$\begin{aligned} \bar{x}^n_{ij} + \bar{x}^n_{kj} + \bar{x}^n_{lj} + \bar{x}^m_{ij} + \bar{x}^m_{kj} + \bar{x}^m_{lj} &\leq 1 \quad \text{holds due to (11)} \\ \bar{x}^n_{ji} - \bar{x}^n_{kj} - \bar{x}^n_{lj} &\leq 0 \quad \text{holds due to (14)} \\ \bar{x}^m_{jk} - \bar{x}^m_{ij} - \bar{x}^m_{lj} &\leq 0 \quad \text{holds due to (14)} \end{aligned}$$

results in (15).

(ii) In case $j \in R$, (11) ensures that $\bar{x}^n_{ij} + \bar{x}^m_{kj} = 0$ and (12) proliferates this to the corresponding y^t variables. It follows from (6) that all $y^t_{ji} = 0$ for $(j,i) \in \delta^+_j$ with $\sigma(t) \neq \sigma(j)$. Since the m and n are from two disjoint nets, at most one of $\bar{x}^n_{ji}$ and $\bar{x}^m_{jk}$ can be non-zero and (15) holds.

(iii) In case $j \in T \setminus R$, (6) requires $\sum_{(i,j)\in\delta^-_j} y^j_{ij} = 1$. Due to (11), (12) this forces $y^t_{ij} = 0$ for all $(i,j) \in \delta^-_j$ with $\sigma(t) \neq \sigma(j)$. It follows from (6) that $y^t_{ji} = 0$ for all $(j,i) \in \delta^+_j$ with $\sigma(t) \neq \sigma(j)$. Since m and n are from two disjoint nets (15) holds. □

Corollary 1 *The* LP *relaxation of the multicommodity flow formulation of the node disjoint two-aligned-layer model is strictly stronger than the* LP *relaxation of the partitioning formulation of the Manhattan one-layer model.*

Proof: Theorem 1 implies that for the STPP it is at least as strong. [25] shows that for the STP the LP relaxation of the multicommodity flow formulation is

equivalent to the directed cut formulation, which in turn is strictly stronger than the undirected partitioning formulation. It follows that this holds for the STPP, since the STP is a special case. □

3 Valid inequalities

3.1 Critical cut inequalities

Consider a graph $G = (V, E)$ with unit edge capacities and a list of nets $\mathcal{N}$. For a node set $W \subset V$ we define $S(W) := \{n \in \mathcal{N} | T_n \cap W \neq \emptyset, T_n \cap (V \setminus W) \neq \emptyset\}$. The cut induced by W is called *critical* if there is no edge disjoint path entering and leaving W possible, i. e., no net without a terminal in W can pass through W in any feasible solution [12]. Hence, if $s(W) := |\delta(W)| - |S(W)| \leq 1$, the sum of the corresponding edge variables has to be zero.

In the following, we consider the node disjoint intersection case. If a node i is a terminal of some net k, then no other net can use the edges incident to i, i.e., $\bar{x}^n_{ij} = 0,\ \forall n \in \mathcal{N}, i, j \in V : T_k \cap \{i, j\} \neq \emptyset$ for some $n \neq k \in \mathcal{N}$.

In the node disjoint case, the support of a critical cut can get even stronger, as can be seen in Figure 3. For a node set $W \subset V$, the cut induced by W is

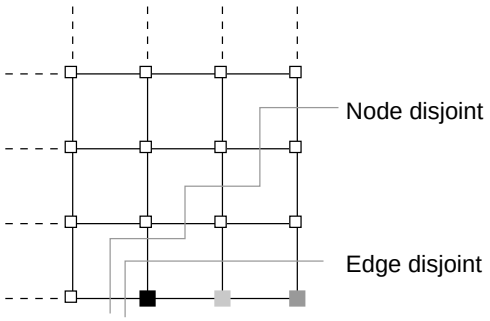


Figure 3: Critical cut in the edge disjoint and node disjoint case

called *(node-disjoint) δ^--critical* if there is no node disjoint path entering and leaving W in any solution for those nets which have no terminal in W, i.e., $\bar{x}^n_{ij} = 0,\ \forall n \in \mathcal{N}, i, j \in V : T_n \cap W = \emptyset, \{i, j\} \cap W \neq \emptyset$. The cut induced by W is called *(node-disjoint) δ^+-critical* if there is no node disjoint path leaving and entering W in any solution for those nets whose terminals belong to W, i.e., $\bar{x}^n_{ij} = 0,\ \forall n \in \mathcal{N}, i, j \in V : T_n \subset W, \{i, j\} \cap V \backslash W \neq \emptyset$. The cut induced by W is called *node-disjoint critical* if it is δ^+-critical and δ^--critical. Clearly, each critical cut is node-disjoint critical and the following holds

$$W \text{ is } \delta^+\text{-critical} \Leftrightarrow V \backslash W \text{ is } \delta^-\text{-critical.}$$

Critical cuts can easily be identified, see [24], which provides an efficient tool to eliminate variables. Finding node-disjoint δ^+-critical and δ^--critical cuts provides a stronger tool, but is not as easy. To do this efficiently, we first have

to strengthen the criterion for critical cuts. Define

$$\nu(W) := \{i \in W \,|\, \delta(W) \cap \delta(\{i\}) \neq \emptyset\}.$$

Therefore:

- If $|\nu(W)| - |S(W)| \leq 0$ or $|\delta(W)| - |S(W)| \leq 1$, then the cut induced by W is node disjoint critical.
- If $|\nu(W)| - |S(W)| \leq 1$, then the cut induced by W is δ^+-critical.

Practically, though these rules are effective for the multi-aligned layer model, they are almost useless in case of the multi-crossed layer model. In the following, we consider some other criteria to find δ^--critical cuts and to fix variables in advance.

Given a set W of nodes. Assume that there exist nets whose terminals do not lie in W. The question is whether the cut induced by W is δ^--critical. We denote $A(W) := \{ij \in A : \{i,j\} \subset W\}$, $\nu(W)^+ := \{i \in W \,|\, \exists ij \in A : j \in V \backslash W\}$, $\nu(W)^- := \{j \in W \,|\, \exists ij \in A : i \in V \backslash W\}$, $\mathcal{N}_{in} := \{n \in \mathcal{N} \,|\, T_n \subset W\}$, $\mathcal{N}_{out} := \{n \in \mathcal{N} \,|\, T_n \cap W = \emptyset\}$, and $\mathcal{N}_r := \mathcal{N} \backslash \{\mathcal{N}_{in} \cup \mathcal{N}_{out}\}$.

We are going to construct a new digraph $G_W = (V_W, A_W)$ as follows:

- G_W contains all nodes and edges of the graph $(W, A(W))$.
- For each net in $\mathcal{N}_{in}$ we construct an artificial node and add it to G_W. Denote the set of these nodes as V_W^{in}.
- For nets in $\mathcal{N}_{out}$ we construct two artificial terminals and add it to G_W. Denote the set of these two nodes as V_W^{out}.
- For each net n in $\mathcal{N}_r$ we construct an artificial terminal t_n^a and add it to G_W. Denote the set of these nodes as V_W^r.
- Connect each node in $V_W^{in} \cup V_W^{out} \cup V_W^r$ to $\nu(W)^-$ and each node in $\nu(W)^+$ to $V_W^{in} \cup V_W^{out} \cup V_W^r$ and add to G_W.

Let $\mathcal{N}_W$ be the set of all nets who have at least a terminal in W, i.e., $\mathcal{N}_W = \mathcal{N}_{in} \cup \mathcal{N}_r$. For each $i \in \mathcal{N}_W$, denote

$$T_i^W := \begin{cases} T_i & \text{if } i \in \mathcal{N}_{in} \\ \{t_i^a\} \cup T_i \cap V_W & \text{otherwise.} \end{cases}$$

We construct an artificial net $N+1$ with two terminals $T_{N+1} = V_W^{out}$. Define the weights of edges as follows

$$\bar{c}_{ij}^n = \begin{cases} 1 & \text{if } n = N+1 \text{ and } i,j \in W \\ 0 & \text{otherwise.} \end{cases}$$

Now we have a set of trees to pack on the graph G_W of $|\mathcal{N}_W| + 1$ nets with terminal sets T_i^W, $i \in \mathcal{N}_W$, and T_{N+1}. If there exists no node-disjoint solution

then the cut induced by W is δ^--critical. Otherwise, we have a criterion to verify whether a node in W does not belong to a net in $\mathcal{N}_{out}$ in each feasible solution of the STP problem. To do this, we have to find a set of trees with the largest total length. Since the length of each tree corresponding to a net in $\mathcal{N}_W$ is 0, only the length of the tree corresponding to net $N+1$ has to be taken into account. This can be formulated as an Integer Program by modifying the multicommodity flow formulation of the STPP problem:

$$\max \sum_{(i,j)\in A_W} \bar{c}_{ij}^{N+1} y_{ij}^{t_{N+1}} \tag{16}$$

$$(6)-(11) \text{ with } V = V_W, A = A_W, \mathcal{N} = \bar{\mathcal{N}}_W, T = \bar{T}_W, R = \bar{R}_W \tag{17}$$

$$\bar{x}_{r_{N+1}j}^{N+1} + \bar{x}_{jt_{N+1}}^{N+1} \leq n_j^-, \ \ \forall j \in W : (r_{N+1}, j), (j, t_{N+1}) \in A_W, n_j^- \leq 1, \tag{18}$$

where r_{N+1} and t_{N+1} are the root and another terminal of net $N+1$, $\bar{\mathcal{N}}_W := \mathcal{N}_W \cup \{N+1\}$, $\bar{R}_W$ is the set of all roots of nets in $\bar{\mathcal{N}}_W$, $T_W := \cup_{i\in\bar{\mathcal{N}}_W} T_i^W$, and $n_j^- := |\{(i,j) \in A \,|\, i \notin W\}|$. The optimal value, denoted L, is an upper bound of the length of each path entering and leaving W, which is used by some net in $\mathcal{N}_{out}$, in each feasible solution of (5). For each node $v \in W$, assume a lower bound l_v of each node disjoint path connecting two points in $\nu(W)$ and containing v is known. For example, a lower bound can be obtained by calculating shortest paths from v to every node in $\nu(W)$, and choosing l_v as the sum of the lengths of two shortest of these paths. If $L < l_v$ then no net in $\mathcal{N}_{out}$ can visit v in a solution of (5), i.e.,

$$\bar{x}_{uv}^n = \bar{x}_{vu}^n = 0, \ \forall n \in \mathcal{N}_{out}, \forall v \in W, u \in V : L < l_v, uv \in A.$$

Note that this problem is easier to solve than the original STPP problem since the sizes of the graph and the number of terminals are smaller than for the original problem.

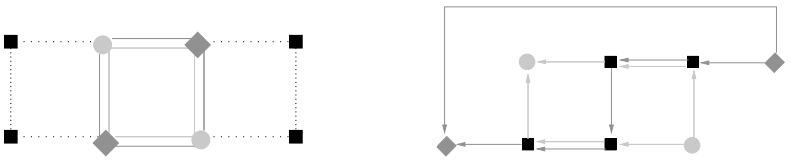
3.2 Grid inequalities

The flow formulation does not ensure

$$\sum_{(i,j)\in\delta_j^-} \bar{x}_{ij}^n \ \leq \sum_{(j,k)\in\delta_j^+} \bar{x}_{jk}^n \quad \text{for all } j \in V \setminus (T \setminus R), n \in \mathcal{N} \tag{19}$$

as shown in [19, 25].

The grid inequality described in [13] basically states that it is not possible for two nets T_1 and T_2 to cross each other in a $2 \times h$, $h \geq 2$ grid graph. As shown in Figure 4(a), there exists a non integral solution for the LP relaxation of the partitioning model in this case (unit edge weights, light gray edges mean $x_{ij}^1 = 0.5$, medium gray edges indicate $x_{ij}^2 = 0.5$). For an integral solution some path outside the $2 \times h$ grid is required. In the multicommodity flow formulation of the directed node disjoint model it is still possible to find a non-integral solution to the LP relaxation, even though the path outside the $2 \times h$



(a) undirected Knock-knee (b) directed node disjoint

Figure 4: Grid inequalities

grid is already present. Figure 4(b) shows the smallest example of this solution (unit arc weights, light gray arcs correspond to $y_{ij}^1 = 0.5$, medium gray arcs indicate $y_{ij}^2 = 0.5$). Note that at least a $3 \times h$ grid plus the two terminals of T_2 are needed for this configuration to occur.

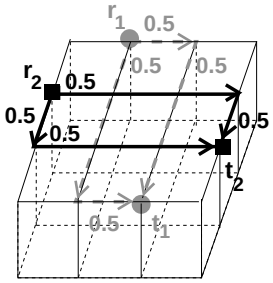


Figure 5: LP relaxation solution violates (20)

For the node disjoint crossed-layer model there is a class of simple but very effective cuts. Consider a STPP problem of two nets, each has two terminals as in Figure 5. The flows shown in the picture correspond to an optimal solution of the LP relaxation. However, it can be seen that if none of the two flows (r_1, t_1) and (r_2, t_2) leaves the upper layer, they have to cross each other, i.e., the node disjoint intersection condition is violated.

Given a STPP problem with a set of roots R and a set of terminals T. The pairs $(s_1, t_1), (s_2, t_2) \in T \times (T \backslash R)$ are called crossed if these four terminals lie on the boundary and in the same layer, $\sigma(s_1) = \sigma(t_1)$, $\sigma(s_2) = \sigma(t_2)$, $\sigma(s_1) \neq \sigma(s_2)$, and moreover the line segments (s_1, t_1) and (s_2, t_2) cross each other. Let $\mathcal{C}$ be the set of all crossing pairs and for each node v we denote v_z as the layer number node v belongs to. Then the following inequality is valid for (5):

$$\sum_{\substack{ij \in A \\ i_z = (r_1)_z, j_z \neq i_z}} y_{ij}^{t_1} + y_{ij}^{t_2} \geq 1, \quad \forall \big((r_1, t_1), (r_2, t_2) \big) \in \mathcal{C}(R), \tag{20}$$

where $\mathcal{C}(R) := \big\{ \big((s_1, t_1), (s_2, t_2) \big) \in \mathcal{C} \,|\, s_1, s_2 \in R \big\}$. Equation (20) means that at least one of the two flows (r_1, t_1) and (r_2, t_2) has to leave the layer containing these terminals. The triple $(r_1, t_1), (r_2, t_2), (r_3, t_3) \in R \times (T \backslash R)$ is called a crossing triple if each two of them are a crossing pair in $\mathcal{C}(R)$. Let $\mathcal{CT}$ be the

set of all crossing triples then the following inequality is valid for (5):

$$\sum_{\substack{ij\in A\\ i_z=(r_1)_z, j_z\neq i_z}} y_{ij}^{t_1}+y_{ij}^{t_2}+y_{ij}^{t_3}\geq 2,\ \ \forall\big((r_1,t_1),(r_2,t_2),(r_3,t_3)\big)\in\mathcal{CT}. \tag{21}$$

This is a direct implication from summing up the three corresponding inequalities of type (20) taking into account that the variables have to be integral.

We now consider another class of valid cuts. Again we assume that all terminals lie on the boundary. Let u and v be two terminals in one layer of a net n and the root of this net does not belong to the layer containing u and v. If there exist a root r and a terminal t of another net such that (u,v) and (r,t) cross each other, i.e., $\big((u,v),(r,t)\big)\in\mathcal{C}$, then the following inequality is valid for (5)

$$\sum_{\substack{ij\in A\\ j_z=t_z, j_z\neq i_z}} y_{ij}^{t}+x_{ij}^{n}\geq 2. \tag{22}$$

The following valid cut is similar to (22) but in this case we need two terminal pairs and three terminals of a third net. We consider an arbitrary net n with at least three terminals and four terminals r_1, t_1, r_2 and t_2 of two other nets n_1 and n_2, $\sigma(r_1)=\sigma(t_1)=n_1\neq n$, $\sigma(r_2)=\sigma(t_2)=n_2\neq n$, $n_1\neq n_2$, with $r_1,r_2\in R$. If there exist three terminals u, v and w of net n lying in the same layer such that

$$\forall\{s,t\}\subset\{u,v,w\}, s\neq t:\big((s,t),(r_1,t_1)\big)\in\mathcal{C}\text{ or }\big((s,t),(r_2,t_2)\big)\in\mathcal{C}, \tag{23}$$

then the following inequality is valid:

$$\sum_{\substack{ij\in A\\ j_z=(t_1)_z, j_z\neq i_z}} y_{ij}^{t_1}+y_{ij}^{t_2}+x_{ij}^{n}\geq 2. \tag{24}$$

Since all terminals lie on the boundary, each line segment between two terminals, which crosses an edge of a "triangle" of three terminals (including the case that they lie in a line), crosses exactly two edges of this triangle. Therefore, condition (23) just ensures that the line segments (r_1,t_1) and (r_2,t_2) cross the triangle (u,v,w) in two different pairs of edges. Inequality (24) means that the total number of vias used by the flows (r_1,t_1) and (r_2,t_2) and the net n to enter the layer containing these terminals is at least two. The proof can be done by case differentiation taking into account that the flows between (u,v,w) form a tree. See Appendix A for details. Moreover, one can prove that, if u, v and w satisfy the above condition and u,v and w do not lie in the same layer as the root of net n then the following inequality is valid:

$$\sum_{\substack{ij\in A\\ j_z=(t_1)_z, j_z\neq i_z}} y_{ij}^{t_1}+y_{ij}^{t_2}+x_{ij}^{n}\geq 3. \tag{25}$$

This proof can also be found in Appendix A. The above cuts are special cases of the following inequalities:

Given an arbitrary net n and $k-1$ terminal pairs $(r_i,\, t_i)$, $i = 1, \ldots, k-1$, of $k-1$ pairwise different nets, which are also different from n. Let r_i be the roots of these nets.

If there exist k terminals of a net n lying in a same layer such that each edge of the convex k-polygon spanned by these k terminals is crossed by at least one line segment $(r_j,\, t_j)$ for some j and for each i the line segment $(r_i,\, t_i)$ is crossed by at least an edge of the polygon, then the following inequality is valid for (5):

$$\sum_{\substack{ij \in A \\ j_z = (t_1)_z, j_z \neq i_z}} \left(\sum_{l=1}^{k-1} y_{ij}^{t_l} \right) + x_{ij}^{n} \geq k - 1, \tag{26}$$

and if these k of n terminals do not lie in the layer containing the root of net n then the following also holds

$$\sum_{\substack{ij \in A \\ j_z = (t_1)_z, j_z \neq i_z}} \left(\sum_{l=1}^{k-1} y_{ij}^{t_l} \right) + x_{ij}^{n} \geq k. \tag{27}$$

The proof follows via induction.

The next type of valid inequality does not require any of the terminals to be the root of the net they belong to. For arbitrary three terminals u_1, v_1 and w_1 of a net n_1, and arbitrary three terminals u_2, v_2 and w_2 of a net $n_2 \neq n_1$, if

$$\forall \{s_1, t_1\} \subset \{u_1, v_1, w_1\}, \exists \{s_2, t_2\} \subset \{u_2, v_2, w_2\} : \big((s_1, t_1), (s_2, t_2)\big) \in \mathcal{C}, \tag{28}$$

and

$$\forall \{s_2, t_2\} \subset \{u_2, v_2, w_2\}, \exists \{s_1, t_1\} \subset \{u_1, v_1, w_1\} : \big((s_1, t_1), (s_2, t_2)\big) \in \mathcal{C}, \tag{29}$$

i.e., the two triangles (u_1, v_1, w_1) and (u_2, v_2, w_2) cross each other, then the following inequality is valid for (5):

$$\sum_{\substack{ij \in A \\ j_z = (u_1)_z, j_z \neq i_z}} x_{ij}^{n_1} + x_{ij}^{n_2} \geq 2 + \delta_1 + \delta_2, \tag{30}$$

where δ_i is 1 if the root of net n_i does not lie in the same layer as u_i, v_i and w_i, and 0 otherwise.

4 Computational results

In this section, we present computational results obtained by generating the integer program resulting from the directed multicommodity flow formulation with Zimpl [17] and then solving it with cplex 12.3. All computations are done on a 48 GB RAM dual quad-core Intel Xeon X5672 at 3.20 GHz with TurboBoost active and Hyperthreading deactivated. Since the crossed-layer model proved to be much harder to solve, we used all eight cores, while just one

Name	Size	N	$\|T\|$	Variables	Constrains	Non-zeros
Knock-knee one-layer model						
augmenteddense-1	16×18	19	59	70,918	62,561	215,158
dense-1	15×17	19	59	63,366	56,057	192,246
difficult-1	23×15	24	66	94,776	78,292	275,712
modifieddense-1	16×17	19	59	67,260	59,410	204,060
moredifficult-1	22×15	24	65	89,440	73,299	258,688
pedagogical-1	15×16	22	56	56,560	44,909	159,580
terminalintens-1	23×16	24	77	119,196	106,403	365,328
Node disjoint two-aligned-layer model						
augmenteddense-2	16×18	19	59	97,940	91,587	326,438
difficult-2	23×15	24	66	131,604	115,399	427,536
moredifficult-2	22×15	24	65	123,890	107,779	400,952
pedabox-2	15×16	22	56	77,168	65,067	245,576
sb11-20-7	21×21	7	77	197,274	243,726	751,884
sb3-30-26d	31×31	29	87	485,212	437,515	1,607,464
sb40-56	41×41	56	112	1,111,264	755,307	3,318,000
sb60-60	61×61	60	149	3,290,218	2,651,399	10,489,672
terminalintens-2	23×16	24	77	164,010	154,947	550,104
Node disjoint two-crossed-layer model						
augmenteddense-2	16×18	19	59	153,664	127,257	522,421
difficult-2	23×15	24	66	168,198	121,349	607,212
moredifficult-2	22×15	24	65	204,880	159,488	662,128
pedabox-2	15×16	22	56	95,592	64,184	341,178
sb11-20-7	21×21	7	77	326,634	362,166	1,245,020
sb3-30-26d	31×31	29	87	805,132	651,415	2,667,643
sb40-56	41×41	56	112	1,845,984	1,125,947	5,513,032
sb60-60	61×61	60	149	5,909,158	4,400,252	28,425,308
terminalintens-2	23×16	24	77	271,348	229,526	909,016
Node disjoint three-aligned-layer model						
dense-3	15×17	19	59	144,668	131,412	482,722
modifieddense-3	16×17	19	59	154,580	140,307	515,986
taq-3	25×25	14	35	115,640	98,508	368,760
Node disjoint three-crossed-layer model						
dense-3	15×17	19	59	229,392	189,570	764,941
modifieddense-3	16×17	19	59	245,086	202,434	817,609
taq-3	25×25	14	35	178,850	137,337	570,570
Node disjoint four-aligned-layer model						
alue-4	25×25	22	55	294,084	236,417	933,830
Node disjoint four-crossed-layer model						
alue-4	25×25	22	55	443,988	326,637	1,409,982

Table 1: STPP instances

core was utilized for the other models. Still, for the crossed-layer models we will use minutes as the unit for reporting time, in contrast to seconds for the other models. As we had expected from earlier experiments the MCF-Cuts [26, 1] introduced by CPLEX 12 had no impact on solving the instances. The reason is that the models used in this paper are not capacitated. If not noted otherwise CPLEX was used in default mode with integer optimality gap tolerance set to 0.0.

Table 1 lists all Steiner tree packing problem instances we have considered. N denotes the number of nets, and $|T|$ the total number of terminals. The columns labeled *Variables*, *Constraints*, and *Non-zeros* list the number of variables, constraints, and non-zero entries in the constraint matrix of the generated integer programs before preprocessing.

Name	B&B Nodes	Time [s]	LP relaxation	Arcs
augmenteddense-1	63	1,649	466.5	469
dense-1	150	1,199	438.0	441
difficult-1	1	17	464.0	464
modifieddense-1	1	29	452.0	452
moredifficult-1	1	12	452.0	452
pedabox-1	1	7	331.0	331
terminalintens-1	1	96	535.0	536

Table 2: Results for the Knock-knee-one-layer model

4.1 Results for the Knock-knee one-layer model

Table 2 shows the results for the Knock-knee one-layer model. *B&B Nodes* denotes the number of Branch-and-Bound nodes including the root node evaluated by CPLEX. The column labled *Time* shows the consumed CPU time in secounds. LP *relaxation* lists the objective function value of the initial LP relaxation of the root node before any cuts applied by CPLEX. Finally, *arcs* is the total number of arcs used in the optimal solution which for one-layer models is equivalent to the optimal objective value.

As we can see from the table, the LP relaxation of the flow model is rather strong. This is in line with other reported results including [24, 15]. Since for *difficult-1, modifieddense-1, moredifficult-1,* and *pedabox-1* the relaxation already provides the optimal value, it is possible to solve these instances without any branching. For *terminalintens-1* the relaxation is one below the optimum, but CPLEX is able to push the lower bound up by generating Gomory rounding and 0-1/2-Chvatal-Gomory cuts. The number of B&B nodes and therefore the computing time depends very much on the branching decisions taken. During our experiments the solutions were always found in the tree and not by heuristics. By using improved settings, like switching off the heuristics and just trying to move the best bound the number of nodes needed for *augmenteddense-1* and *dense-1* can be at least halved.

4.2 Results for the node disjoint multi-aligned-layer model

Table 3 shows results for the node disjoint multi-aligned-layer model. Since this is a multi-layer model we have to assign costs to the vias. These are given in the column labeled *Via-cost*. The column labled *LP relaxation* gives the objective value of the initial LP relaxation. The next three columns list the numbers of vias, "regular" arcs, and vias+arcs in the optimal solution.

Name	B&B Nodes	Time [s]	Via-cost	LP relaxation	Vias	Arcs	Vias +Arcs
augmenteddense-2	1	32	1	504.0	35	469	504
augmenteddense-2	1	20	1000	35,469.0	35	469	504
augmenteddense-2	1	67	0.001	469.035	35	469	504
difficult-2	1	23	1	526.0	56	470	526
difficult-2	7	181	1000	50,310.2727	51	484	535
difficult-2	1	50	0.001	468.8363333	63	469	532
moredifficult-2	5	113	1	518.60	61	461	522
moredifficult-2	37	705	1000	50,276.8465	53	481	534
moredifficult-2	1	38	0.001	460.9916667	61	461	522
pedabox-2	1	10	1	390.0	47	343	390
pedabox-2	11	34	1000	45,885.1333	47	343	390
pedabox-2	14	78	0.001	341.4601533	47	343	390
terminalintens-2	1	28	1	596.0	59	537	596
terminalintens-2	1	31	1000	55,562.0	55	562	617
terminalintens-2	1	28	0.001	537.059	59	537	596
dense-3	1	30	1	471.0	35	436	471
dense-3	1	21	1000	35,436.0	35	436	471
dense-3	1	44	0.001	436.035	35	436	471
modifieddense-3	1	24	1	485.0	35	450	485
modifieddense-3	1	26	1000	35,450.0	35	450	485
modifieddense-3	1	33	0.001	450.035	35	450	485

Table 3: Results for the node disjoint multi-aligned-layer model (part 1)

In case of unit via costs, the objective value of the LP relaxation is equal to the objective value of the optimal integer solution for all instances except for *moredifficult-2*. The value of the LP relaxation for *moredifficult-2* is 518.6 (optimal 522). This is weaker than the value reported in [13][3], while for *pedabox-2* the relaxation is stronger than reported. The instances where the LP relaxation does not reach the optimum are different ones from the Knock-knee-one-layer model.

The *dense* [23] and *modifieddense* [8] problems are not solvable within the Manhattan one-layer model as reported in [13]. Therefore, no solution in the node disjoint two-aligned-layer model is possible. As can be seen in Figures 9(a) and 9(b) both problems have a three-layer solution with only one net using

3 This indicates that some of the strengthening cuts used by [13] to tighten the undirected partitioning formulation can also be used to tighten the directed flow formulation.

the third layer at a single point. To our knowledge, this is the first time these solutions have been computed.

4.2.1 Via minimization

Traditionally via minimization is viewed as a separate problem after the routing has taken place [9]. Since we work with multi-layer models via minimization is part of the routing. As can be seen in Table 3 we tried the "classical" instances with three different cost settings for the vias. First unit costs were used to minimize the total number of arcs, including vias. Next, the number of vias was minimized by setting the cost to 1,000, which is above the total cost of all "regular" arcs, ensuring that a global minimum is reached. Finally, the cost of each via was set to 0.001, effectively minimizing the number of "regular" arcs. This results in solutions that have the same number of arcs as reported in [13] for the Manhattan one-layer model.

Interestingly, the number of vias is constant for *augmenteddense-2, pedabox-2, modifieddense-3,* and *dense-3*. For the other instances, minimization of the number of vias always results in detours, i. e., higher total number of arcs used.

4.2.2 New instances

All the instances presented so far are relatively old and can be solved in less than 12 minutes with CPLEX default settings and in less than 5 minutes with adapted settings. To get an outlook on how far our approach will take us, we tried some new instances. The results can be found in Table 4.

Name	LP solver	B&B Nodes	Time [s]	LP relaxation	Vias	Arcs	Vias +Arcs
sb11-20-7	Simplex	1	6,935	593.0	107	486	593
	Barrier	1	144				
sb3-30-26d	Simplex	1	1,499	1416.0	130	1,286	1,416
	Barrier	1	417				
sb40-56	Simplex	1	276	2,452.0	166	2,286	2,452
	Barrier	1	598				
sb60-60	Simplex	1	90,692	4,698.0	185	4,513	4,698
	Barrier	1	27,418				
taq-3	Simplex	12	85	429.0	66	371	437
	Barrier	9	146				
alue-4	Simplex	10	658	784.2857	117	668	785
	Barrier	5	782				

Table 4: Results for the node disjoint multi-aligned-layer model (part 2)

sb11-20-7, sb3-30-26d, sb40-56, and *sb60-60* are all randomly generated switch-box instances. *sb40-56* is about four times the size of the "classical" instances

and the resulting IP has more than three million non-zero entries in the constraint matrix. *sb60-60*, having 10 million non-zeros, is again about 3 times larger than *sb40-56*. *sb11-20-7* is noteworthy because all nets have eleven terminals. This value is substantially higher compared to the "classical" instances where the nets have at most six terminals. For all four instances the value of the LP relaxation turned out to be equal to the value of the integer optimal solution. Pictures of the solutions can be found in Figures 13, 14, 15, and 16. The via cost was set to one in all instances.

taq-3 (Figure 17) and *alue-4* (Figure 18) are general routing problems based on circuits described in [16]. *alue-4* is the only instance so far that requires four aligned layers. In both instances the LP relaxation does not reach the value of the optimum integer solution. Since for *alue-4* the difference to the optimum is less than 1, it is possible to solve this instance without branching if the optimal solution can be found. The time needed to solve the root LP relaxation looks quite high for instances of the given size. Also the number of simplex iterations needed to reoptimize the subproblems is often several thousand which is quite above the empirical expectation. Looking at the solver logs, obviously, the solver experiences many degenerate pivots. Regarding reoptimization another reason might be that moving from one optimal solution to a new one after fixing a variable requires many variables to change there value. Variables with non-integral values imply that at least two nets are competing for a route. By fixing such a variable at least one net has to be rerouted, possibly several. This leads to a high number of variables that have to change their value.

Using the barrier or interior-point algorithm especially to solve non-root branch-and-bound nodes is quite uncommon as the algorithm is in general not restart capable, i. e., requires to solve each subproblem from scratch. Nevertheless, we observed for several instances that it was faster to solve the LP relaxations from scratch with the barrier algorithm than to reoptimize with the dual simplex algorithm.

For the instances in Table 4, we tested the effect of using the Barrier instead of the Dual Simplex algorithm to solve the root and the subproblems. As can be seen, the Barrier is much faster in solving the initial root relaxation. For *taq-3* it takes only five seconds, while the Simplex needs 30. Also the Barrier setting for some reason seems to lead to fewer B&B nodes in case branching is necessary. Nevertheless, for *taq-3* and *alue-4* it can be seen that the time saved in the root node is later lost in the subproblems, as the average time per node is still higher than for the Simplex.

4.3 Results for the node disjoint multi-crossed-layer model

Finally, we will have a look at the crossed-layer models. As we will see, several trends we observed when computing larger aligned-layer models in the previous section aggravated: CPLEX had more difficulties to find feasible solutions, and the performance of the Simplex algorithm both on the root node and the subproblems became much worse.

Name	Heur [m]	Total [m]	B&B Nodes	LP relaxation	Vias	Arcs	Vias +Arcs	Best Bound
modifieddense-3	–	88	121	476.9078	28	451	479	
dense-3	–	55	77	461.8062	30	434	464	
augmenteddense-2	108	269	227	492.6260	29	469	498	
pedabox-2	18	112	3,027	353.4275	26	336	362	
difficult-2	28	822	2,214	492.5417	39	464	503	
terminalintens-2	97	3,103	3,453	573.1981	46	538	584	
moredifficult-2	60	30,727	24,713	481.1991	38	455	493	
sb11-20-7 (B)	1,907	2,271	50	489.7351	65	470	535	494.0174
sb3-30-26d (B)	411	6,032	425	1,310.3073	105	1,286	1,391	1,322.5219
sb40-56 (B)	720	8,588	524	2,266.5803	148	2,278	2,426	2,304.4128
sb60-60 (B)	457	4,289	1	4,548.9633	185	4,513	4,698	4,577.9898
taq-3	–	3	1	330.9000	22	311	333	
alue-4	308	43,644	10,029	687.2966	71	628	699	696.4552

Table 5: Results for the node disjoint multi-crossed-layer model

It turned out to be much easier to find solutions to the aligned-layer models than to the crossed-layer models. Based on this observation, instead of starting solving the original multi-crossed layers model, we solve several models corresponding to some sparser underlying grids, e.g., the multi-aligned layers grid. After each step we obtain a feasible solution. We then added some edges to the grid and resolve the problem corresponding to the new grid using the solution from the previous step as a start value for the optimization problem. Furthermore, we fixed variables corresponding to edges in some region to the values of some feasible solution, or fixed variables to zero using the methods described in Section 3. The number of variables can be significantly reduced thereby. This way we were able to successively improve a given feasible solution as we changed the fixing region step by step.

For the instances listed in Table 5, except for *dense-3*, *modifieddense-3* and *taq-3*, the above method was used to provide CPLEX with a starting solution. For all instances solved to optimality the provided solution turned out to be already optimal. The time needed to compute these initial solution is given under *Heur*.

To solve the instances we used 8 threads in opportunistic mode, the total times needed including the heuristics is reported in column *Total* as minutes of wall clock time. The *optimization emphasis* of CPLEX was set to "optimality", cut generation was set "aggressive" for Gomory-, 0-1/2-, and cover-cuts, all other cuts were set to "moderate". Furthermore, we explicitly added cuts (20)–(30) presented in Section 3.

While it is possible to find reasonable solutions with our heuristics proving optimality is still hard task. The first part of the table is ordered by increasing difference between the LP relaxation and the optimum value. As can be seen this is reflected quite well in the number of B&B nodes needed to prove optimality. While the cuts we presented in this paper proved quite helpful, the

main reason for the long running time is that solving the node LPs is very time consuming. For example, the number of B&B nodes that CPLEX needs to solve the instance *terminalintens-2* is merely 3,453. However, it takes 51.7 hours to solve the problem and 43.3 hours only for the first 2000 nodes. For the *sb** instances the performance of the Simplex algorithm degregated to the point where is became useful to use the Barrier instead to solve the LP relaxations. This is indicated by a *(B)* after the instance name. The Barrier of course is capable of utilizing all 8 cores, but will not return a vertex solution of the LP. Therefore, a procedure known as cross-over is employed by the solver afterwards. This procedure is very similar to a Simplex algorithm and therefore only runs sequentially. In particular for *sb11-20-7* the cross-over for the root LP took 82 seconds in comparison to the 60 seconds the barrier algorithm needed to solve the LP in the first place. For *sb40-56*, the lower bound of 2,304.4128 is reached after 65 nodes. There was no change until the computation was aborted 460 nodes and nearly 4 days later, i. e., one core needs more than 1.5 hours to solve single LP relaxation. The final challenge is *sb60-60*. Here the barrier algorithm needs about 212 minutes on 8 threads to solve the initial root relaxation and then another 45 minutes to perform the crossover procedure. In the following two days of computation, CPLEX did not finish even the root processing phase. The best solution knwo is identical to the one computed for the aligned-layer model. Regarding *taq-3* it should be noted that it is solved by a combination of generating cutting planes and preprocessing. CPLEX performs two restarts until the lower bound reaches the optimum.

Figure 6 shows the primal and dual bounds during the solving process of the instance *pedabox-2* with CPLEX using default setting and CPLEX using our heuristics, valid cuts, and variables elimination. Clearly, our approach improves both primal and dual bounds. For *pedabox-2* with unit via-cost, our heuristics found an optimal solution after 17.8 minutes, while CPLEX alone could not find the optimal value after more than 80 hours. For this problem, our approach (critical and crossing cuts and providing the solution form the special heuristics) gives a dual bound of value 358.2311 directly after the root node, while default CPLEX reaches this value only after 46.1 hours and 158,507 nodes. We stop solving with default CPLEX after 83.33 hours and 300,516 nodes, and obtain a dual bound of 358.7628. This value is already reached by our approach after 49.6 minutes and 471 nodes.

As we mentioned in Section 4.2, there exists a three layers solution for *modifieddense-3* in the multi-aligned-layer model. Only two layers are needed for the multi-crossed-layer model, see Figure 11.

Figure 10–12 show the optimal solutions of *dense-3*, *modifieddense-3*, and *pedabox-2* in knock-knee-one-layer and multi-crossed-layer models. For the knock-knee-one-layer model, we use connectors as mentioned in Section 1. The numbers of regular arcs needed by the solutions of *dense-3* and *modifieddense-3* in multi-crossed-layer model are less than the ones in knock-knee-one-layer model. But for *pedabox-2* it is converse, i.e., the number of regular arcs needed by the solution of *pedabox-2* in the multi-crossed-layer model is more than the one in the knock-knee-one-layer model.

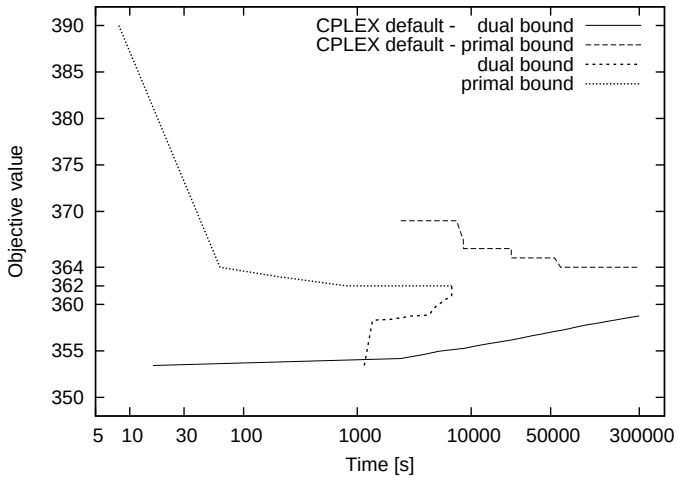


Figure 6: pedabox-2 with unit via cost – primal and dual bounds

5 Conclusion and outlook

Our results represent a significant improvement over previous methods, where it was impossible to solve, or even find good feasible solutions to, the instances in the crossed-layer model. Especially for the aligned-layer case the linear relaxation of the flow formulation proved to be very strong. In general it is possible now to solve multi-layer models that capture more features of the original real-world application.

Still, there are several areas where improvements seem possible: Very often the LP relaxation already yields the value of the optimal integer solution but is not integral. Can this be improved by either problem specific valid inequalities or general cuts? If there is a gap, can classes of violated inequalities be found to improve this? Furthermore, the number of B&B nodes needed depends very much on the branching decisions taken; Would a problem specific branching selection be helpful? Alternatively, can the use of a pivot and complement type heuristic, like those described in [2, 3], help to discover an integral solution once the optimal objective is reached?

The flow formulation itself still has some unexploited freedom. Computational experiments show that changing the root vertices of the nets influences the computational time, especially the effectiveness and number of our cuts. However, it is an open question how to choose the roots in a beneficial way.

Proving optimality could significantly speed up if it were possible to compute a good lower bound on the number of vias in each feasible solution.

Finally, the biggest bottleneck from the computational side is the difficulty of solving the LP relaxations quickly. Obviously, the instances we have used here define a class of hard to solve LP problems. Any progress in LP solving would directly translate into better solvability of the STPP. One possible way is to use parallel computers. Since the time to solve a single LP is often several min-

utes even the use of a distributed memory system seems possible. Nevertheless, one would encounters severe ramp-up and ramp-down problems with these instances, as the number of open nodes to distribute remains quite small and any further progress on the modelling side will further reduce the number of nodes that have to be computed.

References

[1] T. Achterberg and C. Raack, *The MCF-separator – detecting and exploiting multi-commodity flows in MIPs*, Mathematical Programming C, (2010), pp. 125–165.

[2] E. Balas and C. Martin, *Pivot and complement - a heuristic for 0/1 programming*, Management Science, 26 (1980), pp. 86–96.

[3] E. Balas, S. Schmietab, and C. Wallacea, *Pivot and shift - a mixed integer programming heuristic*, Discrete Optimization, 1 (2004), pp. 3–12.

[4] C. Boit, *Personal communication*, 2004.

[5] M. L. Brady and D. J. Brown, *VLSI routing: Four layers suffice*, in Advances in Computing Research: VLSI theory, F. P. Preparata, ed., vol. 2, Jai Press, London, 1984, pp. 245–258.

[6] M. Burstein and R. Pelavin, *Hierachical wire routing*, IEEE Transactions on computer-aided design, 2 (1983), pp. 223–234.

[7] S. Chopra, *Comparison of formulations and a heuristic for packing Steiner trees in a graph*, Annals of Operations Research, 50 (1994), pp. 143–171.

[8] J. P. Coohoon and P. L. Heck, *BEAVER: A computational-geometry-based tool for switchbox routing*, IEEE Transactions on computer-aided design, 7 (1988), pp. 684–697.

[9] M. Grötschel, M. Jünger, and G. Reinelt, *Via minimization with pin preassignments and layer preference*, ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik, 69 (1989), pp. 393–399.

[10] M. Grötschel, A. Martin, and R. Weismantel, *Packing Steiner trees: A cutting plane algorithm and computational results*, Mathematical Programming, 72 (1996), pp. 125–145.

[11] ——, *Packing Steiner trees: Further facets*, European Journal of Combinatorics, 17 (1996), pp. 39–52.

[12] ——, *Packing Steiner trees: Polyhedral investigations*, Mathematical Programming, 72 (1996), pp. 101–123.

[13] M. Grötschel, A. Martin, and R. Weismantel, *The Steiner tree packing problem in VLSI design*, Mathematical Programming, 78 (1997), pp. 265–281.

[14] S. Held, B. Korte, D. Rautenbach, and J. Vygen, *Combinatorial optimization in vlsi design*, in Combinatorial Optimization – Methods and Applications, V. Chvátal, ed., vol. 31 of NATO Science for Peace and Security Series - D: Information and Communication Security, 2011, pp. 33–96.

[15] D. G. Jørgensen and M. Meyling, *Application of column generation techniques in VLSI design*, Master's thesis, Department of Computer Science, University of Copenhagen, 2000.

[16] M. Jünger, A. Martin, G. Reinelt, and R. Weismantel, *Quadratic 0/1 optimization and a decomposition approach for the placement of electronic circuits*, Mathematical Programming, 63 (1994), pp. 257–279.

[17] T. Koch, *ZIMPL, http://zimpl.zib.de.*

[18] T. Koch, *Rapid Mathematical Programming*, PhD thesis, Technische Universität Berlin, 2004. ZIB-Report 04-58.

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

[20] B. Korte, H.-J. Prömel, and A. Steger, *Steiner trees in VLSI-layout*, in Paths, Flows, and VLSI-Layout, B. Korte, L. Lovász, H.-J. Prömel, and A. Schrijver, eds., Springer, 1990.

[21] T. Lengauer, *Combinatorial Algorithms for Integrated Circuit Layout*, Wiley, 1990.

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

[23] W. K. Luk, *A greedy switch-box router*, Integration, 3 (1985), pp. 129–149.

[24] A. Martin, *Packen von Steinerbäumen: Polyedrische Studien und Anwendungen*, PhD thesis, Technische Universität Berlin, 1992.

[25] T. Polzin, *Algorithms for the Steiner Problem in Networks*, PhD thesis, Universität des Saarlandes, 2003.

[26] C. Raack, A. M. C. A. Koster, S. Orlowski, and R. Wessly, *On cut-based inequalities for capacitated network design polyhedra*, Networks, 57 (2011), pp. 141–156.

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

A Proof of the valid cuts

In this section we give proofs of the validity for two classes of cuts presented in Section 3. The other cuts can be proved similarly.

Proposition 1 *For an arbitrary net n with at least three terminals and four terminals r_1, t_1, r_2 and t_2 of two other nets n_1 and n_2,*

$$\sigma(r_1) = \sigma(t_1) = n_1 \neq n, \quad \sigma(r_2) = \sigma(t_2) = n_2 \neq n, \quad n_1 \neq n_2,$$

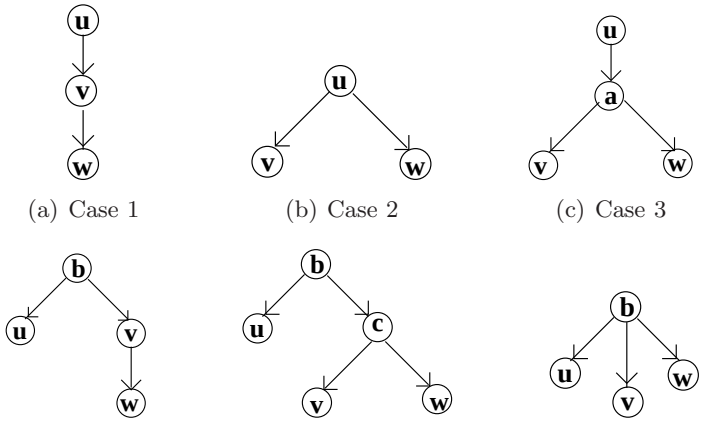
with $r_1, r_2 \in R$, if there exist three terminals u, v and w of net n lying in a same layer such that

$$\forall \{s,t\} \subset \{u,v,w\}, s \neq t : \big((s,t),(r_1,t_1)\big) \in \mathcal{C} \text{ or } \big((s,t),(r_2,t_2)\big) \in \mathcal{C}, \tag{31}$$

then the following inequality is valid

$$\sum_{\substack{ij \in A \\ j_z=(t_1)_z, j_z \neq i_z}} y_{ij}^{t_1} + y_{ij}^{t_2} + x_{ij}^{n} \geq 2. \tag{32}$$

Proof: In each feasible solution, the flows between r and the terminals u, v and w form a tree. The minimal subtree of this tree containing the terminals u, v and w has one of the forms shown in Figure 7, where each edge represents a node disjoint path. One can easily prove that (32) holds. We only consider the



(a) Case 1 (b) Case 2 (c) Case 3

(d) Case 4 (e) Case 5 (f) Case 6

Figure 7: The 6 possible cases.

case 5 as an example. Let us denote the layer that contains u, v, and w by L. We consider the following three cases:

Case 5.1 – $c \notin L$: That means each of the two flows (c,v) and (c,w) has to enter the layer L at least once, i.e., (32) holds.

Case 5.2 – $b \notin L$ and $c \in L$: That means each of the two flows (b,u) and (b,c) has to enter the layer L at least once, i.e., (32) holds.

Case 5.3 – $b \in L$ and $c \in L$: Since each of the three paths (u, v), (v, w), and (w, u) is crossed by either (r_1, t_1) or (r_2, t_2) and $b, c \in L$, at least two of the six flows (r_1, t_1), (r_2, t_2), (b, u), (b, c), (c, v), or (c, w) have to leave the layer L and enter this again, otherwise the node disjoint intersection constraint is violated. By summing up we have that (32) holds. □

Proposition 2 *For an arbitrary net n with at least three terminals and four terminals r_1, t_1, r_2 and t_2 of two other nets n_1 and n_2,*

$$\sigma(r_1) = \sigma(t_1) = n_1 \neq n, \quad \sigma(r_2) = \sigma(t_2) = n_2 \neq n, \quad n_1 \neq n_2,$$

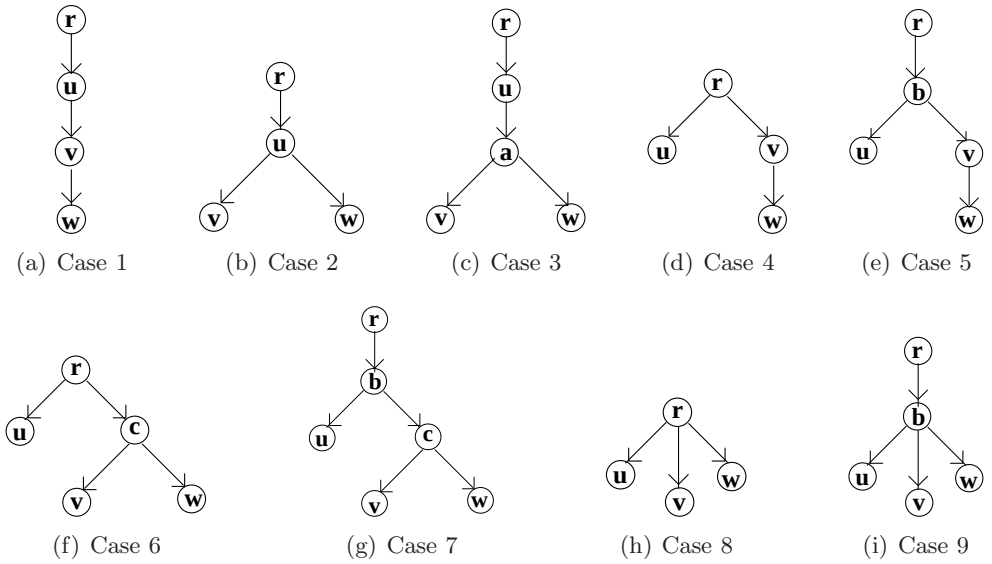
with $r_1, r_2 \in R$, if there exist three terminals u, v and w of net n lying in a same layer, which does not contain the root r of net n, such that

$$\forall \{s, t\} \subset \{u, v, w\}, s \neq t : \big((s, t), (r_1, t_1)\big) \in \mathcal{C} \text{ or } \big((s, t), (r_2, t_2)\big) \in \mathcal{C}, \tag{33}$$

then the following inequality is valid

$$\sum_{\substack{ij \in A \\ j_z = (t_1)_z, j_z \neq i_z}} y_{ij}^{t_1} + y_{ij}^{t_2} + x_{ij}^{n} \geq 3. \tag{34}$$

Proof: In each feasible solution, the flows between r and the terminals u, v and w form a tree with root r. This tree has one of the forms in Figure 8, where each edge represents a node disjoint path. We only consider the case



(a) Case 1 (b) Case 2 (c) Case 3 (d) Case 4 (e) Case 5

(f) Case 6 (g) Case 7 (h) Case 8 (i) Case 9

Figure 8: Nine cases

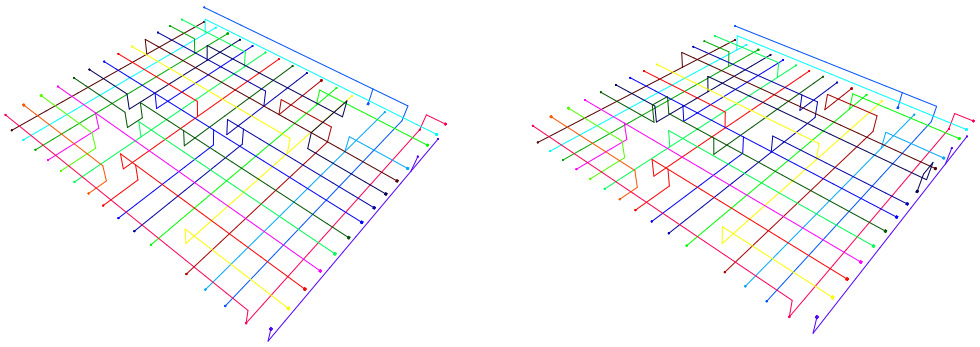
7. Other cases can be treated with similar arguments. Let us denote the layer that contains u, v, and w by L. We consider the following three cases:

Case 7.1 – $b \in L$: That means the flow from r to b has to enter the layer L at least once. The subtree with root b is exactly the case 5 in the proof of Theorem 1. Therefore, (34) holds.

Case 7.2 – $b \notin L$ and $c \in L$: That means each of the two flows (b, u) and (b, c) has to enter the layer L at least once. Since the path between v and w is crossed by either (r_1, t_1) or (r_2, t_2) and c belongs to L, at least one of the four flows (r_1, t_1), (r_2, t_2), (c, v), or (c, w) has to leave the layer L and enter this again. By summing up we have that (34) holds.

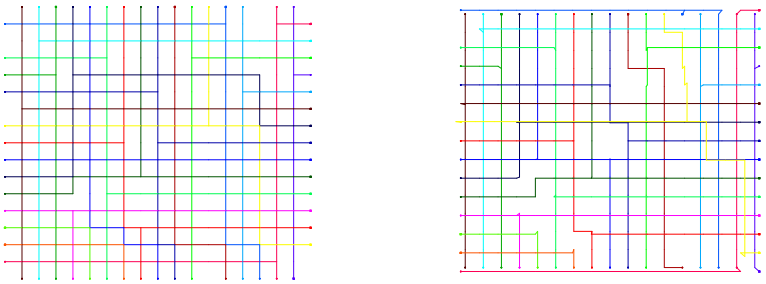
Case 7.3 – $b \notin L$ and $c \notin L$: That means each of the three flows (b, u), (c, v), and (c, w) has to enter the layer L at least once, i.e., (34) holds. □

B Solutions



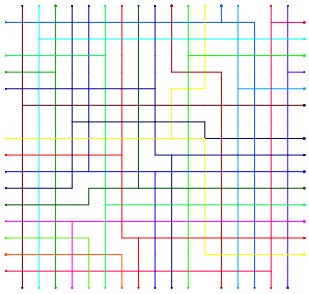
(a) dense-3 (b) modifieddense-3

Figure 9: Node disjoint three-aligned-layer solutions

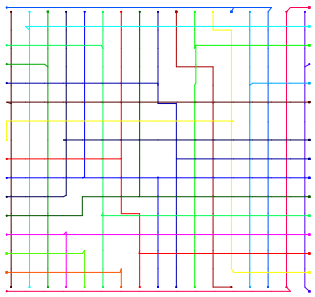


(a) Knock-knee-one-layer (b) Three-crossed-layer

Figure 10: Solutions of dense-3

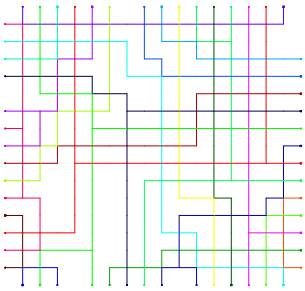


(a) Knock-knee-one-layer

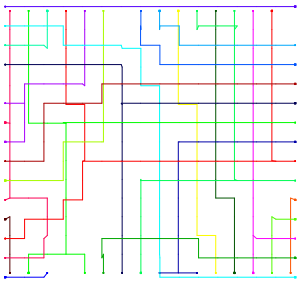


(b) Two-crossed-layer

Figure 11: Solutions of modifieddense-3

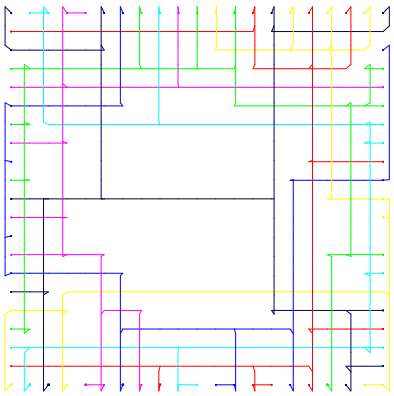


(a) Knock-knee-one-layer

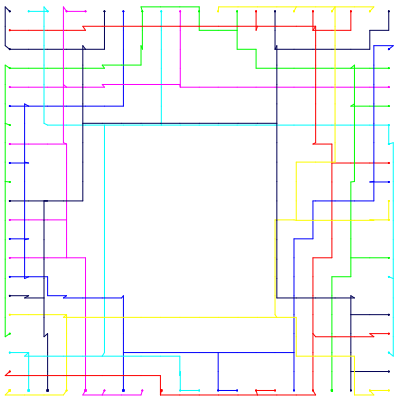


(b) Two-crossed-layer

Figure 12: Solutions of pedabox-2

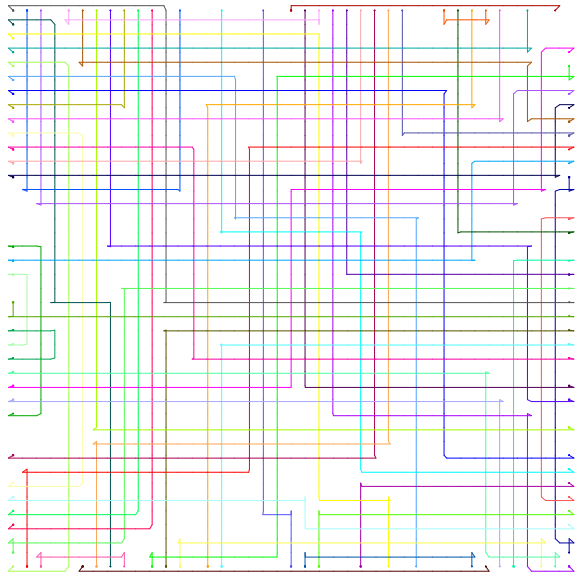


(a) Multi-aligned-layer

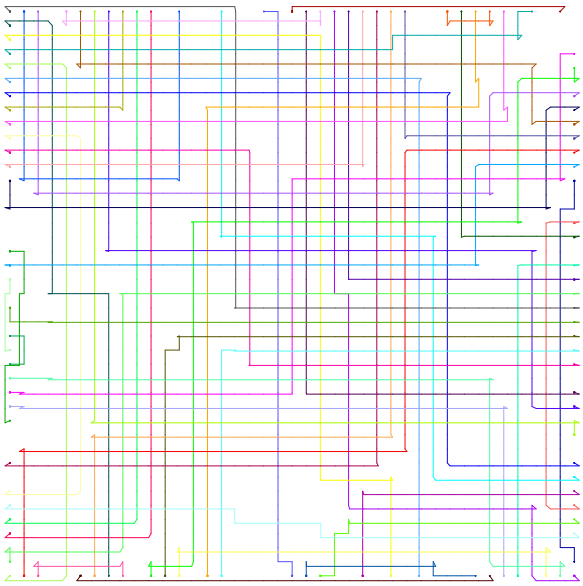


(b) Multi-crossed-layer

Figure 13: sb11-20-7



(a) Multi-aligned-layer



(b) Multi-crossed-layer

Figure 14: sb40-56

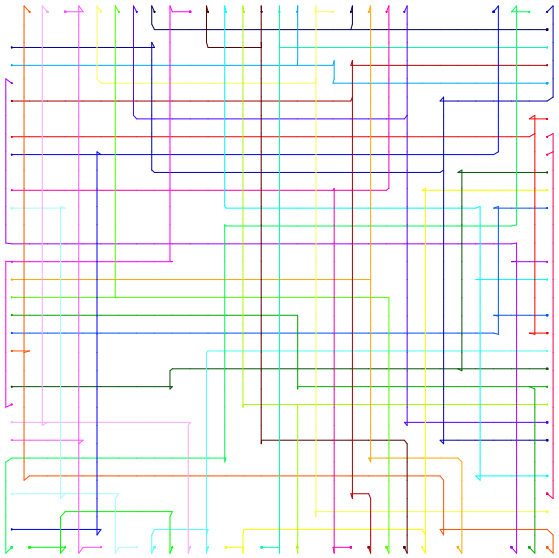


Figure 15:
sb3-30-26d
aligned-layer

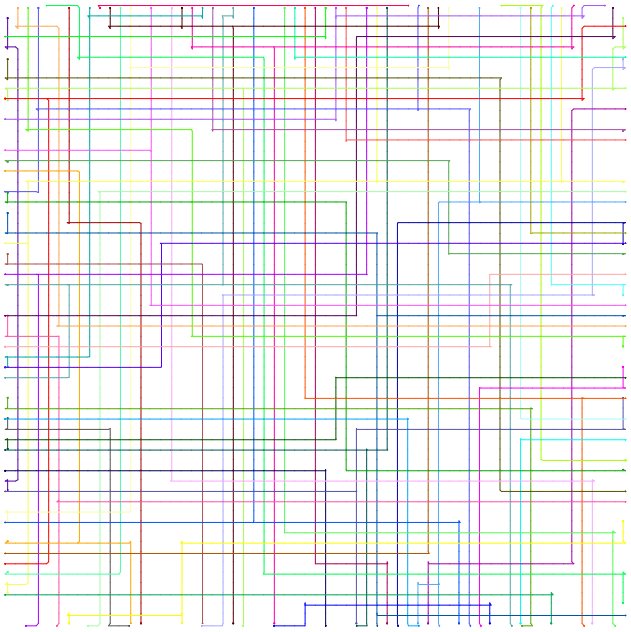
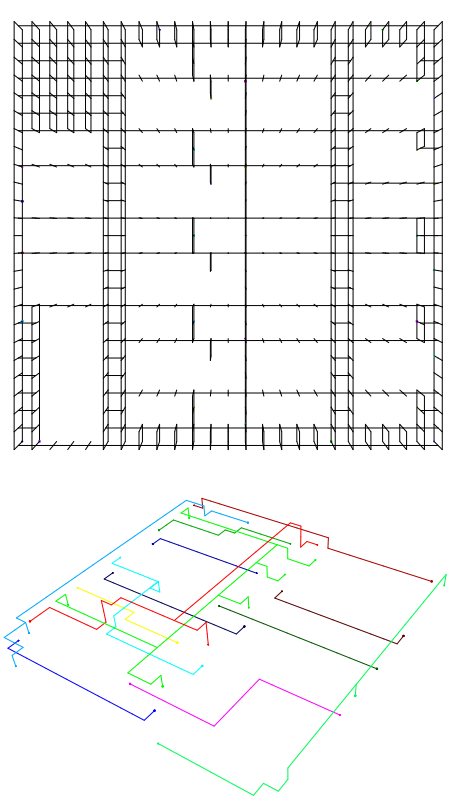
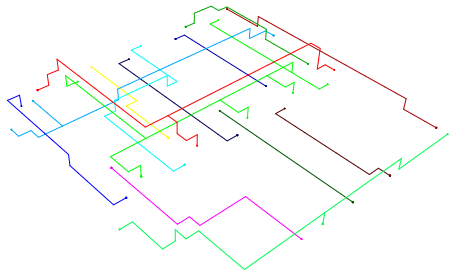


Figure 16:
sb60-60
aligned-layer

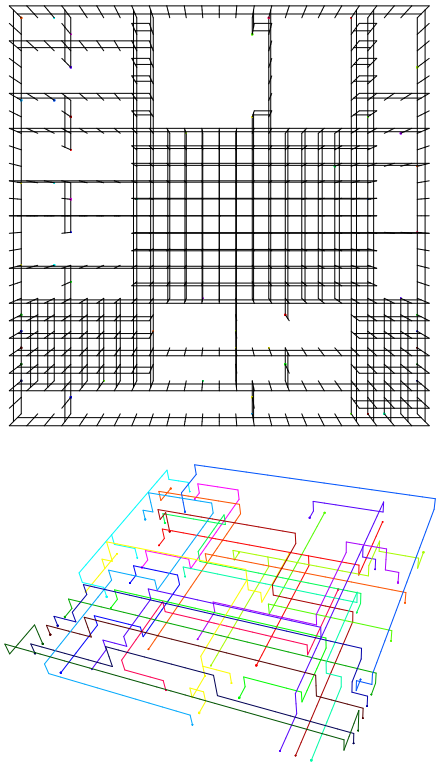


(a) Multi-aligned-layer

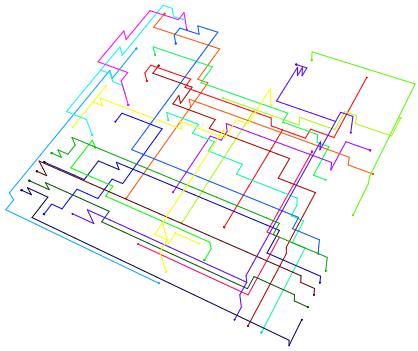


(b) Multi-crossed-layer

Figure 17: taq-3



(a) Multi-aligned-layer



(b) Multi-crossed-layer

Figure 18: alue-4