Vertex-Disjoint Packing of Two Steiner Trees: Polyhedra and Branch-and-Cut Implementation Issues

Eduardo Uchoa
e-mail: uchoa@inf.puc-rio.br
Marcus Poggi de Aragão
e-mail: poggi@inf.puc-rio.br

PUC-Rio.Inf.MCC 12/99 June, 1999.

Abstract: *Consider the problem of routing the electrical connections among two large terminal sets in circuit layout. A realistic model for this problem is given by the vertex-disjoint packing of two Steiner trees (2VPST), which is known to be NP-complete. This work presents an investigation on the 2VPST polyhedra. The main idea is to start from facet-defining inequalities for a vertex-weighted Steiner tree polyhedra. Some of these inequalities are proven to also define facets for the packing polyhedra, while others are lifted to derive new important families of inequalities, including proven facets. Separation algorithms are provided. Branch-and-cut implementation issues are also discussed, including some new practical techniques to improve the performance of the algorithm. The resulting code is capable of solving problems on grid graphs with up to 10000 vertices and 5000 terminals in a few minutes.*

Keywords: Steiner trees, VLSI circuit layout, Routing, Branch-and-Cut.

Resumo: *Considere o problema de rotear as conexões elétricas entre dois grandes conjuntos de terminais, seja no desenho de uma placa de circuito impresso ou em um circuito integrado VLSI. Um modelo realista para este problema é dado pelo empacotamento de duas árvores de Steiner disjuntas nos vértices (2VPST), um problema NP-completo. Esse trabalho investiga o poliedro do 2VPST. A principal idéia é partir de desigualdades que definem facetas para o poliedro do problema da árvore de Steiner com peso nos vértices. Provamos que algumas dessas desigualdades também definem facetas para o poliedro do empacotamento, enquanto outras podem ser submitidas a um procedimento de* lifting *para gerar novas famílias de desigualdades, incluindo facetas. Apresentamos procedimentos de separação e discutimos detalhes de implementação do algoritmo de* branch-and-cut*, incluindo algumas técnicas que parecem ser originais. O código resultante é capaz de resolver problemas em grafos de grade com até 10000 vértices e 5000 terminais em alguns minutos.*

Palavras-chave: Árvores de Steiner, Desenho de circuitos, Roteamento, Branch-and-Cut.

1 Introduction

The Vertex-disjoint Packing of Steiner Trees problem (VPST) is the following: given a connected graph $G = (V, E)$ and K disjoint vertex-sets $N_1, \ldots, N_K$, called *terminal sets*, find disjoint vertex-sets $T_1, \ldots, T_K$ such that, for each $k \in \{1, \ldots, K\}$, $N_k \subseteq T_k$ and $G[T_k]$ (the subgraph induced by T_k) is connected. Deciding whether a VPST instance is feasible or not is NP-complete, and many times a very challenging task. VPST has a wide application in layout of printed boards or chips, as it can model in an accurate way the routing of electrical connections among the circuit components (see Korte et al. [5] or Lengauer [6]).

This work studies the case where $K = 2$, the Vertex-Disjoint Packing of Two Steiner Trees (2VPST). This restricted problem is still NP-complete [5]. Although most circuit layout applications involves more than two terminal sets, many such problems have only two large terminal sets, for instance the ground and the V_{cc} terminals. Finding a solution for these two large sets may represent a major step in solving the whole routing problem. In fact, it is usual in chip layout to route the ground and V_{cc} connections over special layers, separated from the remaining connections. This particular routing problem leads to the 2VPST.

We address an optimization version of the 2VPST: find two disjoint vertex-sets T_1 and T_2, such that $G[T_1]$ and $G[T_2]$ are connected, minimizing $|N_1 \setminus T_1| + |N_2 \setminus T_2|$. In other words, we look for a solution leaving the minimum possible number of unconnected terminals. It is also allowed to set weights in order to specify which terminals have priority in being connected. These features may be very useful for a practitioner and suggests an iterative approach for layout: if the 2VPST instance is not feasible, a best possible solution gives valuable information on how to make small changes in terminal placement to achieve feasibility.

A previous work that is somehow related to ours is Grötschel et al. [2, 3], where Edge-disjoint Packing of Steiner Trees (EPST) is studied. EPST gives an alternative model for the routing problem in circuit layout. Their branch-and-cut code could solve switchbox instances (G is a grid graph and all terminals are placed in the outer face) with up to 345 vertices, 24 terminal sets, and 68 terminals.

2 Formulation

For each vertex $i \in V$ define a binary variable x_i, where $x_i = 1$ if i belongs to T_1 and $x_i = 0$ if i belongs to T_2. This choice of variables obliges every vertex to belong either to T_1 or T_2. Of course, terminals of N_1 in T_2 and terminals of N_2 in

T_1 are considered unconnected. A formulation for 2VPST follows.

$$
\text{(F)} \begin{cases}
\text{Min } z = & -\sum_{i \in N_1} w_i + \sum_{i \in V} w_i x_i & & (1) \\
\text{s.t.} & \sum_{i \in S} x_i - \sum_{i \in R} x_i \leq 1 & \forall S \subseteq V,\ |S| = 2;\ \forall R \in \Delta(S) & (2) \\
& -\sum_{i \in S} x_i + \sum_{i \in R} x_i \leq |R| - 1 & \forall S \subseteq V,\ |S| = 2;\ \forall R \in \Delta(S) & (3) \\
& x_i \in \{0, 1\} & \forall i \in V & (4)
\end{cases}
$$

Weight w_i must be negative if $i \in N_1$, positive if $i \in N_2$ and 0 otherwise. The minimum possible number of unconnected terminals matches z^* when $w_i = -1$ for all $i \in N_1$ and $w_i = 1$ for all $i \in N_2$. In any case the 2VPST instance is feasible iff $z^* = 0$. The symbol $\Delta(S)$ denotes the set of all minimal separators of S. A vertex-set R is a *separator of* S if any path joining the pair of vertices in S has at least one vertex in R. Constraints (F.2) says: if both vertices in S are in T_1, then at least one vertex in any separator of S must also be in T_1. Constraints (F.3) are complementary to (F.2): if both vertices in S are in T_2, then at least one vertex in any separator of S must be in T_2.

3 Polyhedral Investigations

Many combinatorial optimization problems can be naturally decomposed into simpler subproblems. The 2VPST decompose into two Steiner-like subproblems. There are cases where a good polyhedral description for the subproblems yields a good description of the main polyhedra. On the other hand, sometimes even a complete subproblem description yields but a poor description of the main polyhedra. Experiences with column generation showed that this is indeed the case of 2VPST [9]. In these situations, there is a need to also look for *joint inequalities*, i.e. inequalities which are not valid for any subproblem polyhedra but valid for the main polyhedra.

Definition 1 *Given a graph $G = (V, E)$, a* v-tree of G *is a vertex-set $T \subseteq V$ such that $G[T]$ is connected. A* complementary v-tree of G *is a vertex-set $T \subseteq V$ such that $G[V \setminus T]$ is connected. A* double v-tree of G *is a v-tree T of G that is also a complementary v-tree of G. (G being clear, we say v-tree or double v-tree for short).*

Definition 2 *Let χ^U be the incidence vector of a vertex-set U. Define $P(G)$ as the convex hull of vectors χ^T such that T is a double v-tree of G. Define $P_1(G)$ as the convex hull of vectors χ^T such that T is a v-tree of G and $P_2(G)$ as the convex hull of vectors χ^T such that T is a complementary v-tree of G.*

The vectors defining polyhedra P correspond to the solutions of formulation (F), so P is associated to 2VPST. Polyhedra P_1 and P_2 are associated to a Vertex Weighted Steiner Tree problem: given a graph with positive and negative vertex weights, find a tree with maximum/minimum total vertex weight. Note that P_1 and P_2 are complementary polyhedra and that every valid inequality for one can be converted into a valid inequality for the other by variable complementation, i.e. replacing x by $1 - x$. For instance, inequalities (F.2) are valid for P_1 and the complementary inequalities (F.3) are valid for P_2. Therefore whenever we talk about polyhedra P_1, it is implicit that similar results also hold for P_2.

Our idea is to use subproblem polyhedra P_1 and P_2 as a starting point for investigating main polyhedra P. We show that: (i) some inequalities that define facets of P_1 or P_2 also define facets of P and (ii) some of these inequalities which do not define facets of P can be lifted to derive new joint inequalities.

Theorem 1 *$P(G)$, $P_1(G)$ and $P_2(G)$ are full-dimensional.*

Proof: Let T be a spanning tree of G, rooted at an arbitrary vertex. For each $i \in V$, the vertex-set U_i containing i and his descendents in T is a double v-tree, so $\chi^{U_i} \in P(G)$. As $\chi^{\emptyset} \in P(G)$, there are $|V| + 1$ affine independent vectors in $P(G)$. As $P(G) \subseteq P_1(G) \cap P_2(G)$, the result follows. ■

3.1 The p-separator Facets of P_1

Definition 3 *Let S be a vertex-set, where $|S| \geq 2$. A vertex-set R is a* separator *of S if any path joining a pair of vertices in S has at least one vertex in R. A separator R of S is* proper *if $S \cap R = \emptyset$ and* minimal *if no other separator of S is contained in R. The symbol $\Delta(S)$ indicates the set of all proper and minimal separators of S. Given a separator R of S and a vertex $i \in V$, define S_i^R as the maximum subset of vertices in S that can belong to a v-tree T_i containing i and such that $T_i \cap R \subseteq \{i\}$. Note that a separator R of S is minimal iff $|S_i^R| \geq 2$ for each $i \in R$.*

For example, in graph $G1$ (Fig. 1) taking $S = \{a, c, h\}$, $R = \{b, e, k, i\}$ is a proper minimal separator of S, $S_b^R = \{a, c, h\}$, $S_e^R = \{a, h\}$, $S_j^R = \{a\}$ and $S_g^R = \emptyset$. The following theorem suggests that the concept of separator is fundamental to understand the facial structure of P_1. The results of this subsection are proved in the appendix.

Theorem 2 *Every facet of $P_1(G)$ can be defined by a trivial inequality ($-x_i \leq 0$ or $x_i \leq 1$) or by an inequality in format $\sum_{i \in S} \beta_i x_i - \sum_{i \in R} \alpha_i x_i \leq 1$, where β and α are vectors of positive numbers and R is a proper separator of S.*

We define a class of potentially facet-defining inequalities.

Definition 4 *Let S be a vertex-set, where $|S| = p \geq 2$, and $R \in \Delta(S)$. The following inequalities are called* p-separators, *where α_i is an integer greater than $|S_i^R| - 1$.*

$$\sum_{i \in S} x_i - \sum_{i \in R} \alpha_i x_i \leq 1$$

Constraints (F.2) are 2-separators. In graph $G1$, $x_a + x_f - x_b - x_e - x_j \leq 1$ is an example of such inequality. In the same graph, $x_a + x_f + x_g - x_b - 2x_e - 2x_k \leq 1$ is a 3-separator valid for $P_1(G1)$. Sometimes taking $\alpha_i = |S_i^R| - 1$ for all $i \in R$ does not yield a valid inequality for P_1. For instance, in graph $G2$ (Fig. 1) taking $S = \{a, b, c, d\}$ and $R = \{e, f\}$, the 4-separator $\sum_{i \in S} x_i - x_e - x_f \leq 1$ is not valid for $P_1(G2)$. But 4-separators $\sum_{i \in S} x_i - 2x_e - x_f \leq 1$ and $\sum_{i \in S} -x_e - 2x_f \leq 1$ are valid and facet-defining.

Theorem 3 *For each S and each $R \in \Delta(S)$, there is at least one p-separator inequality which is valid and facet-defining for $P_1(G)$.*

Corollary 1 *If the p-separator $\sum_{i \in S} x_i - \sum_{i \in R}(|S_i^R| - 1)x_i \leq 1$ is valid for $P_1(G)$, then it defines a facet of $P_1(G)$.*

The above theorem does not tell how to calculate the α coefficients of a p-separator facet. But given sets S and $R \in \Delta(S)$, the coefficients of a p-separator valid for P_1 can be computed by the following procedure:

Procedure Compute α
  $M \longleftarrow \emptyset$;
  For each $i \in R$ {
    If (exists $j \in M$ such that $S_i^R \cap S_j^R = \emptyset$)
      Then { $\alpha_i \longleftarrow |S_i^R|$; }
      Else { $\alpha_i \longleftarrow |S_i^R| - 1$; $M \longleftarrow M \cup \{i\}$; }
  }

As example, take graph G2 with $S = \{a, b, c, d\}$ and $R = \{e, f\}$. If we pick vertices in order (e, f), the procedure yields $\alpha_e = 1$ and $\alpha_f = 2$. If the order is reversed, it yields $\alpha_f = 1$ and $\alpha_e = 2$. Note that if $p \leq 3$, the procedure always yields $\alpha_i = |S_i^R| - 1$ for all $i \in R$. Then, by Corollary 1, all such 2-separators and 3-separators define facets of $P_1(G)$. The above procedure is not optimal, if $p \geq 4$ the resulting p-separators are not always facet-defining.

3.2 From P_1 to P: p-separators, Lifted p-separators and p-crosses

If a certain p-separator defines a facet of $P_1(G)$, it may also define a facet of $P(G)$, so it is interesting to investigate when it happens. Also interesting is to investigate

when this does not happen: in this case there is room for lifting the p-separator, producing a new joint inequality.

Definition 5 *The inequalities valid for $P(G)$ that can be obtained by lifting α coefficients of a p-separator facet of $P_1(G)$ or $P_2(G)$ are called* lifted p-separators.

The following definitions and lemma (proved in the appendix) are necessary for announcing and proving the main results of this section.

Definition 6 *Let T be a double v-tree of G and t a vertex in T. A* sequence of nested double v-trees from t to T *is any sequence of double v-trees $T_1, \ldots, T_{|T|}$ such that $\{t\} = T_1 \subset \ldots \subset T_{|T|} = T$. (In other words, for each i, $1 < i \leq |T|$, T_{i-1} may be obtained from T_i by removing a vertex distinct from t).*

Lemma 1 *Suppose G is biconnected. Given a double v-tree T of G and $t \in T$, it is always possible to construct a sequence of nested double v-trees from t to T.*

Definition 7 *A separator R of $S = \{s_1, \ldots, s_p\}$* induces *the following* structure *in G. Graph $G[V \setminus R]$ is split in a number of connected components: those containing exactly one vertex in S are called* A-components *and those with no vertices in S are called* B-components. *Define A_i as the vertex-set of the A-component containing s_i (when R is not proper some A-components do not exist). The vertex-sets corresponding to B-components are numbered arbitrarily as $B_1, \ldots, B_q$. Note that $|S_i^R| = 0$ if i belongs to a B-component and $|S_i^R| = 1$ if i belongs to a A-component.*

Theorem 4 *Suppose G is biconnected. Let $\sum_{i \in S} x_i - \sum_{i \in R}(|S_i^R| - 1)x_i \leq 1$ be a valid p-separator, where R induces a structure without B-components and that every A-component is a double v-tree. If for every $i \in R$, there is a double v-tree U_i containing $\{i\} \cup S_i^R$ such that $U_i \cap R = \{i\}$, this inequality defines a facet of $P(G)$.*

Proof: Let $a^t x \leq a_0$ be a valid inequality for $P(G)$ such that

$$F = \{x \in P(G) \mid \sum_{i \in S} x_i - \sum_{i \in R}(|S_i^R| - 1)x_i = 1\} \subseteq F_a = \{x \in P(G) \mid a^t x = a_0\} \ .$$

We show that $a^t x \leq a_0$ is an scalar multiple of this p-separator. Consider the structure induced by R. For every A-component A_j, applying Lemma 1, construct a sequence of nested double v-trees from s_j to A_j. For each double v-tree T in this sequence, $\chi^T \in F$. So $a_{s_j} = a_0$ and for every $i \neq s_j \in A_j$, $a_i = 0$. For every vertex $i \in R$, $\chi^{U_i} \in F$, where U_i is the double v-tree given by hypothesis. So $a_i = -(|S_i^R| - 1)a_0$. ■

For example, in graph G1, the 2-separator $x_a + x_f - x_b - x_e - x_j \leq 1$, 3-separator $x_a + x_f + x_g - x_b - 2x_e - 2x_k \leq 1$ and 4-separator $x_a + x_c + x_g + x_h - 2x_b - 2x_e - x_i - 2x_k \leq 1$ define facets of $P(G1)$. The converse of Theorem 4 does not hold even if $p = 3$, so we need some additional conditions to be able to lift α coefficients.

Theorem 5 *Suppose G is biconnected. Let $\sum_{i\in S} x_i - \sum_{i\in R}(|S_i^R| - 1)x_i \leq 1$ be a valid p-separator, where R induces a structure without B-components and that every A-component is a double v-tree. Suppose that there is a vertex $i \in R$ not satisfying Theorem 4 conditions. If $|S_i^R| = p$, the coefficient α_i can be lifted in at least one unit.*

Proof: We show that for every double v-tree T containing i, $\gamma_T = |S \cap T| - \sum_{i\in R\cap T} \alpha_i \leq 0$. If T contains S_i^R, $|S \cap T| = |S_i^R|$. As T must contain another vertex j of R, $\gamma_T \leq |S_i^R| - (|S_i^R| - 1) - (|S_j^R| - 1) = 2 - |S_j^R| \leq 0$. If T does not contain S_i^R, $\gamma_T \leq (|S_i^R| - 1) - (|S_i^R| - 1) \leq 0$. ■

If $p = 2$, the above results can be collected as follows:

Corollary 2 *Suppose G is biconnected. Let $\sum_{i\in S} x_i - \sum_{i\in R} x_i \leq 1$ be a 2-separator, where R induces a structure without B-components. This inequality defines a facet of $P(G)$ iff for every $i \in R$, there is a double v-tree U_i containing $\{i, s_1, s_2\}$ such that $U_i \cap R = \{i\}$*

For examples, in graph G1, the 2-separator $x_a + x_c - x_c - x_f - x_e - x_h - x_l \leq 1$ do not define a facet of $P(G1)$, by Corollary 2, α_e or α_h can be lifted to 0. The 3-separator $x_a + x_c + x_g - x_b - 2x_e - 2x_k \leq 1$ also do not define a facet $P(G1)$, by Theorem 5, α_e can be lifted to 1.

In graph G3 (Fig. 2), the 2-separator $x_a + x_b - x_d - x_e - x_f \leq 1$ is not facet-defining, although it satisfies the conditions of Corollary 2, except that R induces a structure with B-components. This 2-separator is dominated by the lifted 3-separator $x_a + x_b + x_c - x_d - x_e - x_f \leq 1$ (from 3-separator $x_a + x_b + x_c - 2x_d - 2x_e - 2x_f \leq 1$). In graph G4 (Fig. 2), 3-separator $x_a + x_c + x_e - x_b - x_d - x_f - x_h \leq 1$ is not facet-defining, although it satisfies the conditions of Theorem 4, except that the A-component formed by $\{c, g\}$ is not a double v-tree. This 3-separator is dominated by $x_a + x_c + x_g + x_e - x_b - x_d - x_f - x_h \leq 1$.

The concept of lifted p-separators acts as a convenient classifying device, permitting us to view lots of joint inequalities in a single framework. Even though Theorem 5 does not cover all cases when a p-separator facet of $P_1(G)$ can be lifted. It is certainly possible to improve these results, but technical difficulties appear. For instance, when $p \geq 4$ it is not clear even how to calculate the original α coefficients of a p-separator facet of $P_1(G)$. Also, it is usual that several such coefficients can be lifted at once. Conditions for this "multilifting" are more complicated. Most of all, it is cumbersome to design separation procedures for lifted p-separators starting from the original p-separator.

We give a direct characterization of an important subclass of lifted p-separators that permitted us to overcome such difficulties. These inequalities turned out to be essential to good branch-and-cut performance.

Definition 8 *Let $S = \{s_1, \ldots, s_p\}$ and $R = \{r_1, \ldots, r_p\}$ be two disjoint vertex-sets with $p \geq 2$. Suppose that for any integers a, b, c, d in the range $\{1, \ldots, p\}$, with $a \leq b < c \leq d$, there are no two vertex-disjoint paths joining s_a to s_c and r_b to r_d (in other words, the vertices in every path from r_b to r_d define a separator of $\{s_a, s_c\}$ and vice-versa). Then the following inequality is called a* p-cross

$$\sum_{i \in S} x_i - \sum_{i \in R} x_i \leq 1$$

In graph G1, $x_e + x_j - x_d - x_g \leq 1$ is a 2-cross and $x_a + x_c + x_l + x_j - x_b - x_i - x_k - x_d \leq 1$ is a 4-cross. Most p-crosses are lifted p-separators, it suffices to "complete" R until a proper minimal separator of S is found. For instance, 3-cross $x_a + x_c + x_k - x_b - x_l - x_d \leq 1$ is a lifting of 3-separator $x_a + x_c + x_k - 2x_b - x_f - x_l - x_d \leq 1$ facet of $P_1(G1)$ or 3-separator $-x_b - x_l - x_d + x_a + x_c + 2x_e + x_f + x_k \leq 4$ facet of $P_2(G1)$.

Theorem 6 *The p-crosses are valid for P.*

Proof: By induction on p. The 2-crosses are clearly valid. Suppose that $\sum_{i=1}^{p} x_{s_i} - \sum_{i=1}^{p} x_{r_i} \leq 1$ is a p-cross with $p \geq 3$. By induction hypothesis, $(p-1)$-crosses $\sum_{i=1}^{p-1} x_{s_i} - \sum_{i=1}^{p-1} x_{r_i} \leq 1$ and $\sum_{i=2}^{p} x_{s_i} - \sum_{i=2}^{p} x_{r_i} \leq 1$ are valid. Summing these inequalities with 2-cross $x_{s_1} + x_{s_p} - x_{r_1} - x_{r_p} \leq 1$, dividing the result by 2 and rounding down the right-hand-side, the desired p-cross is obtained. ■

Theorem 7 *If G is biconnected, every 2-cross defines a facet of $P(G)$.*

Proof: Let $x_{s_1} + x_{s_2} - x_{r_1} - x_{r_2} \leq 1$ be a 2-cross and $a^t x \leq a_0$ be a valid inequality for $P(G)$ such that

$$F = \{x \in P(G) \,|\, x_{s_1} + x_{s_2} - x_{r_1} - x_{r_2} = 1\} \subseteq F_a = \{x \in P(G) \,|\, a^t x = a_0\} \ .$$

We show that $a^t x \leq a_0$ is an scalar multiple of this 2-cross. As G is biconnected, there are two vertex-disjoint paths, except in the endpoints, joining r_1 to r_2. Let I_1 and I_2 be the set of vertices of any two such paths. As I_1 and I_2 are separators of S, it is not possible to have $S \subseteq I_1$ or $S \subseteq I_2$.

Case (a): I_1 or I_2 is a proper separator of S.

Suppose I_1 is a proper separator of S and consider the induced structure. As G is biconnected, each component is adjacent to at least two vertices in I_1. We assert that at least one of the two A-components is adjacent to both r_1 and r_2. If this is not true, the path corresponding to I_2 must pass through a B-component and I_1 would not be a separator of S. Suppose A_2 is an A-component adjacent to r_1 and r_2.

Subcase (a.1): A_1 is adjacent to a vertex in $I_1' = I_1 \setminus \{r_1, r_2\}$. In this case, $C_1 = A_1 \cup I_1' \cup \{B_k \,|\, B_k \text{ is adjacent to } I_1'\}$ and $C_2 = A_2$ are double v-trees. Construct

a sequence of nested double v-trees from s_1 to C_1 and from s_2 to C_2. For each T in these sequences, $\chi^T \in F$. So $a_{s_1} = a_{s_2} = a_0$ and $a_i = 0$ if $i \in (C_1 \cup C_2) \setminus S$. As $\chi^{C_1 \cup C_2 \cup \{r_1\}}$ and $\chi^{C_1 \cup C_2 \cup \{r_2\}} \in F$, $a_{r_1} = a_{r_2} = -a_0$. Each component B_k not adjacent to I_1' must be adjacent to both r_1 and r_2. Let j be a vertex in B_k adjacent to r_1. Construct a sequence of nested double v-trees from j to B_k. For each T in this sequence $U_T = C_1 \cup C_2 \cup \{r_1\} \cup T$ is also a double v-tree and $\chi^{U_T} \in F$. So $a_i = 0$, for all $i \in B_k$.

Subcase (a.2): A_1 is not adjacent to I_1'. In this case, A_1 is adjacent to r_1 and r_2. Let $C_1 = A_1$, $C_2 = A_2$ and proceed similar to case a.1, concluding that $a_{s_1} = a_{s_2} = a_0$, $a_i = 0$ if $i \in (C_1 \cup C_2) \setminus S$ and $a_{r_1} = a_{r_2} = -a_0$. Now define C_3 as $I_1' \cup \{B_k \mid B_k$ is adjacent to $I_1'\}$. Let j be a vertex in I_1' adjacent to r_1. Construct a sequence of nested double v-trees from j to C_3. For each T in this sequence $U_T = C_1 \cup C_2 \cup \{r_1\} \cup T$ is also a double v-tree and $\chi^{U_T} \in F$. So $a_i = 0$, for each $i \in C_3$. B-components adjacent to both r_1 and r_2 are treated in a similar way.

Case (b): I_1 and I_2 are not proper separators of S.

Suppose I_1 contains s_1 and I_2 contains s_2. Consider the structure induced by separator I_1. As I_2 contains s_2, A_2 is adjacent to r_1 and r_2. Let $C_1 = I_1' \cup \{B_k \mid B_k$ is adjacent to $I_1'\}$, $C_2 = A_2$ and proceed similar to subcase a.1. B-components adjacent to both r_1 and r_2 are also treated like in a.1. ■

Using similar techniques it is possible to prove that under mild conditions p-crosses with $p \geq 3$ also define facets of $P(G)$.

4 Separation Algorithms

We focus our experiments on single layer circuit layout like instances, where G is planar. Therefore we also developed special separation algorithms for this case. Let $\bar{x}$ be the current LP solution.

4.1 p-separators

Given a set $S = \{s_1, s_2\}$, (the choice of such sets S is discussed as part of the separation strategy, see Sec. 5.3) checking if there is a violated 2-separator over S amounts to finding a minimum cost separator. It can be done by a min-cut algorithm applied over a suitable network. But if G is planar, there is a more efficient way of computing these minimum cost separators by applying a shortest path algorithm. This method is an adaptation of the algorithm described in Reif [10], we only give an outline of it:

Case 1: *Vertices s_1 and s_2 are in the same face f of G.* Produce graph G' by the following operations: (i) If $g \neq f$ is face of G with vertex-set $V(g)$, introduce new edges to make $G'[V(g)]$ complete, (ii) introduce two new vertices v_1 and v_2, v_1 is adjacent to all vertices in one of the paths over f from s_1 to s_2 and v_2 is

adjacent to vertices on the other such path, and (iii) remove vertices s_1 and s_2. The minimum cost separator of S corresponds to a minimum vertex cost path from v_1 to v_2 in G'. Vertex costs are given by $\bar{x}$, new vertices receive 0 cost. For instance, in graph $G1$ with $S = \{a, l\}$ the resulting graph $G1'$ is depicted in Fig.3. If a face g has $|V(g)| \geq 5$, it may be advantageous to change step (i), instead of completing $G'[V(g)]$, add a new vertex adjacent to all vertices in $V(g)$.

Case 2: *Vertices s_1 and s_2 are not in the same face of G.* Suppose w.l.o.g. that s_1 is in the outer face f of G. The idea is to make a *section* of G, splitting some vertices in order to "expose" s_2, i.e. to make s_2 also be in the outer face. For instance, if $G5$ is the 5×5 grid graph shown in Fig.4, $s_1 = u$ and $s_2 = m$, a possible section is to split vertices n and o, as indicated by the dotted line. Then we apply a transformation similar to case 1. The resulting graph $G5'$ is also shown in Fig.4. In this case, the minimum cost separator of S can be found by applying the shortest path algorithm 3 times. Finding the shortest path from v_1 to v_2, a separator like $\{k, q, w\}$ may be found. Finding the path from n to n' separators like $\{n, h, l, r\}$, $\{n, i, d, p, q, r\}$ or $\{n, h, l, q, v, t\}$ may be found. Finally, from o to o'.

When $p \geq 3$, the procedures to find violated p-separators are harder to devise. A heuristic separation for 3-separators follows: (i) Select $S = \{s_1, s_2, s_3\}$, (ii) find a minimum cost separator R_1 of $\{s_1, s_2\}$ with cost $\bar{x}(R_1)$ (iii) if there is a component of $G[V \setminus R_1]$ with vertex-set A containing two vertices in S, say s_2 and s_3, find a separator R_2 in $G[A \cup R_1]$ of $\{s_2, s_3\}$ with cost $\bar{x}(R_2)$. Now $R = R_1 \cup R_2$ is a separator of S and if $\bar{x}_{s_1} + \bar{x}_{s_2} + \bar{x}_{s_3} - \bar{x}(R_1) - \bar{x}(R_2) > 1$, a violated 3-separator is found. Note that if a vertex $i \in R$ has coefficient $\alpha_i = 2$ in the 3-separator, its cost is correctly counted twice. For instance, in graph $G5$ with $S = \{a, i, q\}$, R_1 may be $\{d, h, l, p\}$ and R_2 may be {l,r,x}.

4.2 p-crosses

The p-crosses can be separated in polynomial time over any graph G, for fixed p. This follows from definition 8, since there exists an $O(|V|.|E|)$ algorithm to determine if it is possible to connect two given pairs of vertices by vertex-disjoint paths (Shiloach [11]). However, far better p-cross separation schemes may exist if the particular structure of G is considered.

When G is planar, we look for violated p-crosses where the sequence $s_1, r_1, \ldots, s_p, r_p$ happen in this cyclic order on some face f of G. If G is also 3-connected, these are the only p-crosses. Even if G is not 3-connected, the few p-crosses with vertices in distinct faces seems to be equivalent or dominated by p-separators. For example, in graph $G5$, the 2-cross $x_u + x_m - x_p - x_v \leq 1$ is also a 2-separator. The p-crosses over the same face can be separated in $O(|V|)$ time by dynamic programming:

Procedure Separate p-cross over Planar Graphs
  For each face f of G {
    $A[0] \longleftarrow -\infty$; $B[0] \longleftarrow 0$;
    $C[0] \longleftarrow 0$; $D[0] \longleftarrow -\infty$;
    For each vertex i in cyclic order of f (numbered from 1 to $|V(f)|$) {
      $A[i] \longleftarrow \text{MAX}(A[i-1], B[i-1] + \bar{x}_i - 1)$;
      $B[i] \longleftarrow \text{MAX}(B[i-1], A[i-1] - \bar{x}_i + 1)$;
      $C[i] \longleftarrow \text{MAX}(C[i-1], D[i-1] + \bar{x}_i - 1)$;
      $D[i] \longleftarrow \text{MAX}(D[i-1], C[i-1] - \bar{x}_i + 1)$;
    }
    Iff $(\text{MAX}(B[|V(f)|], C[|V(f)|]) > 1)$ there is a violated p-cross over f;
  }

There are two independent recursions. The first one is over vectors A and B. $A[i]$ gives the greatest *lhs* value (minus 1) of a partial p-cross up to vertex i *unbalanced to* S, i.e. a partial p-cross with format $s_1, r_1, \ldots, s_j$. $B[i]$ is the greatest *lhs* value of a *balanced* p-cross with format $s_1, r_1, \ldots, s_j, r_j$. Therefore, violation is indicated by $B[|V(f)|] > 1$. The other recursion is similar. $D[i]$ is the greatest *lhs* value (plus 1) of a partial p-cross up to vertex i unbalanced to R and $C[i]$ of a balanced p-cross $r_1, s_1, \ldots, r_j, s_j$. Violation is indicated by $C[|V(f)|] > 1$.

5 Branch-and-Cut Implementation Issues

5.1 Using Primal Information to Get Stronger Cuts

If a solution with cost z_{inc} is already known, there is an effective way to lift p-separators, even when they are facet-defining:

Theorem 8 *Let $\sum_{i \in S} x_i - \sum_{i \in R} \alpha_i x_i \leq 1$ be a valid p-separator and $S^1 = \{s_j \in S \mid -(\sum_{i \in A_j \cap N_1} w_i) \geq z_{inc}$ or A_j has a variable fixed to $1\}$. The p-separator can be lifted to $\sum_{i \in S \setminus S^1} x_i - \sum_{i \in R} \alpha_i x_i \leq 1 - |S^1|$.*

Proof: Any double v-tree T with cost better than z_{inc} contains at least one vertex ${s_j}^T$ in each A-component A_j corresponding to vertices in S^1. Let S^T be the set of those vertices ${s_j}^T$. As the p-separator $\sum_{i \in S \setminus S^1} x_i + \sum_{i \in S^T} x_i - \sum_{i \in R} \alpha_i x_i \leq 1$ is valid for T, then the lifted p-separator is also valid for T. ■

5.2 Primal Heuristic

Before the branch-and-cut is started, a greedy solution is constructed and $2|V|$ iterations of a tabu search are spent to improve it. The neighborhood is defined

by all swaps of a single vertex from T_1 to T_2 or vice-versa, without disconnecting $G[T_1]$ or $G[T_2]$. This heuristic performed poorly in the experiments. The solutions obtained were only good enough to provide an initial value z_{inc} which allows lifting most of the violated 2-separators found.

5.3 Separation Strategy

We used the following general approach:

General Separation Strategy
For each "family" of cuts {
    All variables are marked as unused;
    Repeat
        Find (i) a most violated cut over the unused variables,
            (ii) using the minimum number of free (non-fixed) variables,
            (iii) forcing more the objective function;
        Mark the free variables in this cut as already used;
    Until no violated cut is found or some limit for this family is reached
}

We want to separate "high-quality" cuts. Besides finding a most violated one, we have a second tie-breaking criterion of finding a cut over the minimum number of free variables. A similar idea was successfully used by Koch and Martin [4]. By "forcing the objective function", we mean that a cut $a^T x \leq a_0$ must minimize $a^T w$. Even as the third tie-breaking criterion, this idea proved to be very useful. It seems important not to forget the original costs while separating. A cut satisfying all such criteria may be found by standard separation procedures, after introducing suitable perturbations in vector $\bar{x}$. Strongly polynomial separation procedures are recommended, since such perturbations increase substantially the size of the numbers in the input.

The idea of separating many violated cuts by family, as long as their support sets have little or no overlap, is used in some recent branch-and-cut implementations, including [4].

5.3.1 p-crosses

For each face f of G we define a family of cuts and follow the general separation strategy without any limit. In other words, for each face as many non-overlapping violated p-crosses as possible are added in each cut round. Such policy is efficient because p-crosses are (i) very cheap to separate over planar graphs, (ii) very sparse, therefore they do not burden the LPs and (iii) very strong cuts in practice.

5.3.2 2-separators

We use the following procedure to choose a convenient small vertex-set M to form sets $S = \{s1, s2\}$. Let $\bar{D}(i)$ be the set of vertices that can reached from i by a "decreasing path", i.e. a oriented path P from i such that for each arc $(a, b) \in P$, $\bar{x}_a \geq \bar{x}_b$.

Procedure Choose M
  Sort the vertices by:
    (i) vertices fixed to 1 first,
    (ii) higher values $\bar{x}_i$ first,
    (iii) A linear combination of $-w_i$ and the distance of i from the outer face;
  $M \longleftarrow \emptyset$;
  Repeat
    $i \longleftarrow$ The first non-marked vertex;
    $M \longleftarrow M \cup \{i\}$;
    Mark the vertices in $\bar{D}(i)$;
  Until all vertices are marked;

We perform separation over $|M| - 1$ families of 2-separators defined as follows: fix s_1 to be the first vertex included in M and let s_2 be one of the remaining vertices in M. The general separation strategy is applied over each such family S. Note that several cuts may be added with the same set S, if the corresponding sets R are non-overlapping. Since finding minimum cost separators is not so cheap and 2-separators may be dense enough to burden the LPs, we have a cut limit: keep separating over S until a cut violated by less than 0.5 is found.

In almost all cut rounds, s_1 is in the outer face. In order to improve the performance of the algorithm that finds minimum cost separators (Sec. 4.1), we prefer to have other vertices in M near the outer face.

5.3.3 3-separators

The separation heuristics for 3-separators are rather expensive and such cuts have a limited practical importance. We only try to find 3-separators once per node, in the first time no other violated cuts are found. We take advantage of the information gathered while looking for 2-separators. If a minimum cost separator over $S = \{s1, s2\}$ is R_1 with $\bar{x}_{s_1} = \bar{x}_{s_2} = \bar{x}(R_1) = 1$ and R_1 have vertices with fractional $\bar{x}$, we select a third vertex $s_3 \in M$ with $\bar{x}_{s_3} = 1$. Applying the heuristic described in Sec. 4.1, if $\bar{x}(R_2) < 1$ a violated 3-separator is found.

5.4 Enumeration Strategy

Our branch-and-cut for the 2VPST have the following characteristics: (i) the lower bound z_r provided by the root node (its LP value rounded up) is very good,

in fact z^* is usually equal to z_r or yet, to the next integer and (ii) there are many alternative optimal solutions, but they may be a little deep in the branch-and-bound tree. These characteristics are shared by other proposed enumeration schemes. In general, due to (ii), *best first* strategy (explore the node with smaller LP value) may not perform well, as it has a tendency to search the tree in breadth. On most instances, *depth first* (explore the deepest node) is better, but sometimes it wastes a lot of time exploring branches without good solutions.

Therefore we used a mixed strategy: explore the deepest node whose LP value rounded up is equal to the current lower bound. This works fine in practice as it combines the advantages of both methods: it can find optimal solutions quickly, as depth first, but without the risk of losing too much time time in bad branches. This strategy still have the same theoretical guarantee as best first, i.e. no other strategy can explore less nodes in the worst case.

5.5 Dynamic Cost Perturbation

A 2VPST instance can have a large number of alternative solutions with the same cost. This kind of symmetry may bring serious difficulties for a naive branch-and-cut code: the LPs are very degenerate and there is a tendency to tail-off, i.e. to perform many cut rounds without significant changes on the LP values. In such cases, it is usual to add small random perturbations to variable costs in order to reduce this effect (for example, [3] and [4]).

We introduced perturbations in our code, but verified that it is better not do it absolutely at random. For each vertex i not yet perturbated, calculate $n_i = \sum_{j \in N(i)} w_j$, where $N(i)$ is the set of vertices adjacent to i. If n_i is positive, add a small positive random perturbation to w_i; if n_i is negative, a negative perturbation. We repeat the procedure until all vertices are perturbated. The idea works fine, in fact without perturbations our code simply can not solve large instances.

A further and most remarkable improvement was due to the idea of dynamic cost perturbation. When branching over a variable i, for each non-terminal vertex j such that $\bar{x}_j$ is fractional, we also reverse the sign of the perturbation of j (i.e. multiply w_j by -1) in both child nodes. The dynamic cost perturbation turned out to be very effective, it can drive a solution with many fractional variables into an "almost integer" solution with same cost. It saves a lot of branchings.

5.6 Miscellanea

We start with an empty LP, no constraints other than the lower and upper bounds on the variables are present. In order to keep the LPs small, after each cut round we remove constraints with basic slack variables. The removed constraints are discarded, no pool is used. We adopt the branching over a variable, preferring

variables with current value close to 0.5, with large absolute cost and on the outer face.

6 Computational Results

The branch-and-cut code was implemented in C++ using the ABACUS framework [12] and CPLEX 3.0 LP solver. Times were obtained on a Sun ULTRA workstation. In all experiments described in Table 7, G is a square grid graph (grid graphs are typical in circuit layout applications). Two kind of instances were generated:

- Unweighted instances with $|N_1| = |N_2| = |V|/4$. The terminals in each of these sets are randomly chosen among the vertices of G. Vertices in N_1 have weight -1 and vertices in N_2 have weight 1. These instances are referenced as u_n_$Inst$, where $n = |V|$ and $Inst$ is the instance number. The value z^* gives the number of unconnected terminals in an optimal solution.

- Weighted instances. For each unweighted instance we generated a corresponding weighted instance, by picking an integer w_i from an uniform distribution in the range $[-10, -1]$ if $i \in N_1$ and from $[1, 10]$ if $i \in N_2$. The value z^* gives the sum of the absolute weights of the unconnected terminals in an optimal solution. Weighted instances are referenced as w_n_$Inst$.

In Table 7, the $\mathbf{LP}_r$ column presents the LP value obtained after solving the root node, $\mathbf{z}^*$ is the value of the optimal integer solution, **Nd** is the number of nodes explored in the branch-and-bound tree, **nLPs** is the number of solved LPs (which also indicates the number of cut rounds), **nCuts** is the total number of added cuts, **Time** is the total cpu time in seconds spent by the algorithm and **%LP** is the percentual of this time spent by solving LPs. Remark that LP_r values includes p-crosses and some 3-separators. The lower bounds given by the linear relaxation of formulation (F) alone are rather weak, for example, in instance u1600_1 it is 8.7. No instance was solved in the root node, but 9 out of 28 instances were solved in the second node. This quite unusual distribution is due to the dynamic cost perturbation, the root node is unlikely to produce integral solutions, but after changing the perturbations in the child nodes they may be found. Figure 5 shows an optimal solution of instance u1600_1, the black boxes represent the terminals in N_1, the empty boxes are the terminals in N_2, the full lines are edges of a spanning tree for $G[T_1]$ and the dotted lines a spanning tree for $G[T_2]$.

The code turned out to be robust under changes on terminal placement, terminal density and weight distribution. In fact, we used the same set of parameters for solving all instances in table 1, weighted and unweighted. An interesting experience is to generate a random terminal placement avoiding the most obvious

causes of infeasibility: terminals on the outer face, two pairs of opposite terminals crossed over the same grid square and terminals blocked by its neighbors. An optimal solution of such an instance (called t1600_1) with 800 terminals and unitary weights is shown in Fig. 6. The unconnected terminals are in coordinates (4,33), (8,21), (33,36) and (34,15). This solution was obtained in 21 seconds.

7 Final Remarks

A more usual formulation of 2VPST is over binary variables x^1 and x^2, such that $x_i^1 = 1$ iff vertex i belongs to T_1 and $x_i^2 = 1$ iff i belongs to T_2. It is necessary to include packing constraints $x_i^1 + x_i^2 \leq 1$. But the formulation over variables x is superior: (i) it saves variables and constraints and (ii) it is equivalent of using partitioning constraints $x_i^1 + x_i^2 = 1$, it leads to smaller polyhedra and inequalities defining facets of these polyhedra are likely to be stronger cuts.

It is possible to use extended formulations that give a better description of subpolyhedra P_1 and P_2. An example of such a formulation follows. For each edge $e \in E$ define variables z_e^1 and z_e^2, where $z_e^1 = 1$ iff e belongs to a spanning tree of T_1 and $z_e^2 = 1$ iff e belongs to a spanning tree of T_2.

$$
(\text{EF}) \begin{cases}
\text{Min } z = & -\sum_{i \in N_1} w_i + \sum_{i \in V} w_i x_i & & (1) \\
\text{s.t.} & \sum_{e \in E} z_e^1 \geq \sum_{i \in V} x_i - 1 & & (2) \\
& \sum_{e \in E(S)} z_e^1 \leq \sum_{i \in S \setminus \{d\}} x_i & \forall S \subseteq V,\ \forall d \in S & (3) \\
& \sum_{e \in E} z_e^2 \geq -\sum_{i \in V} x_i + |V| - 1 & & (4) \\
& \sum_{e \in E(S)} z_e^2 \leq -\sum_{i \in S \setminus \{d\}} x_i + |S| - 1 & \forall S \subseteq V,\ \forall d \in S & (5) \\
& z_e^1, z_e^2 \geq 0;\ x_i \in \{0,1\} & \forall e \in E;\ \forall i \in V & (6)
\end{cases}
$$

Constraints (EF.3) and (EF.5) are called *generalized subtour elimination constraints* and were proposed independently by Lucena [7] and Goemans [1] for the Steiner problem. They can be separated in polynomial time by $O(|V|)$ applications of the min-cut algorithm. Constraints (EF.2) and (EF.3) provide a quite better description of P_1 than (F.2). Respectively, (EF.4) and (EF.5) are better than (F.3). However, this do not lead to a significant improvement of the description of P. Both formulations are really strengthened by the adition of joint inequalities, like the p-crosses. Therefore we prefer to work with weaker formulation (F), as it leads to much lighter LPs.

The implementation details described in Sec.5 are essential to good performance of this branch-and-cut. For example, our first straightforward implementation could not solve instance u1600_1 in 6 hours of cpu time. Remark that this first implementation already used the fast separation algorithms given in Sec.4. It appears that some of the implementation techniques presented, like considering the original costs while separating and the dynamic cost perturbation, may be useful in other problems.

Appendix

Proof of Theorem 2: Suppose that a facet F of $P_1(G)$ is defined by $a^t x \leq a_0$. The right-hand-side a_0 must be non-negative, otherwise this inequality would not be valid for the empty v-tree. So, w.l.o.g. we can suppose that $a_0 = 0$ or $a_0 = 1$. Let S be the vertex-set corresponding to positive coefficients in a and R the vertex-set corresponding to negative coefficients.

Case (a): $a_0 = 0$. The set S must be empty, otherwise $a^t x \leq 0$ would not be valid for the v-tree formed by a single vertex in S. The set R must be a singleton $\{u\}$, since if there are other vertices in R, $a^t x \leq 0$ would be dominated by $-x_u \leq 0$. So F can be defined by this trivial inequality.

Case (b): $a_0 = 1$. S can not be empty. If S is a singleton $\{u\}$, F can be defined by the trivial inequality $x_u \leq 1$. Suppose $|S| \geq 2$. Graph $G[V \setminus R]$ is split into a number of connected components. No two vertices in S can belong to the same such component, since if u and w are two vertices in S in the same component then $a^t x \leq 1$ would be a linear combination of valid inequalities $\sum_{i \in S \setminus \{u,w\}} a_i x_i + (a_u + a_w) x_u + \sum_{i \in R} a_i x_i \leq 1$ and $\sum_{i \in S \setminus \{u,w\}} a_i x_i + (a_u + a_w) x_w + \sum_{i \in R} a_i x_i \leq 1$. Therefore R is a proper separator of S. ■

The following lemma and definition are used in the proof of Theorem 3.

Lemma 2 *(Sequential Maximum Lifting Lemma, pages 261-263 of [8]) Let P be a polyhedron in $\mathbb{R}^n$. Let N be the set of indices $\{1, \ldots, n\}$ and Z_0 a subset of N. Define P^{Z_0} as the polyhedron $P \cap \{x \in \mathbb{R}^n | x_i = 0, i \in Z_0\}$. Suppose $\sum_{i \in N \setminus Z_0} a_i x_i \leq a_0$ is a valid inequality for P^{Z_0} defining a face of dimension d. Let $\sum_{i \in N \setminus Z_0} a_i x_i + \sum_{i \in Z_0} m_i x_i \leq a_0$ be any inequality obtained by a sequential maximum lifting of one coefficient in Z_0 at a time (in any order). This lifted inequality defines a face of P of dimension $d + |Z_0|$, and if the original inequality defines a facet of P^{Z_0}, the lifted inequality defines a facet of P.*

Definition 9 *Let T be a v-tree of G and t a vertex in T. A* sequence of nested v-trees from t to T *is any sequence of v-trees $T_1, \ldots, T_{|T|}$ such that $\{t\} = T_1 \subset \ldots \subset T_{|T|} = T$. Note that given v-tree T and t, is always possible to construct such a sequence.*

Proof of Theorem 3: Consider the structure induced by R (see definition 7). Inequality $\sum_{i\in S} x_i \leq 1$ is valid for $P_1(G)^{Z_0}$, where $Z_0 = R \cup (\cup_{j=1}^{q} B_j)$. Define $F = \{x \in P_1(G)^{Z_0} | \sum_{i\in S} x_i = 1\}$. For each $j \in \{1, \ldots, p\}$, A_j is a v-tree. Construct a sequence of nested v-trees from s_j to A_j. For every T in these sequences, $\chi^T \in F$. So there are $\sum_{j=1}^{p} |A_j|$ affine independent vectors in F and $\sum_{i\in S} x_i \leq 1$ defines a facet of $P_1(G)^{Z_0}$.

Number the vertices of R from r_1 to $r_{|R|}$. Define $R_0 = \emptyset$ and R_j as the set $\{r_1, \ldots, r_j\}$. For each $k \in \{1, \ldots, q\}$, define $f(k)$ as the smaller number of a vertex in R adjacent to B-component B_k. Define $W_j = R_j \cup \{B_k | f(k) \leq j\}$ and $Z_j = Z_0 \setminus W_j$. We show that for each $j \in \{0, \ldots, |R|\}$, there is an inequality $\sum_{i\in S} x_i - \sum_{i\in R_j} \alpha_i x_i \leq 1$, where each α_i is integer and satisfies $\alpha_i \geq |S_i^R| - 1$, defining a facet of $P_1(G)^{Z_j}$.

This is true when $j = 0$. Suppose $1 \leq j \leq |R|$. Pick the inequality corresponding to $j - 1$. First lift the coefficient of r_j. As there exists a v-tree T_j such that $T_j \cap R = \{r_j\}$, $T_j \cap S = S_j^R$ and $T_j \cap (\cup_{j=1}^{q} B_j) = \emptyset$, $\alpha_j \geq |S_j^R| - 1$. As the lifting is maximum, α_j is integral and there exists a v-tree $U \subseteq (\cup_{j=1}^{p} A_j \cup W_{j-1} \cup \{r_j\})$ such that $r_j \in U$ and $|S \cap U| - \sum_{i\in R_j \cap U} \alpha_i = 1$. For each B-component B_k such that $f(k) = j$, take a vertex $t \in B_k$ adjacent to r_j. Construct a sequence of nested v-trees from t to B_k. Make a maximum lifting of the coefficients of vertices in B_k in the same order this vertices are introduced in the sequence. Since $U \cup T$ is a v-tree for every T in the sequence, all such coefficients must be zero. Repeating this procedure until $j = |R|$, a p-separator defining a facet of $P_1(G)$ is obtained. ■

Proof of correctness of procedure Calculate α: Let U be a v-tree and let $Q = R \cap U$. We must show that $|S \cap U| - \sum_{i\in Q} \alpha_i \leq 1$. Assume that Q is not empty, otherwise this inequality is clearly true. Sort the vertices in Q as $q_1, \ldots, q_{|Q|}$, using the same order in which these vertices were considered in procedure Calculate α. Define Q_i as the set $\{q_1, \ldots, q_i\}$ and $S_{Q_i}^R$ as $\cup_{j\in Q_i} S_j^R$.

We assert that for each $i \in \{1, \ldots, |Q|\}$, $\gamma_i = |S_{Q_i}^R \cap U| - \sum_{i\in Q_i} \alpha_i \leq 1$. This is true when $i = 1$. Suppose $1 < i \leq |Q|$. If $\alpha_i = |S_i^R|$, then $\gamma_i \leq \gamma_{i-1} \leq 1$. Suppose $\alpha_i = |S_i^R| - 1$. The procedure assures that if there exists a vertex $j \in Q_{i-1}$ with $\alpha_j = |S_j^R| - 1$ then $S_i^R \cap S_j^R \neq \emptyset$, and $\gamma_i \leq \gamma_{i-1} \leq 1$. If there is no such vertex $\gamma_{i-1} \leq 0$, so $\gamma_i \leq 1$. As $|S \cap U| - \sum_{i\in Q} \alpha_i \leq \gamma_{|Q|} \leq 1$, the result follows. ■

Proof of Lemma 1: If $T = \{t\}$ the result is immediate. Otherwise, suppose that a partial sequence $T = T_{|T|} \supset \ldots \supset T_i$, $|T| \geq i > 1$, is already formed. Vertices in double v-tree T_i, different from t, but adjacent to a vertex in $V \setminus T_i$ will be called *candidate vertices* (if $T = V$, all vertices in $V - t$ are candidates). As G is biconnected, there is at least one candidate vertex. We show that is always possible to choose a candidate to be removed from T_i to form double v-tree T_{i-1}.

Let v_0 be a candidate. If $T_i - v_0$ is a v-tree, v_0 is chosen. Otherwise $G[T_i - v_0]$ is separated into $j \geq 2$ connected components with vertex-sets $C_1^1, \ldots, C_j^1$. As T_i is a v-tree, at least one vertex in C_j^1 is adjacent to v_0. As G is biconnected, at

least one vertex in each C_j^1 is adjacent to $V \setminus T_i$. Adjust notation for C_1^1 to be a set not containing t. Let v_1 be a candidate vertex in C_1^1. $G[C_1^1 - v_1]$ is separated into $j \geq 0$ connected components with vertex-sets $C_1^2, \ldots, C_j^2$. As C_1^1 is a v-tree, at least one vertex in each C_j^2 is adjacent to v_1. As G is biconnected, at least one vertex in each C_j^2 is adjacent to v_0 or to $V \setminus T_i$. If at least one vertex in each C_j^2 is adjacent to v_0 (or $j = 0$), v_1 is chosen. Otherwise, adjust notation for C_1^2 to be a set containing a candidate vertex.

By repeating this procedure, we get a sequence $v_1, \ldots, v_k$ of candidate vertices contained in smaller sets C_1^k. If each component of $G[C_1^k - v_k]$ is adjacent to $T_i \setminus C_1^k$, v_k is the chosen candidate and $T_i - v_k$ is a v-tree. In the worst case, the procedure is repeated until $C_1^k = \{v_k\}$. ■

References

[1] M. Goemans, *The Steiner Tree Polytope and Related Polyhedra*, Mathematical Programming, Vol. 63, 157-182, 1994.

[2] M. Grötschel, A. Martin and R. Weismantel, *Packing Steiner Trees: polyhedral investigations*, Mathematical Programming, Vol. 72, 101-123, 1996.

[3] M. Grötschel, A. Martin and R. Weismantel, *Packing Steiner Trees: a cutting plane algorithm and computational results*, Mathematical Programming, Vol. 72, 125-145, 1996.

[4] T. Koch and A. Martin, *Solving Steiner Tree Problems in Graphs to Optimality*, Networks, Vol. 32, 207-232, 1998.

[5] B. Korte, H. Prömel and A. Steger, *Steiner Trees in VLSI-Layout*, in B. Korte, L. Lovász, H. Prömel, A. Schrijver (Eds.) "Paths, Flows and VLSI-Layout", Springer-Verlag, Berlin, 1989.

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

[7] A. Lucena, *Tight Bounds for the Steiner Problem in Graphs*, IRC for Process Systems Engineering, Imperial College, London, 1993.

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

[9] M. Poggi de Aragão and E. Uchoa, *The γ-Connected Assignment Problem*, to appear in European Journal of Operational Research, 1998.

[10] J. Reif, *Minimum s-t Cut of a Planar Undirected Network in $O(n \log^2(n))$ Time*, SIAM Journal on Computing, Vol. 12, No.1, 71-81, 1983.

[11] Y. Shiloach, *A Polynomial Solution to the Undirected Two Paths Problem*, Journal of the ACM, Vol. 27, No.3, 445-456, 1980.

[12] S. Thienel, *ABACUS - A Branch-And-Cut System*, PhD thesis, Universität zu Köln, 1995.

Instance	LP_r	z^*	Nd	nLPs	nCuts	Time	%LP
u1600_1	45.7	46	4	96	2881	20	43
u1600_2	40.0	40	2	40	529	8	24
u2500_1	46.0	46	2	56	1118	33	20
u2500_2	55.0	55	3	49	2049	32	31
u3600_1	76.5	77	2	45	2084	41	37
u3600_2	74.0	74	4	57	1432	50	33
u4900_1	99.1	100	4	116	4380	170	32
u4900_2	92.0	92	3	66	2809	108	32
u6400_1	104.0	104	3	126	5808	443	30
u6400_2	127.0	128	4	123	6384	322	36
u8100_1	131.5	132	4	217	10271	1207	25
u8100_2	115.0	115	6	125	6734	713	22
u10000_1	159.8	161	8	120	6440	676	34
u10000_2	151.5	152	4	157	9870	991	27
w1600_1	222.0	222	2	35	1174	9	40
w1600_2	218.0	218	2	25	602	5	34
w2500_1	233.0	233	2	37	1113	14	42
w2500_2	227.5	228	3	39	810	17	32
w3600_1	364.0	364	2	48	1598	33	40
w3600_2	311.0	311	2	74	1715	54	29
w4900_1	420.0	420	2	77	2847	130	29
w4900_2	400.0	400	3	53	2305	90	34
w6400_1	436.0	436	3	106	3509	295	28
w6400_2	534.0	534	3	128	4915	367	28
w8100_1	549.5	550	4	113	4020	563	18
w8100_2	467.0	467	3	82	3175	264	28
w10000_1	685.0	685	3	60	3544	345	27
w10000_2	632.5	634	7	116	4907	621	38

Table 1: Branch-and-Cut Results

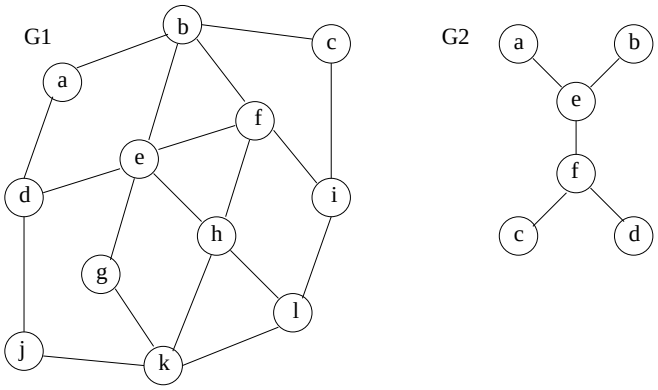

Figure 1: Graphs for examples

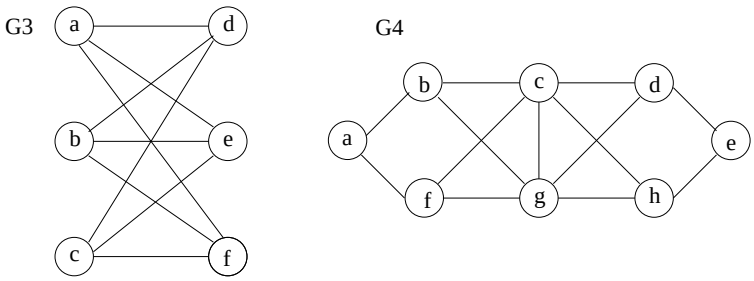

Figure 2: Graphs for examples

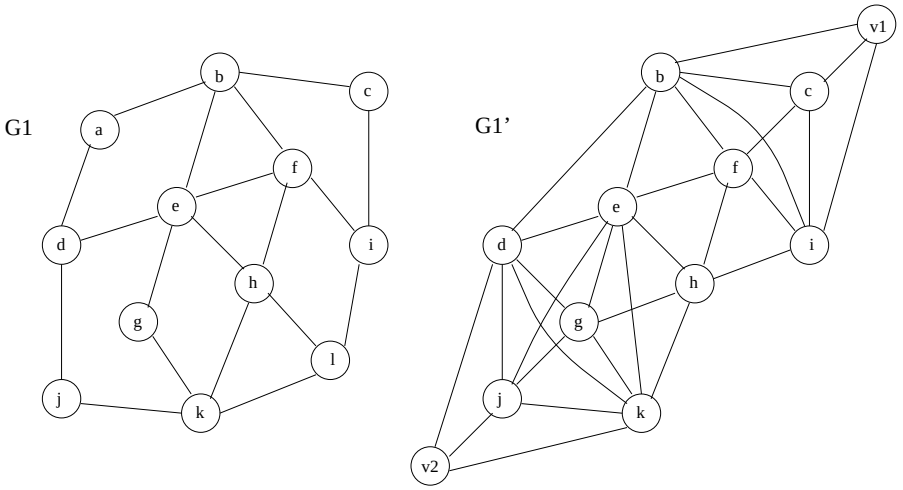

Figure 3: Finding a separator of two vertices in the same face. $S = \{a, l\}$.

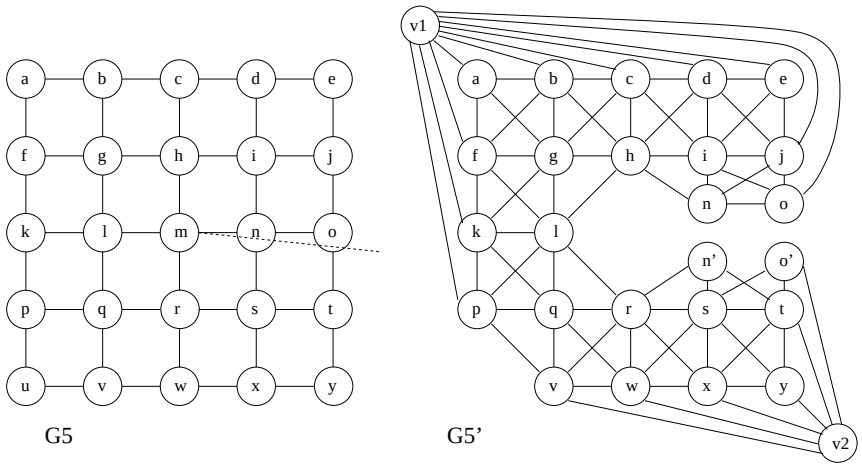

Figure 4: Finding a separator of two vertices not in the same face. $S = \{u, m\}$.

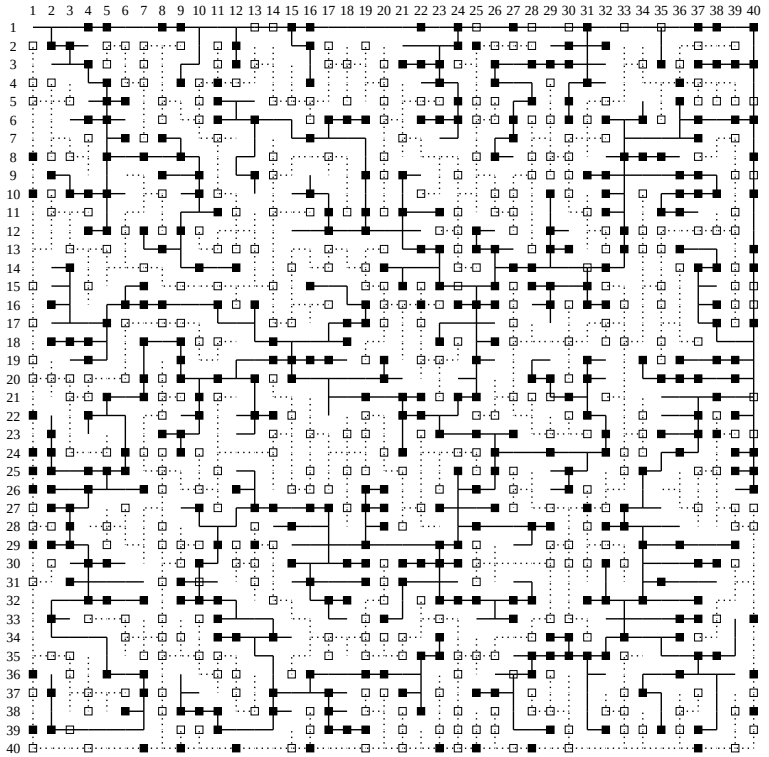

Figure 5: An optimal solution of instance u1600_1, $z^* = 46$.

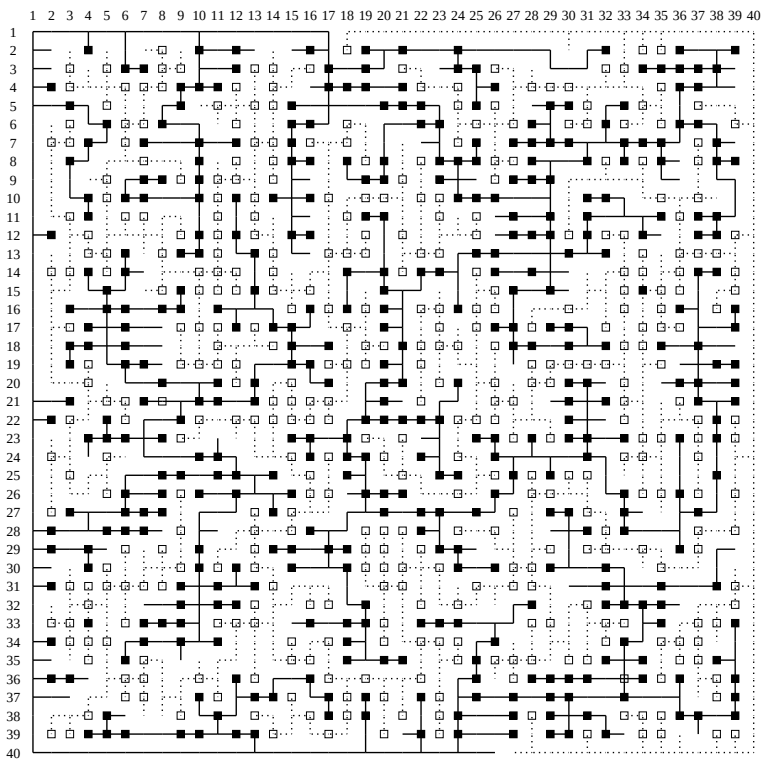

Figure 6: An optimal solution of instance t1600_1, $z^* = 4$.