STEINER TREES IN INDUSTRIES
D.-Z. Du and X. Cheng (Eds.) pp. – - –

Polyhedral Approaches for the Steiner Tree Problem on Graphs

Sunil Chopra
Kellogg Graduate School of Management
Northwestern University, Evanston, IL 60208
E-mail: s-chopra@kellogg.nwu.edu

Chih-Yang Tsai
School of Business Administration
The State University of New York at New Paltz, New Paltz, NY 12561
E-mail: tsaic@newpaltz.edu

Contents

1 Introduction

Given an undirected graph $G = (V, E)$ and a node set $T \subseteq V$, a *Steiner tree* for T in G is a set of edges $S \subseteq E$ such that the graph $(V(S), S)$ contains a path between every pair of nodes in T, where $V(S)$ is the set of nodes incident to the edges in S. Given costs (or weights) on edges and nodes, the *Steiner tree problem on a graph* (STP) is to find a minimum weight Steiner tree. The problem is known to be $\mathcal{NP}$-hard even for planar graphs, bipartite graphs, and grid graphs.

The importance of this problem in the design of electronic circuits as well as the design of telecommunication networks has led to a significant amount of research. A variety of solution methods studied include exact algorithms (see Hakimi [22] and Dreyfus and Wagner [12]), heuristic procedures (see Duin [13], Duin and Volgenant [15], [16], and Winter and Smith [38]), approximation methods(see Takahashi and Matsuyama [33], Zelikovsky [40], Karpinski and Zelikovsky [23]), polynomial algorithms for special instances (see Wald and Colbourn [35], [36]), polyhedral approaches (see Aneja [1], Chopra, Gorres, and Rao [8], Lucena and Beasley [26], Koch and Martin [24], and Suhl and Hilbert [31]), complete inequality descriptions for special instances (see Goemans [18], and Margot, Prodon, and Liebling [27]), Lagrangean relaxation approaches (see Beasley [4], [5], and Wong [39]), problem reduction procedures (see Balakrishnan and Patel [3], Duin and Volgenant [15], [16], and Winter [37]).

In this paper our objective is to survey various polyhedral approaches that have been used to study the problem. All polyhedral approaches formulate STP as an integer program and use linear programming relaxations of the problem to solve the integer program. Given that the problem is $\mathcal{NP}$-hard, it is unlikely that a complete inequality description of the integer polyhedron can be obtained. Thus, polyhedral approaches have been used to obtain three categories of results. First, the study of associated polyhe-

dra has been used to obtain solutions with a guaranteed maximum gap from optimality. This type of approach such as the one in [19] typically provides a polynomial algorithm to obtain a Steiner tree guaranteed to be within a specified gap from the optimal solution. Second, polyhedral approaches have also been used to study specific classes of graphs over which complete inequality descriptions of the integer polyhedron are obtained. This approach typically provides a polynomial time algorithm to obtain the optimal Steiner tree for a specific class of graphs (see [18] and [27] for example). Third, polyhedral approaches have been used to design cutting plane algorithms that have proved very effective in solving large STPs (see [8], [24] and [34]).

2 Integer Programming Formulations

In this section we describe several categories of integer programming formulations that have been used in the literature to model STP. Given the undirected graph G, formulations have been considered on G as well as the bi-directed graph $B = (V, A)$ obtained by replacing each undirected edge $e = (i, j)$ with two directed arcs (i, j) and (j, i). We consider each undirected edge e having a weight c_e. In the bi-directed graph, each arc replacing the edge e gets the weight c_e. Given an edge (i, j), its weight may also be represented as c_{ij} or $c_{(i,j)}$. Without loss of generality we assume that the node $1 \in T$. In the directed case our goal is to find a minimum weight *Steiner arborescence* rooted at the node 1 and containing a directed path from node 1 to every other terminal node in $T \setminus \{1\}$.

Given an undirected graph $G = (V, E)$, for a node set $X \subset V$, the edge set $\delta(X) = \{(i, j) : i \in X, j \in V \setminus X\}$ denotes the cut defined by the node set X. Given a directed graph $D = (V, A)$, for a set $X \subset V$, the arc set $\delta^-(X) = \{(i, j) \in A : i \in V \setminus X, j \in X\}$ denotes the directed cut defined by the arcs entering X. The arc set $\delta^+(X) = \{(i, j) \in A : i \in X, j \in V \setminus X\}$ denotes the directed cut defined by the set of arcs leaving X.

2.1 Multi-commodity Flow Based Formulations

In this section we present three multi-commodity flow based formulations. They are based on the idea that a Steiner tree will contain a path from the node 1 to every other node in $T \setminus \{1\}$. The formulations thus attempt to pick edges such that one unit of flow can be sent from the node 1 to every other node in $T \setminus \{1\}$. The formulations described here can be used when all edge weights are non-negative and there are no node weights. Multi-commodity

flow based formulations have typically been used for Lagrangean relaxation based solution methods (see Wong [39] and Balakrishnan et. al [2]).

For each node k in $T \setminus \{1\}$, define a commodity k. Given the undirected graph $G = (V, E)$ and the corresponding bi-directed graph $B = (V, A)$, for each arc $a = (i, j)$, the variable f_{ij}^k represents the flow of commodity k from node i to node j. The variable x_e defined for each edge e takes on the value 1 if edge e is in the Steiner tree and 0 otherwise. The natural multi-commodity flow (NMC) formulation for STP is as follows:

$$\text{Minimize} \sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{i \in V} f_{ij}^k - \sum_{i \in V} f_{ji}^k = \begin{cases} -1 & \text{if } j = 1 \\ 1 & \text{if } j = k \\ 0 & \text{otherwise for } j \in V \setminus \{1\} \end{cases} \tag{1}$$

$$f_{ij}^k \leq x_e \text{ for every edge } e = (i, j), \text{ commodity } k \tag{2}$$

$$f_{ij}^k \geq 0, x_e \text{ integer}$$

The equations (1) ensure one unit of flow from the node 1 to every other node in T, and if x_e is integral there exists a path from the node 1 to every other node in T resulting in a Steiner tree. In this formulation, there are $|E|$ capacity variables x_e, $|T||A|$ flow variables f_{ij}^k, $|V|$ flow conservation constraints (1), and $|T||A|$ flow capacity constraints (2). Define the polyhedron P^{NMC} to be the LP-relaxation of the NMC formulation where

$$P^{NMC} = \{(x, f) \geq 0 \text{ satisfying (1) and (2)}\}$$

The polyhedron P_x^{NMC} is the projection of P^{NMC} onto the space of x variables and is given by

$$P_x^{NMC} = \{x : (x, f) \in P^{NMC}\}$$

An extended multi-commodity flow (EMC) formulation can be obtained by considering capacities on the bi-directed graph $B = (V, A)$. Define variables y_a for each arc $a \in A$ where y_a takes integer values. STP can then be formulated as

$$\text{Minimize} \sum_{e \in E} c_e x_e$$

subject to:

$$x \text{ and } f \text{ satisfy (1)}$$

$$f_{ij}^k \leq y_{ij} \text{ for every arc } a = (i,j) \tag{3}$$

$$y_{ij} + y_{ji} = x_e \text{ for all edges } e = (i,j) \tag{4}$$

$$y \text{ integer}$$

Observe that the x variables in the EMC formulation can be eliminated using equations (4). The x variables are maintained so that we can compare the various formulations in terms of the projection of their LP-relaxations to the space of x variables. The EMC formulation was first proposed by Wong [39]. Define the polyhedron P^{EMC} to be the LP-relaxation of the EMC formulation where

$$P^{EMC} = \{(y, x, f) \geq 0 \text{ satisfying (1), (3), and (4)}\}$$

The polyhedron P_x^{EMC} is the projection of P^{EMC} onto the space of x variables and is given by

$$P_x^{EMC} = \{x : (y, x, f) \in P^{EMC}\}$$

Another flow based formulation, common-flow (CF) formulation, proposed by Polzin and Daneshmand [28] looks further into the flow shared by two commodities on the same arc. Common-flow (CF) formulation can be obtained by adding variables and constraints to the EMC formulation on the bi-directed graph $B = (V, A)$. Define variables $\hat{f}_{ij}^{kl}$ for each arc $(i,j) \in A$ and commodities $k, l \in T \setminus \{1\}$ as the common flow shared by commodities k and l on arc (i,j). The common-flow (CF) formulation can be written as

$$\text{Minimize} \sum_{e \in E} c_e x_e$$

subject to:

$$(x, y, f) \text{ satisfy (1), (3), and (4)}$$

$$\sum_{i \in V} \hat{f}_{ij}^{kl} - \sum_{i \in V} \hat{f}_{ji}^{kl} \geq \begin{cases} -1 & \text{if } j = 1, \text{ and } k, l \in T \setminus \{1\} \\ 0 & \text{if } j \in V \setminus \{1\}, \text{ and } k, l \in T \setminus \{1\} \end{cases} \tag{5}$$

$$\hat{f}_{ij}^{kl} \leq f_{ij}^{k} \text{ for all } k, l \in T \setminus \{1\} \text{ and } (i,j) \in A \tag{6}$$

$$\hat{f}_{ij}^{kl} \leq f_{ij}^{l} \text{ for all } k, l \in T \setminus \{1\} \text{ and } (i,j) \in A \tag{7}$$

$$f_{ij}^{k} + f_{ij}^{l} - \hat{f}_{ij}^{kl} \leq y_{ij} \text{ for all } k, l \in T \setminus \{1\} \text{ and } (i,j) \in A \tag{8}$$

$$\sum_{i \in V} y_{ij} \leq \sum_{i \in V} y_{ji} \text{ for } j \in V \setminus T \tag{9}$$

$$y \text{ integer}$$

The x variables in the CF formulation can be eliminated using equations (4) as before. Define the polyhedron P^{CF} to be the LP-relaxation of the CF formulation where

$$P^{CF} = \{(y, x, f, \hat{f}) \geq 0 \text{ satisfying (1), (3), (4), (5), (6), (7), (8), and (9) } \}$$

The polyhedron P_x^{CF} is the projection of P^{CF} onto the space of x variables and is given by

$$P_x^{CF} = \{x : (y, x, f, \hat{f}) \in P^{CF}\}$$

2.2 Cut Based Formulations

Cut based formulations for STP can be defined on both the undirected as well as the bi-directed graph. We discuss both formulations here. They are based on the idea that a Steiner tree contains a path from the node 1 to every other terminal node in $T \setminus \{1\}$. Thus every cut separating the node 1 from any other node in $T \setminus \{1\}$ must contain at least one edge from the Steiner tree in the undirected case or at least one arc from the Steiner arborescence in the directed case. The formulations presented here may be used if all edge weights are non-negative and there are no node weights. Cut based formulations have typically been used for cutting plane based solution approaches (see Chopra, Gorres, and Rao [8], Suhl and Hilbert [31], and Koch and Martin [24]).

For the undirected graph $G = (V, E)$ and terminal set T, given a node set $N \subset V$, the cut $\delta(N)$ is called a *Steiner cut* if $1 \in N$ and $|\{V \setminus N\} \cap T| \geq 1$. The undirected cut based (UCB) formulation for STP is given by

$$\text{Minimize } \sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{e \in \delta(N)} x_e \geq 1 \text{ for every Steiner cut } \delta(N) \text{ of } G \tag{10}$$

$$x \text{ integer for all } e \in E$$

The UCB formulation was first considered by Aneja [1]. Define the polyhedron P^{UCB} to be the LP-relaxation of the UCB formulation where

$$P^{UCB} = \{x \geq 0 \text{ satisfying } (10)\}$$

The directed cut based (DCB) formulation for STP can be obtained on the bi-directed graph $B = (V, A)$. Given $N \subset V$, the directed cut $\delta^+(N)$ is called a *directed Steiner cut* if $1 \in N$ and $|\{V \setminus N\} \cap T| \geq 1$. The directed cut based (DCB) formulation for STP is given by

$$\text{Minimize } \sum_{e \in E} c_e x_e$$

subject to:

$$\sum_{a \in \delta^+(N)} y_a \geq 1 \text{ for every directed Steiner cut } \delta^+(N) \text{ of } B \tag{11}$$

$$y_{ij} + y_{ji} = x_e \text{ for all edges } e = (i, j) \tag{12}$$

$$y \text{ integer for all } e \in E$$

The DCB formulation was first considered by Chopra and Rao [9]. Observe that we can eliminate the x variables from the DCB formulation using equations (12). Define the polyhedron P^{DCB} to be the LP-relaxation of the DCB formulation where

$$P^{DCB} = \{(y, x) \geq 0 \text{ satisfying } (11), (12)\}$$

The polyhedron P_x^{DCB} is the projection of P^{DCB} onto the space of x variables and is given by

$$P_x^{DCB} = \{x : (y, x) \in P^{DCB}\}$$

2.3 Node Variable Based Formulations

In node variable based formulations we also include node variables z_i for nodes in $V \backslash T$. The variable z_i takes on the value 1 if the node i is spanned by the Steiner tree and 0 otherwise. Observe that all nodes in the terminal set T are spanned by the Steiner tree and for this reason we do not consider node variables associated with them. We consider node weights w_i for each node $i \in V \setminus T$ along with edge weights c_e. Node variable based formulations can be used to find minimum weight Steiner trees when the graph has both edge as well as node weights. By setting all node weights to 0, these formulations can also be used when there are no node weights.

The first formulation we present is based on the idea that a tree can have at most k-1 edges from the set of edges spanned by a node set of size k. Such a constraint is referred to as a *subtour elimination* constraint. Given a node set $X \subseteq V$ define $E(X) = \{e = (i,j) \in E : i \in X \text{ and } j \in X\}$ to be the edges in E with both end nodes in X. The subtour elimination (SE) formulation for STP is as follows:

$$\text{Minimize } \sum_{e \in E} c_e x_e + \sum_{i \in V \backslash T} w_i z_i$$

subject to:

$$\sum_{e \in E} x_e = \sum_{i \in V \backslash T} z_i + |T| - 1 \tag{13}$$

$$\sum_{e \in E(X)} x_e \leq \sum_{i \in X \backslash T} z_i + |X \cap T| - 1 \text{ for } X \subset V, |X \cap T| \geq 1 \tag{14}$$

$$\sum_{e \in E(X)} x_e \leq \sum_{i \in X \backslash \{k\}} z_i \text{ for } k \in X \subseteq V \setminus T \tag{15}$$

$$z_i \leq 1 \text{ for } i \in V \setminus T \tag{16}$$

$$x_e \geq 0 \text{ and integer for } e \in E, z_i \text{ integer for } i \in V \setminus T$$

The SE formulation was proposed by Goemans [18] and a related formulation was considered by Margot, Prodon, and Liebling [27]. The SE formulation can be used for all edge and node weights. Define the polyhedron P^{SE} to be the LP-relaxation of the SE formulation where

$$P^{SE} = \{(x,z) \geq 0 \text{ satisfying } (13), (14), (15), \text{ and } (16)\}$$

The polyhedron P^{SE}_x is the projection of P^{SE} on to the space of x variables and is given by

$$P_x^{SE} = \{x : (x, z) \in P^{SE}\}$$

The cut based formulation DCB can be extended to include node variables in the directed cut based node variable (DCBN) formulation shown here for the bi-directed graph $B = (V, A)$. The variables x, y, and z are as defined earlier. The formulation as shown is applicable for the case when all edge weights are non-negative. Any problem on a directed graph, however, can be modified to the case when all arc weights are non-negative. The modification does not result in arcs (i, j) and (j, i) having the same weight.

$$\text{Minimize } \sum_{e \in E} c_e x_e + \sum_{i \in V \setminus T} w_i z_i$$

Subject to:

$$(y, x, z) \text{ satisfies (11) and (12)}$$

$$\sum_{a \in \delta^+(N)} y_a \geq z_i \text{ for } i \in V \setminus N, T \subseteq N \subset V \tag{17}$$

$$(y, x, z) \geq 0, y, z \text{ integer}$$

Observe that the x variables can be eliminated from the formulation using equations (12). The x variables are maintained to be able to compare the projections of the LP-relaxations of the various formulations on to the space of x variables. The DCBN formulation was first proposed by Chopra and Gorres [21]. The polyhedron P^{DCBN} is defined by the LP-relaxation of the DCBN formulation where

$$P^{DCBN} = \{(y, x, z) \geq 0, z_i \leq 1 \text{ for } i \in V \setminus T \text{ satisfying (11), (12), (17)}\}$$

The polyhedron P_x^{DCBN} is the projection of P^{DCBN} on to the space of x variables and is given by

$$P_x^{DCBN} = \{x : (y, x, z) \in P^{DCBN}\}$$

3 A Comparison of LP-relaxations

In this section we compare the various LP-relaxations of the formulations for STP considered in Section 2. Some of the LP-relaxations have been compared previously in [9], [20], and [28]. We will first consider a comparison

of all LP-relaxations when there are no node weights. We will also compare the LP-relaxations of the SE and DCBN formulations in case node weights are present.

The optimal solution to the LP-relaxation of any of the formulations in Section 2 provides a lower bound for the integer optimum. For a given STP, we can compare the strengths of the various LP-relaxations in terms of the lower bound they provide for the integer optimum. The higher the lower bound provided, the stronger the LP-relaxation.

First consider STP in an undirected graph $G = (V, E)$ with edge weights $c_e \geq 0$ for all edges $e \in E$ and no node weights. All formulations discussed in Section 2 are applicable in this case. We compare the projections of the LP-relaxations of the various formulations on to the space of x variables. Define

$$Z^{NMC} = \text{Min } \{\sum_{e \in E} c_e x_e | x \in P_x^{NMC}\}$$

$$Z^{EMC} = \text{Min } \{\sum_{e \in E} c_e x_e | x \in P_x^{EMC}\}$$

$$Z^{CF} = \text{Min } \{\sum_{e \in E} c_e x_e | x \in P_x^{CF}\}$$

$$Z^{UCB} = \text{Min } \{\sum_{e \in E} c_e x_e | x \in P^{UCB}\}$$

$$Z^{DCB} = \text{Min } \{\sum_{e \in E} c_e x_e | x \in P_x^{DCB}\}$$

$$Z^{SE} = \text{Min } \{\sum_{e \in E} c_e x_e | x \in P_x^{SE}\}$$

$$Z^{DCBN} = \text{Min } \{\sum_{e \in E} c_e x_e | x \in P_x^{DCBN}\}$$

The results are stated here without proof. The proofs are either given in ([9]), ([20]), and ([28]) or they can be obtained using similar proof techniques.

Theorem 3.1 *The LP-relaxations to the NMC and UCB formulations provide the same lower bound for STP, i.e., $Z^{NMC} = Z^{UCB}$.*

From Theorem 3.1 it follows that the formulations NMC and UCB provide equally strong LP-relaxations. Neither LP-relaxation, however, is particularly strong. Consider the graph $G_6 = (V_6, E_6)$ on six nodes where $V_6 =$

$\{1,2,3,4,5,6\}$, $E_6 = \{(1,2),(1,6),(2,3),(2,4),(2,6),(3,4),(4,5),(4,6),(5,6)\}$ and $T = \{1,3,5\}$. If all edge weights are 1, the integer optimum is 4 and the optimal Steiner tree consists of the edges (1,2), (2,3), (3,4), and (4,5). Using the LP-relaxations to NMC and UCB we obtain $Z^{NMC} = Z^{UCB} = 3$ which is less than 4. The optimal solution to the LP-relaxation has $x_e = 0.5$ for $e \in \{$(1,2), (1,6), (2,3), (3,4), (4,5), (5,6)$\}$ and $x_e = 0$ for all other edges. Thus, even on this simple graph, the two LP-relaxations provide solutions that are far from optimal. Chopra and Rao [9], [10] and Goemans [18] provide families of facet defining inequalities that strengthen the LP-relaxation to NMC and UCB.

Theorem 3.2 *The LP-relaxations to the EMC, DCB, SE, and DCBN formulations provide the same lower bound for STP, i.e.,* $Z^{EMC} = Z^{DCB} = Z^{SE} = Z^{DCBN}$.

From Theorem 3.2 it follows that the formulations EMC, DCB, SE, and DCBN provide equally strong LP-relaxations when solving STP without node weights. These LP-relaxations provide better lower bounds than the LP-relaxations to NMC and UCB.

Theorem 3.3 *The LP-relaxations to the EMC, DCB, SE, and DCBN formulations provide at least as strong a lower bound as the LP-relaxations for the NMC and UCB formulations, i.e.,* $Z^{EMC} = Z^{DCB} = Z^{SE} = Z^{DCBN} \geq Z^{NMC} = Z^{UCB}$.

The LP-relaxations to EMC, DCB, SE, and DCBN formulations generally provide better lower bounds than the LP-relaxations to NMC and UCB. For example, on the graph G_6 discussed earlier, all four LP-relaxations provide the integer optimum. Thus, from a practical perspective, the EMC, DCB, SE, and DCBN formulations are likely to be much more effective than either the NMC or the UCB formulation. This fact is verified by computational results in Chopra, Gorres, and Rao [8]. In the sample of 459 problems considered by them, the LP-relaxation of DCB resulted in the integer optimum for all but 12 problems. When UCB and DCB were compared, the LP-relaxation of DCB resulted in the optimal solution for each problem while the LP-relaxation of UCB resulted in an average gap of 17% from optimality. These findings were confirmed by Koch and Martin [24] who did an even more extensive computational study using the DCB formulation.

The strength of LP-relaxations can also be compared in terms of the gap of the optimal solution to the LP-relaxation from the integer optimum in the worst case. Goemans and Bertsimas [19] show the following result.

Theorem 3.4 *The optimum, Z^{UCB}, to the LP-relaxation of UCB is within a factor of $1/(2-2/|T|)$ of the integer optimum, if all edge costs satisfy the triangle inequality.*

Theorem 3.3 and Theorem 3.4 together show that if edge costs satisfy the triangle inequality, optimum solutions to any of the LP-relaxations considered here are guaranteed to be within a factor of $1/(2-2/|T|)$ of the integer optimum. The factor approaches $\frac{1}{2}$ when $|T|$ is large. In practice, however, the gaps from optimality tend to be much smaller than the worst-case guarantee.

The strength of the EMC, DCB, SE, and DCBN formulations is also shown by the following result proved by Goemans [18] and Margot, Prodon, and Liebling [27].

Theorem 3.5 *If G is a series-parallel graph, the polyhedron P^{SE} has integer extreme points.*

As a result, for series-parallel graphs, the optimum solution to the LP-relaxations Z^{EMC}, Z^{DCB}, Z^{SE} and Z^{DCBN} has the same value as the integer optimum.

Because the CF formulation uses a great number of additional variables to define the flow on arcs, its LP-relaxation produces better lower bounds than those obtained from the LP-relaxation of formulation EMC [28]. This result and Theorem 3.3 give us the following conclusion.

Theorem 3.6 *The LP-relaxations to the CF formulation is at least as strong a lower bound as the LP-relaxations for the EMC, DCB, SE, and DCBN formulations, i.e., $Z^{CF} \geq Z^{EMC} = Z^{DCB} = Z^{SE} = Z^{DCBN} \geq Z^{NMC} = Z^{UCB}$.*

Although the LP-relaxation of the CF formulation provides a better lower bound, it has many more variables and constraints compared to the EMC formulation. There is no computational study that has tested the effectiveness of the CF formulation.

In the presence of node weights it can be shown that the LP relaxations to the SE and DCBN formulations provide the same lower bound.

4 Strengthening LP-relaxations Using Facets

When an LP-relaxation provides an incomplete inequality description of the integer polyhedron, the optimal LP solution often tends to be fractional. Any polyhedral approach attempts to strengthen the LP-relaxation by identifying inequalities that are valid for the integer polyhedron but are violated by the current fractional solution. Addition of such inequalities allows the LP-relaxation to be strengthened. *Facet defining inequalities* are the strongest inequalities that can be added to the LP-relaxation. The convex hull of integer points satisfying a facet defining inequality at equality has dimension one less than the dimension of the integer polyhedron. In this section we discuss how the various LP-relaxations discussed in Section 2 can be strengthened with the addition of facet defining inequalities.

Given that the NMC and UCB formulations do not provide very strong LP-relaxations they have not been used in practice to solve large instances. Facet defining inequalities that strengthen the two formulations are discussed in Chopra and Rao [10], Goemans [18] and Balakrishnan, Magnanti, and Wong [2]. Here we introduce two classes of facet defining inequalities for the DCB and DCBN formulations.

4.1 Facet Defining Inequalities for the DCB Formulation

Given a directed graph $D = (V, A)$ and a set of terminals T with the root node $1 \in T$, define $IP^{DCB}(D, T)$ to be the convex hull of all integer points in the polyhedron $P^{DCB}(D, T)$. The extreme points of $IP^{DCB}(D, T)$ are thus incidence vectors of Steiner arborescences in D. Our goal is to identify facet defining inequalities for $IP^{DCB}(D, T)$. The results in this section are discussed in detail in [10]. All facet defining inequalities discussed here can be lifted to facet defining inequalities for larger graphs using lifting procedures discussed in [9] and [10].

We first define *odd wheel inequalities*. Consider the graph $DW_k = (V_k, A)$ where $T_k = \{1, t_j, j = 1, 2, \ldots, k\}$ with node 1 as the root, $V_k = T_k \cup \{v_j, j = 1, 2, \ldots, k\}$, $A = \{(i, j) | i, j \in V_k\}$ and $A_1 = \{(1, v_j), j = 1, 2, \ldots, k\} \cup \{(t_j, v_j), (t_j, v_{j-1}), j = 1, 2, \ldots, k\}$, $A_2 = \{(v_j, 1), j = 1, 2, \ldots, k\} \cup \{(v_j, t_j), (v_{j-1}, t_j), j = 1, 2, \ldots, k\}$. All indices are defined modulo k with $0 = k$. Define $A_3 = \{(t_j, 1), j = 1, 2, \ldots, k\}$. T_k is the set of terminals that must be spanned by the Steiner tree.

Theorem 4.1 *The odd wheel inequality*

$$\sum_{a \in A_1 \cup \{A \setminus \{A_1 \cup A_2 \cup A_3\}\}} y_a \geq (k+1)/2 \tag{18}$$

is facet defining for $IP^{DCB}(DW_k, T_k)$ *for* $k \geq 3$ *and odd.*

Next we define *bipartite inequalities.* Consider the graph $DB_k = (V_k, A)$ with node t_{k-1} as the root where $T_k = \{t_j, j = 1, 2, \ldots, k\}$, $V_k = T_k \cup \{v_j, j = 1, 2, \ldots, k-1\}$, $A = \{(i,j) | i, j \in V_k\}$ with $A_1 = \{(t_j, v_i), i = 1, 2, \ldots, k, j \neq i - 1(\bmod k)\}$, $A_2 = \{(v_i, t_j), i = 1, 2, \ldots, k, j \neq i - 1(\bmod k)\}$, and A_3 is the set of all arcs entering t_{k-1}. All indices are defined modulo k with $0 = k$. T_k is the set of terminals that must be spanned by the Steiner tree.

Theorem 4.2 *The bipartite inequality*

$$\sum_{a \in A_1 \cup \{A \setminus \{A_1 \cup A_2 \cup A_3\}\}} y_a \geq 2 \tag{19}$$

is facet defining for $IP^{DCB}(DB_k, T_k)$ *for* $k \geq 4$.

Other facet defining inequalities for $IP^{DCB}(D, T)$ are discussed in [10].

4.2 Facet Defining Inequalities for the DCBN Formulation

Given a directed graph $D = (V, A)$ and a set of terminals T with the root node $1 \in T$, define $IP^{DCBN}(D, T)$ to be the convex hull of all integer points in the polyhedron $P^{DCBN}(D, T)$. Several families of facet defining inequalities are described by Gorres [21]. Here we only describe two that are related to the facet defining inequalities for $IP^{DCB}(D, T)$.

The *node variable odd wheel inequality* is defined for graphs $DW_k = (V_k, A)$ discussed earlier. The sets T_k, A_1, A_2, and A_3 are as defined for the odd wheel inequalities. The terminals to be spanned by the Steiner tree are represented by $\bar{T} \subseteq T_k$. We assume that the root node $1 \in \bar{T}$. In this case we have node variables z along with the arc variables y.

Theorem 4.3 *The node variable odd wheel inequality*

$$\sum_{v_j \in T_k \setminus \bar{T}} z(v_j) - \sum_{a \in A_1 \cup \{A \setminus \{A_1 \cup A_2 \cup A_3\}\}} y_a \leq (k-1)/2 + |\bar{T}| \tag{20}$$

is facet defining for $IP^{DCBN}(DW_k, \bar{T})$ *for* $k \geq 3$ *and odd.*

The *node variable bipartite inequality* is defined for graphs $DB_k = (V_k, A)$ discussed earlier. The sets T_k, A_1, A_2, and A_3 are as defined for the odd wheel inequalities. The terminals to be spanned by the Steiner tree are represented by $\bar{T} \subseteq T_k$. We assume that the root node $t_{k-1} \in \bar{T}$. In this case we have node variables z along with the arc variables y.

Theorem 4.4 *The node variable bipartite inequality*

$$\sum_{v_j \in T_k \setminus \bar{T}} z(v_j) - \sum_{a \in A_1 \cup \{A \setminus \{A_1 \cup A_2 \cup A_3\}\}} y_a \leq (k-2) - |\bar{T}| \qquad (21)$$

is facet defining for $IP^{DCBN}(DB_k, \bar{T})$ *for* $k \geq 4$.

5 Branch-and-cut Approach for Solving STP on Graphs

In this section we present a branch-and-cut approach that has proved very effective at solving STP on graphs when there are no node weights and all edge weights are non-negative (see [34] and [24] for example). Many network and VLSI design problems in practice can be modeled as having non-negative edge weights and no node weights ([34], [11]). When node weights are present, this approach with some modifications also shows great success in solving large instances (see [7] and [21]). Those modifications are briefly mentioned at the end of this section.

The approach detailed here applies to an undirected graph but can easily be modified for directed graphs. Given the undirected graph $G = (V, E)$ with terminals T and edge weights $c_e \geq 0$ for $e \in E$, the branch-and-cut procedure can be summarized as the following steps:

1. Preprocessing (Graph Reduction)
2. Initialization
3. select a leaf from the branch-and-cut tree and consider associated LP
4. solve the LP
5. identify violated inequalities (cuts) and augment LP. Eliminate inequalities that are not tight for the current solution. If some violated inequalities are found go to step 3

6. run heuristic to obtain a Steiner tree

7. branch if optimal solution is fractional and a gap exists from the best heuristic solution, else remove the leaf from tree

8. If branch-and-cut tree is empty STOP, else go to step 2.

Each LP is solved using the dual simplex method because the addition of violated inequalities results in an LP for which the existing solution is not primal feasible but is dual feasible. A few critical success factors to a branch-and-cut approach are highlighted next. More detailed discussions can be found in [8] and [24].

5.1 Preprocessing STP on a Graph

Preprocessing plays a very important role when solving STPs in practice. In most instances, preprocessing reduces problem size by a significant amount making it feasible to solve large problems. Preprocessing should be used no matter what the solution procedure is. Preprocessing, however, is particularly effective when using cutting plane based approaches because it reduces the size of each LP relaxation. It is applied to the undirected graph G and the goal is to identify edges that must be in some optimal solution as well as edges that cannot be in any optimal solution. The edges in some optimal solution are contracted and the edges in no optimal solution are deleted to get an equivalent problem on a smaller graph.

The preprocessing algorithms discussed here are from [3], [15], [16], [8], [13], and [24]. Refer to [34] and [29] for more preprocessing algorithms.

5.1.1 Degree Tests for STP on Graphs

If node $j \in T$ has degree one, the edge adjacent to j is always part of an optimal Steiner tree and can be contracted. The resulting node belongs to the new set of terminals T'.

If node $j \in V \setminus T$ has degree one, the edge adjacent to j can be deleted because edge weights are non-negative and this edge cannot be part of an optimal Steiner tree.

If node $j \in V \setminus T$ has degree two, the two edges (j, i) and (j, k) adjacent to it can be replaced by the edge (i, k) with weight $c_{ik} = c_{ji} + c_{jk}$.

5.1.2 Terminal Distance Test

This test was introduced by Duin and Volgenant [16] and is also discussed in [24]. Consider a node set W with $|W \cap T| \geq 1$ and $|\{V \setminus W\} \cap T| \geq 1$, that induces a connected subgraph $H = (W, F)$ of G. Let $e = (u, v)$ with $u \in W$ be the edge with the lowest weight and f the edge with the second lowest weight in the cut $\delta(W)$. Define the shortest path from a node u to a set of nodes X to be the minimum of the shortest paths from u to each node in X. Let d_u be the length of the shortest path from the node u to the set $W \cap T$ and d_v the length of the shortest path from the node v to the set $\{V \setminus W\} \cap T$. The edge e can be contracted if

$$c_f \geq d_u + d_v + c_e$$

Duin [13] provides an $O(V^2)$ approach to obtaining all edges that can be contracted using the terminal distance test.

In practice, however, we often perform only special cases of this test to save time. The minimum spanning tree based $R-R$ Edge Deletion described in [3] is one example. Other test includes the special case when $\{u, v\} \subseteq T$ and one of the nodes is the nearest neighbor of the other. It also includes the special case where $W = \{j\}$, $j \in T$, u is the nearest neighbor of j, and $(u, v) \in E$ where $v \in T$. Let i be the second closest neighbor of j. The edge (j, u) can be contracted if

$$c_{uv} + c_{uj} \leq c_{ji}$$

5.1.3 Special Distance Test

This test was introduced by Duin[13] and is also discussed in [16]. Consider any edge $e = (u, v) \in E$. Consider any path P from u to v in G not using the edge e and containing at least one node from T. The path can be divided into segments with end points belonging to $T \cup \{u, v\}$. Define $d(P)$ to be the length of the largest such segment in the path P. Also define

$s(u, v) = \min \{d(P) : P \text{ is a path connecting } u \text{ and } v \text{ not containing edge } e\}$

$s(u, v)$ has been called the *special distance* by Duin. It is easy to verify that the edge (u, v) cannot be in an optimal Steiner tree and can be deleted if

$$s(u, v) \leq c_{uv}$$

The special case where we only consider paths in the graph induced by the nodes $T \cup \{u, v\}$ can be checked faster and is reasonably effective. In fact

it is very fast to restrict attention to this graph and only consider paths of length two or three. When considering paths of length two we consider any node $j \in T$ and any two of its neighbors u and v. If

$$\max \{c_{uj}, c_{vj}\} \leq c_{uv}$$

the edge (u, v) can be deleted. When considering paths of length three we consider any edge (i, j) where $\{i, j\} \subseteq T$, and two adjacent edges (u, i) and (v, j). The edge (u, v) can be deleted if

$$\max \{c_{ui}, c_{ij}, c_{vj}\} \leq c_{uv}$$

These graph reduction steps can be performed repeatedly until no further reduction is achieved. They work efficiently on general graphs but do not work as well in grid graphs which exist in many VLSI design problems. Uchoa, Aragão and Ribeiro [34] enhance the graph reduction algorithms discussed in this section for grid graphs and show significant improvement in performance.

5.2 Initializing to Solve STP

Given the undirected graph $G = (V, E)$ with terminals T obtained at the end of preprocessing, construct the bi-directed graph $B = (V, A)$ where each undirected edge (i, j) is replaced by two directed arcs (i, j) and (j, i). Each arc gets the same weight c_e as the undirected edge. We introduce the directed cut based (DCB) formulation discussed earlier on the bi-directed graph. Without loss of generality assume that the node $1 \in T$. Define arc variables y_a as described earlier. We do not include the edge variables x_e in the formulation because their values can be obtained given the value of the y variables. The formulation used in the branch-and-cut procedure is as follows.

$$\text{Minimize } \sum_{a \in A} c_a y_a$$

subject to:

$$\sum_{a \in \delta^+(N)} y_a \geq 1 \text{ for every directed Steiner cut } \delta^+(N) \text{ of } B \qquad (22)$$

$$y_a \text{ integer for all } a \in A$$

Since there is an exponential number of directed Steiner cuts in the formulation, a small subset of those cuts is used for the initial formation and additional cuts are added and dropped in each subsequent iteration as needed. We set up the first LP and initialize the branch-and-cut tree with the root representing the whole problem. The starting LP consists of the directed Steiner cut inequalities (22) for $N = V \setminus \{j\}$ for each node $j \in T \setminus \{1\}$. For grid graphs, it helps to also include all cuts defined by each row or column of the grid.

5.3 Identifying Violated Inequalities for STP

Separation for Steiner cut inequalities (22) is performed using a max-flow algorithm where the capacity on each arc a equals the value of y_a in the current solution. If the minimum cut separating the root node 1 from any other terminal node in $T \setminus \{1\}$ has capacity less than 1, we have a violated Steiner cut inequality.

While the basic separation problem is easy to solve, ideas of Chopra et al. [8] and Koch and Martin [24] are extremely important in speeding up the solution time. The three basic ideas have been referred to as *back cuts*, *nested cuts*, and *creep flow*.

The idea of back cuts, proposed in [8], increases the number of cuts generated at each iteration by reversing the direction of all arcs and considering the minimum cut separating any node j from the root node 1 rather than the other way around. This approach provides many distinct cuts that are added to the LP at each iteration and significantly speeds up the overall solution time. In addition to the idea of back cut, the use of breadth-first-search (BFS) in place of the max-flow algorithm at the stage when the induced graph contains disconnected components could also improve solution times because BFS has lower order of complexity compared to the max-flow algorithm. BFS can be executed on both original and reverse graphs. A switch can be built to activate the appropriate algorithm depending on the connectivity status of the graph [11].

The idea of nested cuts, proposed in [24], finds additional violated cuts between a node j and the root node 1 by setting all arcs in previously identified cuts to 1. This procedure also increases the time spent in identifying violated inequalities but decreases the total time.

The idea of creep flow, proposed in [24], is to identify minimum weight cuts that are also minimal in terms of the number of arcs included in the cut. This is done by adding a tiny capacity (for example 10^{-6}) to each arc

when finding the minimum weight cuts.

Of the three ideas, back cuts play the most significant role in speeding up solution times.

5.4 Primal Heuristics for STP

During our branch-and-cut procedure, we obtain fractional LP solutions that can provide valuable information for constructing good heuristic solutions especially when the fractional solution is already very close to the optimal solution. Good heuristic solutions not only provide upper bounds to an optimal solution but also curtail the branching tree to improve solution times. We introduce three primal heuristics that make use of the current fractional solutions.

The first primal heuristic is based on the fact that if we know the set of Steiner nodes $T' \subseteq V \setminus T$ to be included in an optimal Steiner tree, STP can be solved as a minimum spanning tree problem on the graph induced by $T \cup T'$. Given a fractional solution y, define a node value r_j for each node $j \in V \setminus T$, where

$$r_j = \sum_{a \in \delta^-(\{j\})} y_a$$

r_j is the sum of all arc variables entering the node j. Consider the undirected graph $G_y = (V_y, E_y)$, where E_y consists of edges (i, j) for which at least one of y_{ij} or y_{ji} is positive, and V_y consists of all nodes spanned by the edges in E_y. Let $T' = \{j \in V \setminus T | r_j = 1\}$. The heuristic then proceeds as follows:

1. Set $N = T \cup T'$
2. If the subgraph of G_y induced by nodes in N is connected, go to step 4 else go to step 3.
3. For each node $j \in V \setminus N$, generate a random number n_j, $0 \leq n_j \leq 1$. Add all nodes j for which $n_j \leq r_j$ to the set N. Go to step 2.
4. Find the minimum spanning tree on the subgraph of G_y induced by the nodes in N. This is a Steiner tree because $T \subseteq N$. Prune the tree until all leaf nodes belong to T.

The heuristic is applied multiple times for the same fractional solution and the Steiner tree with the lowest weight is selected as the heuristic solution. The old heuristic solution is updated only if the new one has lower weight.

The second heuristic applies Takahashi and Matsuyama's heuristic [33] on the bi-directed graph B whose arc weight, c'_a is dictated by the current LP solution, for example, $c'_a = (1 - y_a)c_a$ in [24]. Let $P(N, j)$ be the shortest path between a node set $N \subset V$ and a single node $j \in V$, $j \notin N$ using the edge weight c'_a. The procudure goes as follows.

1. Initialization: $N = \{1\}$ and $C = 0$.
2. if $T \subseteq N$ Stop. Else go to the next step.
3. Search: find $P(N, \bar{j}) = min\{P(N, j) | j \in T \text{ and } j \notin N\}$.
4. Update: $N = N \cup J$ and $C = C + \text{length}(P(N, \bar{j}))$ where J is the set of nodes on $P(N, \bar{j})$. Go to Step 2.

The heuristic solution has total weight C and contains all arcs on $P(N, \bar{j})$ chosen in all iterations.

The third heuristic applies Takahashi and Matsuyama's heuristic [33] on the subgraph induced by arcs with $y_a > 0$ in the fractional solution using the edge weight c_a [8].

5.5 Modifications for the Node Weighted STP

The branch-and-cut procedure used for the node weighted STP has the same steps as the procedure for STP without node weights. The preprocessing steps and heuristic discussed earlier for STP without node weights can be modified easily to accommodate node weights. Thus we only describe the initialization phase and identification of violated inequalities. We use the directed cut based node variable formulation (DCBN) discussed earlier on the bi-directed graph B created from the preprocessed undirected graph G. The goal is to find a minimum weight (including node weights) Steiner arborescence rooted at 1 that spans all nodes in T. We define arc variables y_a for each arc $a \in A$ and node variables z_j for each node $j \in V \setminus T$ as described earlier. For the same reason stated early, edge variables x_e do not need to be included. The formulation used in the branch-and-cut procedure is as follows:

$$\text{Minimize} \sum_{a \in A} c_a y_a + \sum_{j \in V \setminus T} w_j z_j$$

subject to:

$$\sum_{a \in \delta^+(N)} y_a \geq 1 \text{ for every directed Steiner cut } \delta^+(N) \text{ of } B \qquad (23)$$

$$\sum_{a \in \delta^+(N)} y_a \geq z_i \text{ for } i \in V \setminus N, T \subseteq N \subset V \tag{24}$$

$$(y, z) \geq 0, z_j \leq 1 \text{ and integer for all } j \in V$$

Initially we set up the first LP and initialize the branch-and-cut tree with the root node representing the whole problem. The starting LP consists of the directed Steiner cut inequalities (23) for $N = V \setminus \{j\}$ for each node $j \in T \setminus \{1\}$, the inequalities (24) for $N = V \setminus \{j\}$ for each node $j \in V \setminus T$. Each node variable z_j is assigned an upper bound of 1.

Separation for both the Steiner cut inequalities (23) and the inequalities (24) can be performed in polynomial time using any polynomial max-flow algorithm on a graph where the capacity of each arc equals the value of the corresponding variable in the LP-solution. If the minimum weight cut separating the root node 1 from any other terminal node in $T \setminus \{1\}$ is less than 1, a violated Steiner cut inequality (23) is obtained where N corresponds to the shore of the minimum weight cut containing the root node 1. For each node $j \in V \setminus T$, if the minimum cut separating the root node 1 from the node j has capacity less than z_j, the value of the corresponding node variable in the LP solution, we have a violated inequality (24). Computational studies applying the branch-and-cut approach on the node weighted STP can be found in ([7]) and ([21]). Solving node weighted STP using Lagrangean relaxation approach can be found in ([30]) and ([17]).

The ideas of back cuts, BFS cuts, nested cuts, and creep flow discussed earlier can be implemented for both the inequalities (23) and (24). As mentioned earlier, the use of back cuts has a significant impact on total solution times.

5.6 Computational Results for STP

In this section we report some computational results in the Steiner tree literature based on a branch-and-cut approach. For comparability reason, we only pick those studies that solve STP instances from two benchmark libraries, OR-Library [6] and SteinLib [25]. The instances in both libraries are STP without node weights.

Table 1 lists the studies that solve three subsets of OR-Library STP instances each containing 20 instances. The first study [5] applies a Lagrangean relaxation approach instead of a branch-and-cut approach. We

source	[5]	[8]	[26]	[24]	[29]
Formulation	-	DCB	RSST*	DCB	DCB**
machine	X-MP/48 Cray	VAX 8700 Digital	Indigo Silicon Graphic	SPARC 20 Sun	Pentium-II Intel
Set C	20	20	20	20	20
Set D	20	20	20	20	20
Set E	14	19	16	20	20

* Restricted Shortest Spanning Tree formulation (a node based formulation not discussed in this review)

** This study applies a synthesized approach on DCB formulation.

Table 1: Computational Results on OR-Library

citation	[24]	[34]	[29]
Formulation	DCB	DCB	DCB*
machine/CPU	SPARC 20 Sun	ULTRA Sun	Pentium-II Intel
VLSI instances			
alue (15)	7	13	13
alut (9)	3	7	7
diw (21)	14	21	21
dmxa (14)	13	14	14
gap (13)	12	13	13
msm (30)	25	30	30
taq (14)	10	14	14

* This study applies a synthesized approach on DCB formulation.

Table 2: Computational Results on OR-Library

include it because it is the first study using the instances in OR-Library. Numbers in the table shows the number of instances in each category solved to optimality by each study. We do not report computational times because those studies were conducted on different computers. However, the result reported in Polzin and Daneshmand [29] is by far the best result in terms of both number of instances solved to optimality and computational times factoring into the difference in cpu speed. Their approach is a synthesized approach based on the DCB formulation. It combines various approaches, including dual ascent, cut generation, Lagrangean relaxation, and heuristics. The synthesized approach is very effective in graph reduction. In fact, most of the instances reported here are solved to optimality by the reductions. A comparison of the performance of graph reductions used in the five studies shows the following order from the most to the least effective, ([29]), ([26]), ([24]), ([5]), and ([8]). The less effective reductions usually implement algorithms that are special cases of the more complete ones. They are simple and fast algorithms but can only pick up a smaller set of edges to be contracted or deleted. The result in Table 1 indicates that the LP relaxation of DCB is a strong formulation which solves more instances to optimality. It also demonstrates the importance of preprocessing.

For computational studies based on instances from StienLib Library, there are three recent studies solving various sets of instances from the library. However, the VLSI set is the only one solved by all studies and it is one of the hardest among various classes of STP instances. Table 2 shows the number of instances solved to optimality by the three studies. Each row records the result from solving a subset of VLSI instances with the number in the parentheses indicating total number of instances in the subset. Computational times reported in ([29]) are far better than the rest. However, preprocessing results are not reported for each instance in this study. Between the remaining two studies, the reduction algorithms designed specifically for grid graphs [34] outperform those developed for general STP [24] by a great margin. That is an important reason why Uchoa, Aragão, and Ribeiro [34] are able to solve more instances to optimality in their computational study than the study conducted by Koch and Martin [24].

Computational results confirm that cutting plane approaches coupled with good preprocessing algorithms are very effective at solving Steiner tree problems on graphs.

References

[1] Y. Aneja, An integer programming approach to the Steiner problem in graphs, *Networks* 10 (1980), pp. 167-178.

[2] A. Balakrishnan, T. Magnanti, and R. Wong, A dual-ascent procedure for large-scale uncapacitated network design, *Operations Research* 37 (1989), pp. 716-740.

[3] A. Balakrishnan and N. Patel, Problem reduction methods and a tree generation algorithm for the Steiner network problem, *Networks* 17 (1987), pp. 65-85.

[4] J. Beasley, An algorithm for the Steiner problem in graphs, *Networks* 14 (1984), pp. 147-159.

[5] J. Beasley, An SST-based algorithm for the Steiner problem in graphs, *Networks* 19 (1989), pp. 1-16.

[6] J. Beasley, http://graph.ms.ic.ac.uk/info.html, 1990.

[7] S. Chopra and E. Gorres, On the node weighted Steiner tree problem, Working Paper, New York University (1991).

[8] S. Chopra, E. Gorres, and M.R. Rao, Solving the Steiner tree problem on a graph using branch and cut, *ORSA Journal on Computing* 4 (1992), pp. 320-335.

[9] S. Chopra and M.R. Rao, The Steiner Tree Problem I: Formulations, composition and extension of facets, *Mathematical Programming* 64 (1994), pp. 209-229.

[10] S. Chopra and M.R. Rao, The Steiner Tree Problem II: Properties and classes of facets, *Mathematical Programming* 64 (1994), pp. 231-246.

[11] S. Chopra, and C-Y Tsai, A branch-and-cut approach for minimum cost multi-level network design, to appear in *Discrete Mathematics*.

[12] S.E. Dreyfuss and R.A. Wagner, The Steiner problem in graphs, *Networks* 1 (1972), pp. 195-207.

[13] C. Duin, Steiner's problem in graphs, Ph.D. thesis, University of Amsterdam (1993).

[14] C. Duin and A. Volgenant, Some generalizations of the Steiner problem in graphs, *Networks* 17 (1987), pp. 353-364.

[15] C. Duin and A. Volgenant, An edge elimination test for the Steiner problem in graphs, *Operations Research Letters* 8 (1989), pp. 79-83.

[16] C. Duin and A. Volgenant, Reduction tests for the Steiner problem in graphs, *Networks* 19 (1989), 549-567.

[17] S. Engevall, Maud Göthe-Lundgren, and P. Värbrand, A strong lower bound for the node weighted Steiner tree problem, *Networks* 31 (1998), pp. 11-17.

[18] M.X. Goemans, The Steiner tree polytope and related polyhedra, *Mathematical Programming* 63 (1994), pp. 157-182.

[19] M.X. Goemans and D. Bertsimas, Survivable networks, linear programming relaxations and the parsimonious property, *Mathematical Programming* 60 (1993), pp. 145-166.

[20] M.X. Goemans and Y. Myung, A catalog of Steiner tree formulations, *Networks* 23 (1993), pp. 19-28.

[21] E. Gorres, On the node weighted Steiner tree problem, Ph.D. dissertation, New York University (1992).

[22] S.L. Hakimi, Steiner's problem on graphs and its applications, *Networks* 1 (1971), pp. 113-133.

[23] M. Karpinski and A. Zelikovsky, New approximation algorithms for the Steiner tree problems, *Journal of Combinatorial Optimization* 1 (1997), pp. 47-65.

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

[25] T. Koch and A. Martin and S. Voß, SteinLib: An Updated Library on Steiner Tree Problems in Graphs, *ZIB-Report 00-37*, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2000. http://elib.zib.de/steinlib.

[26] A. Lucena and J.E. Beasley, A branch and cut algorithm for the Steiner problem in graphs, *Networks* 31 (1998), pp. 39-59.

[27] F. Margot, A. Prodon, and Th.M. Liebling, Tree polytope on 2-trees, *Mathematical Programming* 63 (1994), pp. 183-191.

[28] T. Polzin and S. V. Daneshmand, A Comparision of Steiner Tree Relaxation, to appear in *Discrete Applied Mathematics.*

[29] T. Polzin and S. V. Daneshmand, Improved Algorithms for the Steiner problem in networks, to appear in *Discrete Applied Mathematics.*

[30] A. Segev, The node weighted Steiner tree problem, *Networks* 17 (1987), pp. 1-17.

[31] U.H. Suhl and H. Hilbert, A branch-and-cut algorithm for solving generalized multiperiod Steiner problems in graphs, *Networks* 31 (1998), pp. 273-282.

[32] P. Sussner, P.M. Pardalos, and G.X. Ritter, On integer programming approaches for morphological template decomposition, *Journal of Combinatorial Optimization* Vol.1 No.2 (1997), pp. 177-188.

[33] H. Takahashi and A. Matsuyama, An approximation solution for the Steiner problem in graphs, *Math. Japonica* 6 (1980), pp. 573-577.

[34] E. Uchoa, M.P. de Aragão, and C. Ribeiro, Preprocessing Steiner probelms from VLSI layout, *PUC-Rio Technical Report MCC* 32/99, (1999).

[35] J.A. Wald and C.J. Colbourn, Steiner trees in outerplanar graphs, *Proceedings of 13th southeastern conference on combinatorics, graph theory and computing* (1982), pp. 15-22.

[36] J.A. Wald and C.J. Colbourn, Steiner trees, partial 2-trees and minimum IFI networks, *Netwokrs* 13 (1983), pp. 159-167.

[37] P. Winter, Reductions for the rectilinear Steiner tree problem, *Networks* 26 1995, pp. 187-198.

[38] P. Winter and J. M. Smith, Path-distance heuristics for the Steiner problem in undirected networks, *Algorithmica* 7 (1992), pp. 309-327.

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

[40] A. Zelikovsky, An 11/6-approximation algorithm for the network Steiner problem, *Algorithmica* 9 1993, pp. 463-470.