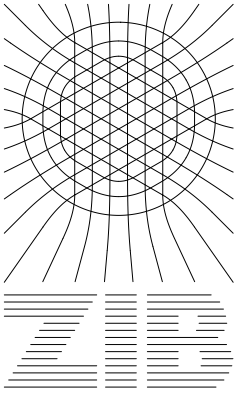
Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany



DANIEL BIENSTOCK ANDREAS BLEY

Capacitated Network Design with Multicast Commodities

ZIB-Report 00-14 (March 2000)

Capacitated Network Design with Multicast Commodities

Daniel Bienstock* and Andreas Bley†

Abstract

This paper addresses the problem of designing a minimum cost network whose capacities are sufficiently large to allow a feasible routing of a given set of multicast commodities. A multicast commodity consists of a set of two or more terminals that need to be connected by a so called broadcast tree, which consumes on all of its edges a capacity as large as the demand value associated with that commodity.

We model the network design problem with multicast commodities as the problem of packing capacitated Steiner trees in a graph. In the first part of the paper we present three mixed-integer programming formulations for this problem. The first natural formulation uses only one integer capacity variable for each edge and and one binary tree variable for each commodity-edge pair. Applying well-known techniques from the Steiner tree problem, we then develop a stronger directed and a multi-commodity flow based mixed-integer programming formulation.

In the second part of the paper we study the associated polyhedra and derive valid and even facet defining inequalities for the natural formulation.

Finally, we describe separation algorithms for these inequalities and present computational results that demonstrate the strength of our extended formulations.

Keywords: Capacitated Network Design, Steiner Tree Packing
Mathematical Subject Classification (1991): 90B12, 90C11, 90C27, 90C90

1 Introduction

In this paper we study the problem of designing a minimum cost network whose capacities are sufficiently large to accommodate a collection of multicast commodities simultaneously.

In contrast to a unicast commodity, which consists of two terminals that must be connected by a path, a multicast commodity consists of a set of two or more terminals that need to be connected by a so called broadcast tree. This tree consumes on all of its edges the same capacity—the demand value associated with that commodity.

Multicast traffic arises in various new network applications like video-conferencing, tele-teaching, or audio broadcasts. Since there is no adequate support for multicast traffic in the existing network protocols, in most of the networks multicast traffic is

*Dept. Industrial Engineering and Operations Research, Columbia University, New York, NY 10027

†Konrad-Zuse-Zentrum für Informationstechnik Berlin (ZIB), Takustrasse 7, D-14195 Berlin, Germany

realized via unicast routing. This means that each multicast commodity is replaced by a number of unicast commodities with the same bandwidth demand. Obviously, the bandwidth required to accommodate all these unicast commodities may be substantially larger than the bandwidth needed for the multicast commodity if the number of terminals is large. Better support for multicast traffic is expected to be introduced into the protocols of the next generation networks, which will raise the question of how to design networks for a mix of unicast and multicast traffic.

Mathematically, the network design problem with multicast commodities can be modeled as the problem of packing capacitated Steiner trees in a weighted capacitated graph. Let $G = (V, E)$ be an undirected graph, $T^1, \ldots, T^N$ with $T^k \subseteq V$ be a collection of node sets, and $d^1, \ldots, d^N \in \mathbb{Q}^+$ be demand values associated with these node sets T^k. Each pair (T^k, d^k) corresponds to a multicast commodity in the network design problem. We denote $\mathcal{N} := \{1, \ldots, N\}$. For $T \subseteq V$ we call an edge set $S \subseteq E$ a spanner with respect to T if S connects all pairs of nodes in T. An edge-minimal spanner is a Steiner tree.

For given edge capacities $y_e \in \mathbb{N}$, $e \in E$ a collection of edge sets $S^1, \ldots, S^N$, such that S^k is a spanner with respect to T^k and $\sum_{k:e \in S^k} d^k \leq y_e$ for each edge $e \in E$, is called a y-capacitated spanner packing. If, for all $k \in \mathcal{N}$, the set S^k is a Steiner tree with respect to T^k, then such a collection is called y-capacitated Steiner tree packing.

The goal in the capacitated Steiner tree packing problem is to find integer edge capacities $y \in \mathbb{N}^E$ that minimize $\sum_{e \in E} y_e w_e$ for some non-negative edge costs w_e, $e \in E$, such that there exists a feasible y-capacitated Steiner tree packing.

In practical network design problems, one usually wishes to install capacities which are multiples of certain units or which are chosen from a finite set of feasible capacities. However, in this paper we restrict to the case where only multiples of one capacity unit λ_1 can be installed. To keep notation as simple as possible, we assume that $\lambda_1 = 1$ and that the original demand values are scaled accordingly.

There has been some work on the uncapacitated version of this problem, which was motivated by applications in VLSI design. In the uncapacitated version for each terminal set T^k the associated demand value is $d^k = 1$ and all the capacities are given and integer. The problem is to layout the Steiner trees in the graph such that no capacity constraint is violated. In [Cho92] Chopra discusses several models and present a Lagrangean algorithm to solve the problem, in [Mar92, GMW96c, GMW96a, GMW96b, GMW96d] Martin et.al. investigate the polyhedral structure of the polytop given by all feasible Steiner tree packings.

The single Steine tree problem in graphs has been extensively studied in various variants in the literature. There are many results about theoretical approximability, different formulations, and computational issues. We cannot give a complete list of all relevant publications here. Let us just mention the papers [CR94a, CR94b], where the polyhedral structure of the classical Steiner tree polyhedron is studied, [Goe94, GM93], where several alternative mixed-integer formulations are discussed, and [KM98], which focuses on computational aspects of the problem.

In the first part of this paper we present three mixed-integer programming formulations for this problem. The first (natural) formulation uses only one integer capacity variable for each edge and and one binary tree variable for each commodity-edge pair. Applying well-known modelling techniques for the Steiner tree problem, we then develop stronger extended mixed-integer programming formulations for the Steiner tree packing network design problem. One of these models is based on the directed cut formulation, the other one is based on the multicommodity-flow formulation for the Steiner tree problem.

In the second part of the paper we study the polyhedra associated with these formulations, focusing on the polyhedron given by the natural formulation. We discuss some basic properties of this polyhedron, its relation to the (single) Steiner tree polyhedron, and derive valid and facet defining inequalities.

Finally, we describe separation algorithms for these inequalities and report on computational tests that demonstrate the strength of the extended formulations.

2 The Problem

In this section we describe the problem considered in this paper formally.

Definition 2.1 *Let $G = (V, E)$ be a graph and $T \subseteq V$. We call an edge set $S \subseteq E$ a **spanner** with respect to T if S connects all pairs of nodes in T, i.e., the subgraph $(T \cup V(S), S)$ is connected. A spanner which is a tree and whose leaves are in T is called a **Steiner tree**.*

Definition 2.2 *Let $N \in \mathbb{N}$ and denote $\mathcal{N} := \{1, \ldots, N\}$. Given a graph $G = (V, E)$ and a collection of terminal sets $\mathcal{T} = \{T^1, \ldots, T^N\}$, $T^k \subseteq V$, a **spanner packing** is a collection of edge sets $\mathcal{S} = \{S^1, \ldots, S^N\}$, $S^k \subseteq E$, such that for each $k \in \mathcal{N}$ the set S^k is a spanner with respect to T^k. If, for each $k \in \mathbb{N}$, S^k is a Steiner tree with respect to T^k, then $\mathcal{S}$ is called a **Steiner tree packing**, or simply Steiner packing.*

Formally, the problem of finding a feasible capacitated spanner or Steiner tree packing is defined as follows.

Definition 2.3 *(y-capacitated Steiner tree packing problem CSP)*

Instance:

- *undirected graph $G = (V, E)$ with capacities $y : E \to \mathbb{N}$,*
- *terminal sets $\mathcal{T} = \{T^1, \ldots, T^N\}$, $T^k \subseteq V$ for all $k \in \mathcal{N}$, and*
- *demand values $d^1, \ldots, d^N$, $d^k \in \mathbb{Q}^+$ for all $k \in \mathcal{N}$.*

Problem:

Find a collection of sets $\mathcal{S} = \{S^1, \ldots, S^N\}$, $S^k \subseteq E$ for all $k \in \mathcal{N}$, such that

(i) S^k is a spanner with respect to T^k for each $k \in \mathcal{N}$ and

(ii) $\sum_{k:e \in S^k} d^k \leq y_e$ for each $e \in E$.

In the corresponding network design problem the goal is to find integer edge capacities $y_e \in \mathbb{N}$, $e \in E$, such that there exists a feasible y-capacitated spanner (or Steiner tree) packing that minimizes the cost $\sum_{e \in E} y_e w_e$ for some given non-negative edge costs $w_e \in \mathbb{N}$, $e \in E$.

Definition 2.4 *(Steiner tree packing network design problem SPN)*

Instance:

- *undirected graph $G = (V, E)$ with costs $w_e \in \mathbb{N}$, $e \in E$,*
- *terminal sets $\mathcal{T} = \{T^1, \ldots, T^N\}$ with $T^k \subseteq V$ for all $k \in \mathcal{N}$, and*
- *demand values $d^1, \ldots, d^N$, $d^k \in \mathbb{Q}^+$ for all $k \in \mathcal{N}$.*

Problem:

Find capacities $y_e \in \mathbb{N}$, $e \in E$, and sets $\mathcal{S} = \{S^1, \ldots, S^N\}$, $S^k \subseteq E$ for all $k \in \mathcal{N}$, such that

(i) S^k *is a spanner with respect to* T^k *for each* $k \in \mathcal{N}$,

(ii) $\sum_{k:e \in S^k} d^k \leq y_e$ *for each* $e \in E$, *and*

(iii) $\sum_{e \in E} y_e w_e$ *is minimized.*

Obviously, for a given graph G, terminal sets $T^1, \ldots, T^N$, demands $d^1, \ldots, d^N$, and capacities y_e, $e \in E$, any feasible y-capacitated Steiner tree packing is also a y-capacitated spanner packing. Furthermore, it is easy to show that for any non-negative cost function $w : E \to \mathbb{N}$ each spanner packing can be reduced to a Steiner tree packing without increasing its costs. Hence, the Steiner tree packings are the minimal solutions of the spanner packing problems.

Proposition 2.5

(i) For any instance $(G, \mathcal{T}, d, y)$ of CSP *there exists a feasible spanner packing if and only if there exists a feasible Steiner tree packing.*

(ii) For any instance $(G, \mathcal{T}, d, w)$ of SPN *with $w_e \geq 0$ for each $e \in E$ there exists an optimal spanner packing which is a Steiner tree packing.*

Proposition 2.5 justifies the terminology 'Steiner tree packing' introduced above. In principle we allow arbitrary spanner packings as solutions to CSP and SPN, but all minimal solutions will be Steiner tree packings (or at least be easily transformable to Steiner tree packings).

It is not surprising that both problems CSP and SPN are combinatorially hard (See [GJ79, Pap94] for an introduction to computational complexity).

Proposition 2.6

(i) CSP *is $\mathcal{NP}$-complete.*

(ii) SPN *is $\mathcal{APX}$-hard. Furthermore,* SPN *cannot be approximated within a factor of α for $\alpha < \frac{3}{2}$, unless $\mathcal{P} = \mathcal{NP}$.*

The $\mathcal{NP}$-completeness of CSP follows directly from the $\mathcal{NP}$-completeness of the disjoint path problem[GJ79]. Since the minimum weighted Steiner tree problem is $\mathcal{APX}$-hard[BP], so is the Steiner tree packing network design problem SPN. In order to prove that SPN cannot be approximated within a factor of less than $\frac{3}{2}$, even if all edge costs w_e are 1, one can construct a reduction from the exact set partitioning problem, which is $\mathcal{NP}$-hard[GJ79].

Before we continue with the mixed-integer programming formulations, we have to introduce some necessary terminology. Let $G = (V, E)$ be a graph. For two sets $V_1, V_2 \subseteq V$ we denote $[V_1, V_2] := \{uv \in E \mid u \in V_1, v \in V_2\}$. Given a vertex set $W \subset V$ the cut induced by W is the set $\delta(W) := [W, V \setminus W]$. A cut $\delta(W)$ is called Steiner cut with respect to some terminal set $T \subseteq V$ if $\emptyset \neq W \cap T \neq T$. Given a partition $\{V_1, \ldots, V_p\}$ of the vertex set, i.e., $V_i \subseteq V$, $V_1 \cup \ldots \cup V_p = V$, and $V_i \cap V_j = \emptyset$ for $i \neq j$, we denote $\delta(V_1, \ldots, V_p) := \bigcup_{i=1}^{p} \delta(V_i)$.

3 Integer Programming Formulations

In the following section we present three different integer linear programming formulations for the Steiner tree packing network design problem and compare the associated LP-relaxations.

3.1 The Undirected Formulation

The first formulation we present for SPN is the natural undirected formulation. It is based on the undirected cut formulation for the (single) Steiner tree problem. For the uncapacitated Steiner tree packing problem a similar formulation was studied in [Cho92, Mar92, GMW96c, GMW96a, GMW96b, GMW96d].

We introduce the following variables:

- $y_e \in \mathbb{N}$ for all $e \in E$: capacity installed on edge e, and
- $x_e^k \in \{0,1\}$ for all $e \in E$, $k \in \mathcal{N}$: one if edge e is contained in the spanner for T^k, zero otherwise.

With these variables, the Steiner tree packing network design problem can be formulated as the following integer linear program.

Formulation 3.1 (Undirected Cut Model)

$$
\begin{aligned}
\min \sum_{e \in E} w_e y_e & & & \\
\sum_{e \in \delta(W)} x_e^k &\geq 1 & k \in \mathcal{N},\ W \subset V,\ \emptyset \neq T^k \cap W \neq T^k, & \quad (1)\\
\sum_{k \in \mathcal{N}} d^k x_e^k &\leq y_e & e \in E, & \quad (2)\\
1 \geq x_e^k &\geq 0 & e \in E,\ k \in \mathcal{N}, & \quad (3)\\
y_e &\geq 0 & e \in E, & \quad (4)\\
x_e^k, y_e &\in \mathbb{N} & e \in E,\ k \in \mathcal{N}. &
\end{aligned}
$$

It is obvious that any integer solution of Formulation 3.1 corresponds to a solution of SPN with the same costs and vice versa. The **Steiner cut inequalities** (1) ensure that for each terminal set T^k, $k \in \mathcal{N}$, all Steiner cuts are covered with at least one edge. That the capacities are sufficiently large to accommodate the demands of all spanners selected is guaranteed by the **capacity constraints** (2).

Note that this formulation contains only $|E| + |E| \cdot N$ variables, but exponentially many inequalities (order of $N \cdot 2^{|V|}$ constraints of type (1)).

We denote the polyhedra associated with the integer linear program 3.1 and with its linear programming relaxation by $\mathcal{P}_u$ and $\mathcal{LP}_u$, i.e.,

$$
\begin{aligned}
\mathcal{P}_u &:= conv\{(y,x) \in \mathbb{Z}^{E \cup E \times \mathcal{N}} \mid (y,x) \text{ satisfies } (1),\ldots,(4)\}, \text{ and}\\
\mathcal{LP}_u &:= conv\{(y,x) \in \mathbb{R}^{E \cup E \times \mathcal{N}} \mid (y,x) \text{ satisfies } (1),\ldots,(4)\}.
\end{aligned}
$$

Let $\mathcal{DP}_u$ be the dominant of of $\mathcal{P}_u$, i.e., $\mathcal{DP}_u := \mathcal{P}_u \oplus \mathbb{R}^{E \cup E \times \mathcal{N}}$.[1] Clearly, for non-negative cost functions w optimizing over $\mathcal{P}_u$ is the same as optimizing over $\mathcal{DP}_u$. Hence, for our practical application it would be sufficient to study the polyhedron

[1] $A \oplus B := \{a + b \mid a \in A,\ b \in B\}$.

$\mathcal{DP}_u$ in order to get a strong mixed-integer programming formulation for the Steiner packing network design problem.

Since in this paper we will also present some inequalities which help to strengthen the linear programming relaxation in practice and which are not valid for general spanner packings but only for Steiner tree packings, we formally introduce the polyhedron defined by the convex hull of all feasible Steiner tree packings:

$$\mathcal{P}_u^* = conv\{(y, \chi^{S^1}, \ldots, \chi^{S^N}) \in \mathcal{P}_u \mid S^k \text{ is Steiner tree w.r.t. } T^k\}.$$

3.2 The Directed Formulation

Given an instance $(G, \mathcal{T}, d, w)$ of the Steiner tree packing network design problem SPN one can restate the problem in a directed context as follows: Let $D = (V, A)$ the bidirected graph associated with G, i.e., the arc set A contains arcs (u, v) and (v, u) if and only if $uv \in E$. For each $k \in \mathcal{N}$ we choose one vertex $r^k \in T^k$ as root. Analogously to the undirected case, we now define the directed versions of a spanner and a spanner packing.

Definition 3.2 *A **directed spanner** for $T \subseteq V$ is an arc set $S \subseteq A$ such that the subgraph (V, S) contains a directed (r, t)-path for all $t \in \bar{T} = T \setminus \{r\}$. A directed spanner which is an arborescence (rooted at r) and whose leaves are in T is called a **Steiner arborescence**.*

Definition 3.3 *A directed spanner or Steiner arborescence packing is a collection of arc sets $\mathcal{S} = \{S^1, \ldots, S^N\}$, $S^k \subseteq A$, such that for each $k \in \mathcal{N}$ the set S^k is a directed spanner or Steiner arborescence for T^k, respectively.*

To formulate the Steiner tree packing problem in the directed context, we introduce the following variables:

- $y_e \in \mathbb{N}$ for all $e \in E$: the capacity installed on edge e,
- $x_a^k \in \{0, 1\}$ for all $a \in A$, $k \in \mathcal{N}$: one if arc a is contained in the (directed) spanner for T^k, zero otherwise.

Note that by definition the capacities installed on the edges are undirected and each spanner consumes the same capacity on all of its edges, no matter which vertex is chosen as root and in which direction the spanner is oriented. Hence, it is sufficient to introduce only one capacity variable y_e for the capacity installed for both directions together.

Let $\delta(W)$, $W \subset V$ be a Steiner cut for a terminal set T^k in the undirected graph G. Without loss of generality we may assume that $r^k \notin W$. The corresponding undirected Steiner cut inequality $\sum_{e \in \delta_G(W)} x_e^k \geq 1$ in Formulation 3.1 translates to the **directed Steiner cut inequality**

$$\sum_{a \in \delta_D^-(W)} x_a^k \geq 1,$$

with $\delta_D^-(W) := [V \setminus W, W]_D := \{(v, w) \in A \mid v \in V \setminus W, w \in W\}$.

Consider an undirected edge $e = uv \in E$ and let (u, v) and (v, u) be its corresponding arcs in A. In the undirected model each $k \in \mathcal{N}$ whose spanner contains e consumes a capacity of d^k on e. In the undirected model this capacity is consumed

if either[2] (u, v) or (v, u) are contained in the directed spanner. Hence the capacity constraint for e can be translated into

$$\sum_{k \in \mathcal{N}} d^k (x^k_{(u,v)} + x^k_{(v,u)}) \leq y_{uv}.$$

The Steiner tree network design problem SPN can now be formulated as

Formulation 3.4 (Directed Cut Model)

$$\begin{aligned}
\min \sum_{e \in E} w_e y_e & & & \\
\sum_{a \in \delta^-(W)} x^k_a \geq 1 & & k \in \mathcal{N},\, W \subset V, & \\
& & r^k \notin T^k \cap W \neq T^k, & (5) \\
\sum_{k \in \mathcal{N}} d^k \left(x^k_{(u,v)} + x^k_{(v,u)} \right) \leq y_e & & uv = e \in E, & (6) \\
1 \geq x^k_a \geq 0 & & a \in A,\, k \in \mathcal{N}, & (7) \\
y_e \geq 0 & & e \in E, & (8) \\
x^k_a, y_e \in \mathbb{N} & & a \in A,\, k \in \mathcal{N}. &
\end{aligned}$$

We define the polyhedra associated with this formulation as

$$\begin{aligned}
\mathcal{P}_d &= conv\{(y, x) \in \mathbb{Z}^{E \cup A \times \mathcal{N}} \mid (y, x) \text{ satisfies } (5), \ldots, (8)\}, \text{ and} \\
\mathcal{LP}_d &= conv\{(y, x) \in \mathbb{R}^{E \cup A \times \mathcal{N}} \mid (y, x) \text{ satisfies } (5), \ldots, (8)\}.
\end{aligned}$$

We can be easily shown that the directed cut formulation for the Steiner tree packing network design problem is stronger than the undirected cut formulation 3.1 in terms of the lower bounds these formulations provide.

3.3 A Multi-commodity Flow Formulation

In this subsection we present a third formulation of the Steiner tree packing problem, which is based on multi-commodity flows. It can be derived from the directed cut formulation 3.4 by replacing the directed cut formulation for each of the N Steiner tree problems with the corresponding multi-commodity flow formulation for the Steiner tree problem (See for example [Bea84]). It is easy to show that the new formulation is equivalent to the the directed cut formulation in terms of the lower bound provided by the respective LP-relaxations for non-negative cost functions. The advantage of the multi-commodity flow based formulation is that, in contrast to the two formulations presented before, it contains only a polynomial number of variables and constraints. With such a strong *and* compact formulation it is—in principle—possible to generate the integer linear program for a given problem instance and employ a general purpose integer programming solver to solve the problem. However, for real-world problem instances this approach is infeasible. The number of integer variables in the formulation is much larger than what can be handled by any state-of-the-art solver in reasonable time.

Let $D = (V, A)$ and r^k for each $k \in \mathcal{N}$ be defined as in the previous section. In the multi-commodity flow formulation we model each Steiner tree problem as a multi-commodity flow problem. For each terminal set T^k, $k \in \mathcal{N}$, we introduce the

[2] It is easy to see that by orienting any undirected spanner S^k for T^k only one arc (u, v) or (v, u) is contained in the resulting directed spanner.

commodities (r^k, t), $t \in \bar{T}^k := T^k \setminus \{r^k\}$. The total number of commodities created is $C := \sum_{k \in \mathcal{N}} |\bar{T}^k|$.

In order to ensure that the directed spanner S^k for the terminal set T^k contains a directed (r^k, t)-path for each $t \in \bar{T}^k$, we force an integer flow of value one for each commodity (r^k, t) from r^k to t. An arc a is contained in the spanner S^k, if the flow value for at least one of the commodities (r^k, t), $t \in \bar{T}^k$, is one.

We formulate the problem with the following variables:

- $y_e \in \mathbb{N}$ for all $e \in E$: capacity installed on edge e,
- $x_e^k \in \{0, 1\}$ for all $e \in E$, $k \in \mathcal{N}$: one if edge e is contained in spanner for T^k, zero otherwise, and
- $f_a^{k,t} \in \mathbb{R}_+$ for all $a \in A$, $k \in \mathcal{N}$, $t \in \bar{T}^k$: flow for commodity (r^k, t) on arc a.

Formulation 3.5 (Multi-commodity Flow Model)

$$\min \sum_{e \in E} w_e y_e$$

$$\sum_{a \in \delta^-(v)} f_a^{k,t} - \sum_{a \in \delta^+(v)} f_a^{k,t} = \begin{cases} 1 & v = t \\ -1 & v = r^k \\ 0 & otherwise \end{cases} \qquad \begin{array}{l} v \in V,\ k \in \mathcal{N}, \\ t \in \bar{T}^k, \end{array} \tag{9}$$

$$f_{(u,v)}^{k,t} + f_{(v,u)}^{k,t} \leq x_e^k \qquad \begin{array}{l} uv = e \in E, \\ k \in \mathcal{N},\ t \in \bar{T}^k \end{array} \tag{10}$$

$$\sum_{k \in \mathcal{N}} d^k x_e^k \leq y_e \qquad e \in E, \tag{11}$$

$$1 \geq f_a^{k,t} \geq 0 \qquad \begin{array}{l} a \in A,\ k \in \mathcal{N}, \\ t \in \bar{T}^k, \end{array} \tag{12}$$

$$1 \geq x_e^k \geq 0 \qquad e \in E,\ k \in \mathcal{N}, \tag{13}$$

$$y_e \geq 0 \qquad e \in E \tag{14}$$

$$x_e^k, y_e \in \mathbb{N} \qquad e \in E,\ k \in \mathcal{N}.$$

As stated previously, this model contains only a polynomial number of variables and constraints, namely $|E| + |E| \cdot N + |A| \cdot C$ variables and $|V| \cdot C + |E| \cdot C + |E|$ linear constraints.

We define the polyhedra associated with this integer linear program and its LP-relaxation as

$$\mathcal{P}_{mcf} = conv \left\{ \begin{array}{l} (y, x, f) \in \mathbb{Z}^{E \cup E \times \mathcal{N}} \times \mathbb{R}^{A \times C} \mid \\ (y, x, f) \text{ satisfies } (9), \ldots, (14) \end{array} \right\}, \text{ and}$$

$$\mathcal{LP}_{mcf} = conv \left\{ \begin{array}{l} (y, x, f) \in \mathbb{R}^{E \cup E \times \mathcal{N} \cup A \times C} \mid \\ (y, x, f) \text{ satisfies } (9), \ldots, (14) \end{array} \right\}.$$

3.4 Comparison

In this section we present a few results about the strength of the three integer programming formulations we have introduced. Obviously, for non-negative costs the optimal integer solution values of the formulations 3.1, 3.4, and 3.5 are identical. But in terms of the optimum solution value of their linear programming relaxations these formulations differ.

Let $(G, \mathcal{T}, d, w)$ be an instance of the Steiner tree packing problem with non-negative edge costs w_e, $e \in E$. Define

$$\begin{aligned} LB_u &:= \min\{\sum_{e \in E} w_e y_e \mid (y, x) \in \mathcal{LP}_u\} \\ LB_d &:= \min\{\sum_{e \in E} w_e y_e \mid (y, x) \in \mathcal{LP}_d\} \\ LB_{mcf} &:= \min\{\sum_{e \in E} w_e y_e \mid (y, x, f) \in \mathcal{LP}_{mcf}\} \end{aligned}$$

We will not prove the following two results in this paper. Both follow directly from the well-known results about the strength of the underlying Steiner tree formulations (see [GM93]).

Proposition 3.6 *The values LB_d and LB_{mcf} are independent from the choice of the root nodes $r^k \in T^k$ for each $k \in \mathcal{N}$.*

Proposition 3.7

(i) $LB_u \leq LB_d = LB_{mcf}$.

(ii) There are instances $\phi = (G, \mathcal{T}, d, w)$ with $LB_u(\phi) < LB_d(\phi)$.

In Section 6 we will use an explicit formulation containing one 0/1-variable for each possible Steiner tree S^k for T^k, $k \in \mathcal{N}$, to model the feasibility problem CSP. One can show that the LP-relaxation of an analogous formulation for the Steiner tree packing network design problem is even stronger than the relaxations of the directed cut formulation 3.4 and the multi-commodity flow formulation 3.5. But this explicit formulation contains exponentially many variables and the column generation problem for this formulation reduces to a Steiner tree problem, which is $\mathcal{NP}$-hard. Hence, the explicit formulation does not seem suitable to solve practical problem instances. Besides that, one can show that this explicit formulation is exactly as strong as the formulations obtained by adding for each $k \in \mathcal{N}$ all (lifted) facets of the (single) Steiner tree polytopes for T^k to the directed cut or multi-commodity flow formulation. Hence, we think that for a practical implementation working with formulation 3.4 or 3.5 and strengthening its LP-relaxation with additional inequalities is preferable.

4 Basic Polyhedral Results

In order to apply linear programming techniques successfully, a good description of the polyhedra associated with the formulations of the Steiner packing network design polytope is indispensable. In the following sections we will prove some basic properties of these polyhedra and derive some valid and facet defining inequalities that can be used to strengthen the LP-relaxations. We will restrict our attention to the polyhedra associated with the undirected cut formulation. Analogous results can be shown for the polyhedra based on the other two formulations and the inequalities we derive for the undirected formulation translate to inequalities for the directed and multi-commodity-flow based formulations.

In this section we study the dimension of the polyhedra $\mathcal{P}_u$ and $\mathcal{P}_u^*$ and show that facet defining inequalities of the (single) Steiner tree polyhedron yield facet-defining inequalities for $\mathcal{P}_u$ as well.

The dimension of $\mathcal{DP}_u$ is obviously $(N+1)|E|$. Let $SP(G,T)$ and $SP^*(G,T)$ denote the spanner and Steiner tree polyhedron for the graph G and the terminal set T, respectively, i.e.,

$$\begin{aligned} SP(G,T) &:= conv\{\chi^S \mid S \text{ is spanner for } T\}, \\ SP^*(G,T) &:= conv\{\chi^S \mid S \text{ is Steiner tree for } T\}. \end{aligned}$$

Theorem 4.1

(i) $dim(\mathcal{P}_u) = |E| + \sum_{k\in\mathcal{N}} dim(SP(G,T^k))$

(ii) $dim(\mathcal{P}_u^*) = |E| + \sum_{k\in\mathcal{N}} dim(SP^*(G,T^k))$

Proof. We will only prove (i) here. Part (ii) can be shown analogously.

For notational convenience, let $dim(k) := dim(SP(G,T^k))$. Obviously $dim(\mathcal{P}_u) \leq |E| + \sum_{k\in\mathcal{N}} dim(k)$.

To see that $dim(\mathcal{P}_u) \geq |E| + \sum_{k\in\mathcal{N}} dim(k)$ we present $|E| + \sum_{k\in\mathcal{N}} dim(k)$ affinely independent vectors in $\mathcal{P}_u$. For each $k \in \mathcal{N}$ let $\chi_0^k, \ldots, \chi_{dim(k)}^k$ be $dim(k)+1$ affinely independent integer points (i.e., characteristic vectors of spanners for T^k) in $SP(G,T^k)$. Let $D = \sum_{k\in\mathcal{N}} d^k$ and $y_e = D$ for all $e \in E$. Its easy to see that the following vectors are affinely independent:

- $(y, (\chi_0^1, \ldots, \chi_0^N))$,
- $(y, (\chi_0^1, \ldots, \chi_0^{k-1}, \chi_i^k, \chi_0^{k+1}, \ldots, \chi_0^N)$ for each $k \in \mathcal{N}$ and $i = 1, \ldots, dim(k)$, and
- $(y + \chi_e, (\chi_0^1, \ldots, \chi_0^N))$.

Since the capacity D is sufficiently large to accommodate all demands simultaneously, each of these $|E| + \sum_{k\in\mathcal{N}} dim(k) + 1$ vectors corresponds to a feasible solution to the Steiner packing network design problem. Hence, $dim(\mathcal{P}_u) \geq |E| + \sum_{k\in\mathcal{N}} dim(k)$. □

Corollary 4.2 *If G is two-connected, then $dim(\mathcal{P}_u) = dim(\mathcal{P}_u^*) = (N+1)|E|$.*

Proof. If G is biconnected, then the Steiner tree polyhedron $SP^*(G,T^k)$ is full-dimensional for each terminal set T^k. □

In the remainder of this paper we assume that G is biconnected. Otherwise, if there is a cut vertex v_o in G, the Steiner packing network design problem decomposes into smaller instances on the biconnected components of G. Now we will present the main result of this section.

Theorem 4.3

(i) Any facet defining inequality of $SP(G,T^k)$ defines a facet of $\mathcal{P}_u$.

(ii) Any facet defining inequality of $SP^*(G,T^k)$ defines a facet of $\mathcal{P}_u^*$.

Proof. Analogous to the proof of Theorem 4.1, except that for one $k' \in \mathcal{N}$ we have only $dim(k') := dim(SP(G,T^{k'}))$ (or $dim^*(k') := dim(SP^*(G,T^{k'}))$) many affinely independent vectors in $SP(G,T^{k'})$ ($SP^*(G,T^{k'})$) which satisfy the given inequality for $SP(G,T^{k'})$ ($SP^*(G,T^{k'})$) with equality. □

Corollary 4.4 *For $k \in \mathcal{N}$ and $e \in E$ the inequalities $x_e^k \geq 0$ and $x_e^k \leq 1$ define facets of $\mathcal{P}_u$.*

Proof. Follows directly from the fact that $x_e^k \geq 0$ and $x_e^k \leq 1$ define facets of $SP(G, T^k)$ [GM90]. □

5 Edge Capacity Polytope

In this section we show that from the capacity constraints (2) for one single edge in the graph one can derive several classes of facet defining inequalities for $\mathcal{P}_u$.

For each edge $e \in E$, the capacity constraint (2) can be regarded as a knapsack problem with one integer variable representing the capacity of the knapsack. We will show that this knapsack induces a number of facets for $\mathcal{P}_u$. For an edge $e \in E$ the edge capacity polyhedron is defined as

$$\mathcal{P}_Y(e) = conv\left\{(y_e, x_e) \in \mathbb{N} \times \{0,1\}^N \mid \sum_{k \in \mathcal{N}} d^k x_e^k \leq y_e\right\}.$$

For notational convenience we also write $\mathcal{P}_Y$ instead of $\mathcal{P}_Y(e)$.

In [BGW96] Brockmüller, Günlük and Wolsey studied the edge capacity polyhedron for the case that multiples of two capacity types can be installed on an edge. The edge capacity polyhedron itself is discussed by Van de Leensel in [vdL99] and Van Hoesel et. al. in [vHKvdLS99]. The closely related knapsack problem with one continuous capacity variable is studied by Marchand and Wolsey [MW99]. For another related version of the knapsack problem, in which the binary variables are relaxed to continuous variables, Magnanti et. al. in [MMV95] derive a complete description of the corresponding polyhedron.

Let us begin with two simple observations.

Proposition 5.1 *The dimension of $\mathcal{P}_Y$ is $|\mathcal{N}| + 1$.*

Proposition 5.2 *The trivial inequalities $x^k \geq 0$ and $x^k \leq 1$ define facets of $\mathcal{P}_Y$.*

The following theorem is the main result of this section: Any facet defining inequality for the single edge capacity polyhedron carries over to a facet defining inequality for the entire Steiner tree packing network design problem. This implies that in order to obtain a good description for the polyhedron $\mathcal{P}_u$ in terms of linear inequalities it is necessary to find a good formulation for the capacity polytope $\mathcal{P}_Y$ associated with each single edge.

Theorem 5.3 *Any non-trivial facet-defining inequality for $\mathcal{P}_Y$ defines a facet of $\mathcal{P}_u$.*

The proof of Theorem 5.3 is not very difficult but technical. Since it provides no further insight into the underlying problem it is omitted here.

As a consequence from this theorem, all classes of valid or facet defining inequalities derived for the edge capacity polyhedron also can be applied to strengthen the LP-relaxation of the Steiner tree packing network design problem. We will not discuss all known classes of facet defining and valid inequalities for $\mathcal{P}_Y$ here. A quite comprehensive catalog of such inequalities and separation methods can be found in [vdL99]. In this paper we only present the simplest class of facet defining inequalities

that is derived from the class of cover inequalities for the classical knapsack problem (see [BZ78]).

Let $I \subset \mathcal{N}$ and denote $d(I) := \sum_{k\in I} d^k$ and $D(I) := \sum_{k\in I} \lceil d^k \rceil$. Then the **lifted cover inequality**

$$y \geq \sum_{k\in I} \lceil d^k \rceil x^k + \lceil d(I) \rceil - D(I) \tag{15}$$

is valid for $\mathcal{P}_Y$. If I is a minimal cover for the knapsack defined by $\{x \in \{0,1\}^N \mid \sum_{k\in\mathcal{N}} d^k x^k \leq \lfloor d(I) \rfloor\}$, then inequality (15) (and similar inequalities) can be derived via integer lifting from the corresponding minimal cover inequality for this knapsack and it is facet defining for $\mathcal{P}_Y$ (see [vdL99], [vHKvdLS99]). Also, with $c := D(I) - \lceil d(I) \rceil$ the lifted cover inequalities (15) are a subclass of the c-strong inequalities that that have been investigated in [BGW96].

6 Metric Inequalities

In the previous two sections we have seen that facet defining inequalities from the single Steiner tree problem and from the single edge capacity problem carry over to facets of the Steiner tree packing network design problem. In the following we will show how 'joint inequalities' involving variables for more than one terminal set and more than one edge can be derived. In this section we will present the so called metric inequalities, which in some sense are the simplest (and weakest) joint inequalities.

Let $\mathcal{S}_k$ denote the set of all feasible Steiner trees for the terminal set T^k. Consider a formulation of CSP that contains one binary variable z_S^k for each $k \in \mathcal{N}$ and each possible Steiner tree $S \in \mathcal{S}_k$ for T^k. Suppose the given capacities y do not allow a feasible Steiner packing. Then there is some $\alpha \in \mathbb{R}_+$ such that if α units of capacity are installed simultaneously on all edges in addition to the given capacity y there exists a feasible Steiner packing for the new capacities. The following mixed-integer formulation models the problem of finding the minimum such α.

Formulation 6.1 (Tree Formulation)

$$\min \alpha$$

$$\sum_{S\in\mathcal{S}_k} z_S^k = 1 \qquad k \in \mathcal{N} \tag{16}$$

$$\sum_{k\in\mathcal{N}} \sum_{S\in\mathcal{S}_k : e\in S} d^k z_S^k - \alpha \leq y_e \qquad e \in E \tag{17}$$

$$z_S^k \in \{0,1\} \qquad k \in \mathcal{N},\ S \in \mathcal{S}_k, \tag{18}$$

$$\alpha \geq 0 \tag{19}$$

It is easy to see that this mixed integer program has an optimal solution of value $\alpha_{IP} = 0$ if and only if the original capacities y are sufficiently large to accommodate a Steiner tree packing. Note that, due to Proposition 2.5, in Formulation 6.1 we have to consider only the set $\mathcal{S}_k$ of all Steiner trees for each $k \in \mathcal{N}$ instead of all spanners.

If the integrality constraints (18) are relaxed to $0 \leq z_S^k \leq 1$ we obtain a linear programming relaxation of (6.1) whose solutions correspond to fractional Steiner tree packings. Analogously to the integer case, there exists an optimal solution to the linear programming relaxation of (6.1) with value $\alpha_{LP} = 0$ if and only if there exists a fractional Steiner tree packing satisfying all the capacities y_e.

Now consider the dual of this linear program, whose dual variables are l_e, $e \in E$, for inequalities (17) and π^k, $k \in \mathcal{N}$, for equalities (16).

Formulation 6.2 (Dual Tree Formulation)

$$\max \sum_{k \in \mathcal{N}} d^k \pi^k - \sum_{e \in E} y_e l_e$$

$$\sum_{e \in E} l_e = 1 \tag{20}$$

$$\pi^k - \sum_{e \in S} l_e \leq 0 \qquad k \in \mathcal{N},\ S \in \mathcal{S}_k \tag{21}$$

$$l_e \geq 0 \qquad e \in E \tag{22}$$

Obviously, in any optimal solution of (6.2) inequalities (21) hold with equality, i.e., π^k is the length of a minimum Steiner tree with respect to T^k for edge lengths l_e.

Theorem 6.3 *The following statements are equivalent:*

(i) The capacities y_e, $e \in E$, allow a fractional Steiner tree packing.

(ii) The optimum solution value α_{LP} of the linear programming relaxation of (6.1) *is* 0*.*

(iii) The optimum solution value α_{DLP} of the dual linear program (6.2) *is* 0*.*

(iv) For all non-negative edge lengths $l_e \geq 0$, $e \in E$, the inequality $\sum_{k \in \mathcal{N}} d^k \pi^k \leq \sum_{e \in E} y_e l_e$ holds, where π^k is the length of a minimum Steiner tree w.r.t. T^k for edge lengths l_e.

Proof. LP duality. □

For given non-negative edge lengths $l_e \geq 0$, $e \in E$, the inequality $\sum_{k \in \mathcal{N}} d^k \pi^k \leq \sum_{e \in E} y_e l_e$ is called the **metric inequality** induced by the metric $(l_e)_{e \in E}$. If the metric (l_e) is integer, that is $l_e \in \mathbb{N}$ for all $e \in E$, then these inequalities can be strengthened:

Proposition 6.4

(i) For each metric (l_e), $l_e \geq 0$ for all $e \in E$, the metric inequality $\sum_{k \in \mathcal{N}} d^k \pi^k \leq \sum_{e \in E} y_e l_e$ induced by (l_e) is valid for $\mathcal{P}_u$.

*(ii) For each integer metric (l_e), $l_e \in \mathbb{N}$ for all $e \in E$, the **strengthened metric inequality** $\left\lceil \sum_{k \in \mathcal{N}} d^k \pi^k \right\rceil \leq \sum_{e \in E} y_e l_e$ induced by (l_e) is valid for $\mathcal{P}_u$.*

The separation problem for general metric and strengthened metric inequalities is $\mathcal{NP}$-hard. Thus, in practice we restrict our attention the class of metrics that are induced by cuts sets — or more general by graph partitions — of G and heuristically try to separate violated strengthened metric inequalities for these metrics.

7 Cut-Set and Graph-Partition Inequalities

Let $\{V_1, \ldots, V_p\}$ be a partition of the vertex set of G. Then

$$\chi_e^{\delta(V_1,\ldots,V_p)} := \begin{cases} 1 & e \in \delta(V_1, \ldots, V_p) \\ 0 & \text{otherwise} \end{cases}$$

is the metric induced by this graph partition. With

$$\pi^k := \pi^k(\delta(V_1, \ldots, V_p)) := min\{|S^k \cap \delta\{V_1, \ldots, V_p\}| \mid S^k \text{ is Steiner tree for } T^k\}$$

the strengthened metric inequality induced by the partition $\{V_1, \ldots, V_p\}$ is

$$\left\lceil \sum_{k \in \mathcal{N}} d^k \pi^k \right\rceil \leq \sum_{e \in \delta(V_1, \ldots, V_p)} y_e. \tag{23}$$

We will also refer to this inequality as **cut-set** or **graph-partition inequality**.

We have two simple criteria, when a cut-set or graph-partition inequality is *not* facet-defining for $\mathcal{P}_u$. It is easy to see that whenever $G[V_i]$ is disconnected for some V_i, then the graph-partition inequality (23) for the partition $\{V_1, \ldots, V_p\}$ is dominated by the graph-partition inequality induced by the partition $\{V_1, \ldots, V_{i-1}, V_i^1, \ldots, V_i^q, V_{i+1}, \ldots, V_p\}$, where $V_i^1, \ldots, V_i^q$ are the connected components of V_i.

For $p \geq 3$ let $\hat{G} := G/V_1/\ldots/V_p$ be the graph obtained by contracting the vertices of each component V_i to a single vertex $\hat{v}_i$. Then the graph-partition inequality (23) may be facet defining only if $\hat{G}$ is two-connected. Suppose $\hat{G}$ is not two-connected. Then there is at least one articulation vertex $\hat{v}_j$ in $\hat{G}$, i.e., a vertex whose removal disconnects $\hat{G}$ into two components $\hat{G}_1$ and $\hat{G}_2$. We may assume $\hat{G}_1 = (\hat{V}_1, \hat{E}_1)$ and $\hat{G}_2 = (\hat{V}_2, \hat{E}_2)$ with $\hat{V}_1 = \{\hat{v}_1, \ldots, \hat{v}_{j-1}\}$ and $\hat{V}_2 = \{\hat{v}_{j+1}, \ldots, \hat{v}_p\}$. With $V_j' := V \setminus V_1 \setminus \ldots \setminus V_{j-1}$ and $V_j'' := V \setminus V_{j+1} \setminus \ldots \setminus V_p$ both $\{V_1, \ldots, V_{j-1}, V_j'\}$ and $\{V_j'', V_{j+1} \ldots, V_p\}$ define partitions of G. It is easy to verify that then inequality (23) induced by the partition $\{V_1, \ldots, V_p\}$ is dominated by the sum of the two inequalities (23) induced by the partitions $\{V_1, \ldots, V_{j-1}, V_j'\}$ and $\{V_j'', V_{j+1} \ldots, V_p\}$.

Unfortunately, there are no simple criteria for when such an inequality is facet defining (unless all demands d^k are integral). In general, inequalities (23) do not define facets of $\mathcal{P}_u$, even if $G[V_i]$ is connected for all V_i and $\hat{G}$ is two-connected. Nevertheless, it is possible to derive facet defining inequalities for $\mathcal{P}_u$ from a cut-set or a graph-partition of G.

But under some further conditions all (important) facet defining inequalities for the problem on the shrunken graph $\hat{G}$ define facets for the original problem. Let $\{V_1, \ldots, V_p\}$ be a graph-partition of G and consider the Steiner packing network design instance $(\hat{G}, \hat{\mathcal{T}}, \hat{d}, \hat{w})$ obtained from the original instance $(G, \mathcal{T}, d, w)$ by contracting all edges in $E \setminus \delta(V_1, \ldots, V_p)$. Contracting an edge in $(G, \mathcal{T}, d, w)$ is defined in the canonical way, i.e., removing this edge from G, identifying its endpoints in G and for each $T^k \in \mathcal{T}$.

Let $\hat{\mathcal{P}}_u := \mathcal{P}_u(\hat{G}, \hat{\mathcal{T}}, \hat{d}, \hat{w})$ and $\mathcal{D}\hat{\mathcal{P}}_u := \mathcal{DP}_u(\hat{G}, \hat{\mathcal{T}}, \hat{d}, \hat{w})$ be the Steiner packing network design polyhedra associated with the shrunken problem instance.

Proposition 7.1 *Any valid inequality for $\hat{\mathcal{P}}_u$ is valid for $\mathcal{P}_u$.*

Proof. This observation follows from the fact that every spanner packing for $(G, \mathcal{T}, d)$ induces a spanner packing for $(\hat{G}, \hat{\mathcal{T}}, \hat{d})$.

Unfortunately, inequalities that define facets of $\hat{\mathcal{P}}_u$ do not necessarily also define facets of $\mathcal{P}_u$. But it is not hard to show that every facet defining inequality for $\hat{\mathcal{P}}_u$ either also defines a facet for $\mathcal{P}_u$ or the face it induces in $\mathcal{P}_u$ is contained within a facet given by $x_e^k = 1$ for some $k \in \mathcal{N}$ and $e \in E \setminus \delta(V_1, \ldots, V_p)$. Hence, all 'important' inequalities derived from the shrunken problem, which are those inequalities that define facets of both $\hat{\mathcal{P}}_u$ and $\mathcal{D}\hat{\mathcal{P}}_u$, also define to facets of the polyhedron $\mathcal{DP}_u$.

Theorem 7.2 *Let $\{V_1, \ldots, V_p\}$ be a graph-partition of G such that $G[V_i]$ is connected for each $i = 1, \ldots, p$ and $\hat{G}$ is two-connected. Then any facet defining inequality for $\mathcal{D}\hat{\mathcal{P}}_u$ defines a facet for $\mathcal{DP}_u$.*

Corollary 7.3 *Let $\{V_1, \ldots, V_p\}$ be a graph-partition of G such that $G[V_i]$ is two-connected for each $i = 1, \ldots, p$ and $\hat{G}$ is two-connected. Then any facet defining inequality for $\hat{\mathcal{P}}_u$ defines a facet of $\mathcal{P}_u$.*

Proof. Let $k \in \mathcal{N}$ and $f \in E \setminus \delta(V_1, \ldots, V_p)$. It is easy to see that if $G[V_i]$ is two-connected for each $i = 1, \ldots, p$, then a given Steiner tree packing in the shrunken problem $(\hat{G}, \hat{\mathcal{T}}, \hat{d}, \hat{w})$ can be extended to a Steiner tree packing of the original problem with $x_f^k = 0$ and $y_e = \hat{y_e}$ and $x_e^k = \hat{x_e^k}$ for all $e \in \delta(V_1, \ldots, V_p)$ and $k \in \mathcal{N}$. Hence, for each $k \in \mathcal{N}$ and $f \in E \setminus \delta(V_1, \ldots, V_p)$ we can extend any feasible Steiner tree packing for $(\hat{G}, \hat{\mathcal{T}}, \hat{d}, \hat{w})$ which satisfies the facet defining inequality of $\hat{\mathcal{P}}_u$ with equality to a Steiner tree packing of $(G, \mathcal{T}, d, w)$, still satisfying this inequality with equality. But then the face induced by this inequality is not contained in any of the faces $x_e^k = 1$, $k \in \mathcal{N}$ and $e \in E \setminus \delta(V_1, \ldots, V_p)$.

Applying Theorems 7.1,7.2, and 7.3 one now can 'easily' derive valid and facet defining inequalities for $\mathcal{P}_u$ and $\mathcal{DP}_u$ from $\hat{\mathcal{P}}_u$ and $\mathcal{D}\hat{\mathcal{P}}_u$ induced by some graph-partition. If the shrunken problems are small enough all facets for $\hat{\mathcal{P}}_u$ and $\mathcal{D}\hat{\mathcal{P}}_u$ can be computed and used to strengthen the formulation of the original problem.

In the remainder of this section we present two classes of valid inequalities for $\mathcal{P}_u$ and $\mathcal{P}_u^*$ that are based on cut-sets or graph-partitions. All these inequalities are variants of the lifted cover inequalities derived for the single edge capacity polyhedron.

The idea behind the following inequalities is the following: We combine subsets the variables y_e and x_e^k for the edges $e \in \delta(V_1, \ldots, V_p)$ in such a way, that for each $k \in \mathcal{N}$ the aggregated variable $\hat{x^k}$ of some tree variables x_e^k is bounded from above by a small number (but not necessarily non-negative) and the aggregated variable $\hat{y}$ of the capacity variables y_e represents some integer capacity. Then the feasible set of the aggregated variables can be described by a knapsack with one integer capacity variable and one bounded integer variable for each terminal set. Applying similar techniques as for the single edge capacity polyhedron, we then obtain valid inequalities for this knapsack and transform them into valid inequalities for $\mathcal{P}_u$.

We focus on aggregations for which the aggregated tree variables $\hat{x_e^k}$ are bounded from above by 1. In this case the inequalities we derive are variants of the lifted cover inequalities discussed in Section 5 and can be separated with the same algorithms.

7.1 Cut-Knapsack and Partition-Knapsack Inequalities

The graph-partition inequalities in the previous section were motivated by the observation, that if any Steiner tree S^k for a terminal set T^k must 'cross' the graph-partition at least π^k times, than at least $\pi^k d^k$ capacity units are consumed on the edges of the partition by this terminal set. Now we generalize this idea, taking into account that the chosen Steiner tree might cross the graph partition more than π^k times, thus consuming more than $\pi^k d^k$ capacity units on the partition's edges.

As in the previous section, let $\{V_1, \ldots, V_p\}$ be a graph partition of G and for each $k \in \mathcal{N}$ let π^k denote the minimum number of edges in $\delta(V_1, \ldots, V_p)$ that any spanner for T^k must contain.

We denote $\hat{y} := y(\delta(V_1, \ldots, V_p))$ for the sum of the capacities installed across the partition. For each $k \in \mathcal{N}$ we introduce a pseudo-variable $\hat{x^k}$ such that $\hat{x^k} = 1$ if the

Steiner tree for T^k contains at least $\pi^k + 1$ edges in $\delta(V_1, \ldots, V_p)$. This is achieved, for example, by any of the following settings:

$$\hat{x^k} := \begin{cases} x^k_{e^k} & \text{for some } e^k \in [V_i, V_k] \text{ with} \\ & V_i \cap T^k = \emptyset \text{ or } V_j \cap T^k = \emptyset, \\ \sum_{e \in F^k} x^k_e - (|F^k| - 1) & \text{for some } F^k \subseteq [V_i, V_j] \end{cases} \tag{24}$$

For notational convenience we set $F^k := \{e^k\}$ if $\hat{x^k}$ is chosen to be $x^k_{e^k}$ for some $e^k \in [V_i, V_k]$.

It is easy to see that for all feasible points $(y, x) \in \mathcal{P}_u$ $\hat{x^k} \leq 1$ holds for each $k \in \mathcal{N}$. We define the polyhedron associated with the graph partition

$$\mathcal{P}_Y(V_1, \ldots, V_p) := conv \left\{ (\hat{y}, \hat{x}) \in \mathbb{N}^{1+|\mathcal{N}|} \mid \begin{array}{l} \sum_{k \in \mathcal{N}} d^k \pi^k + \sum_{k \in \mathcal{N}} d^k \hat{x^k} \leq \hat{y} \\ \hat{x^k} \leq 1 \text{ f.a. } k \in \mathcal{N} \end{array} \right\}$$

The following proposition describes how valid inequalities for this knapsack translate to valid inequalities for $\mathcal{P}_u$.

Proposition 7.4 *Let $\hat{y}$ and $\hat{x^k}$, $k \in \mathcal{N}$, be defined as above and let $\hat{\alpha}\hat{y} + \hat{\beta}\hat{x} \leq \hat{\gamma}$ be a valid inequality for $\mathcal{P}_Y(V_1, \ldots, V_p)$. Then $\alpha y + \beta x \leq \gamma$ is a valid inequality for $\mathcal{P}_u$, where*

$$\begin{aligned} \alpha_e &:= \begin{cases} \hat{\alpha} & e \in \delta(V_1, \ldots, V_p) \\ 0 & else \end{cases} \\ \beta^k_e &:= \begin{cases} \hat{\beta^k} & e \in F^k \\ 0 & else, \end{cases} \\ \gamma &:= \hat{\gamma} - \sum_{k \in \mathcal{N}} (|F^k| - 1)\hat{\beta^k} \end{aligned}$$

Analogous to Section 5 we derive the lifted cover inequalities (25) from the knapsack $\mathcal{P}_Y(V_1, \ldots, V_p)$, which we call **cut-knapsack inequalities** or **partition-knapsack inequalities**:

$$\begin{aligned} \sum_{e \in \delta(V_1, \ldots, V_p)} y_e \quad \geq \quad & \sum_{k \in I} \sum_{e \in F^k} \lceil d^k \rceil x^k_{e^k} + \left\lceil \sum_{k \in \mathcal{N}} d^k \pi^k + \sum_{k \in I} d^k \right\rceil \\ & - \sum_{k \in I} |F^k| \lceil d^k \rceil. \end{aligned}$$

where $I \subseteq \mathcal{N}$ and $F^k \subseteq \delta(V_1, \ldots, V_p)$ for $k \in I$. Note that for $I = \emptyset$ the cut-knapsack or partition-knapsack inequality (25) reduces to the the cut-set or graph-partition inequality (23).

7.2 Flow-Cut and Flow-Partition Inequalities

The basic idea underlying both the graph-partition and the partition-knapsack inequalities (or cut-set and cut-knapsack inequalities, respectively) was to assign an integer capacity to the total capacity across a graph-partition. After adding these inequalities to the LP-relaxation of the undirected cut Formulation 3.1 there are feasible extremal points which assign the required integer amount of capacity across a partitions but distribute this capacity fractionally among the partition's edges. In

this section we generalize the graph-partition (or cut-set) inequalities to include capacity variables y_e only for a subset F' of the edges of the partition. To achieve this, we "substitute" the capacity variables y_e of the other edges of the partition, which we call the flow edges, by the tree variables x_e^k on these edges. The inequalities derived from these so called flow-partitions cut off some of the fractional extreme points from the feasible region.

Let $\{V_1, \ldots, V_p\}$ be a graph partition of G and π^k, $k \in \mathcal{N}$, be defined as in the previous sections. Furthermore, choose a subset of flow-edges $F \subset \delta(V_1, \ldots, V_k)$ and let $F' := \delta(V_1, \ldots, V_p) \setminus F$ be the set of non-flow edges of $\delta(V_1, \ldots, V_p)$.

Since any spanner for T^k must contain at least π^k edges of $\delta(V_1, \ldots, V_p)$, it consumes at least $(\pi^k - x^k(F))d^k$ capacity units on the non-flow edges F' of the partition. Hence, the following inequality must be satisfied by any feasible solution (y, x) in $\mathcal{P}_u$:

$$\sum_{k \in \mathcal{N}} d^k(\pi^k - x^k(F)) \leq y(F').$$

We set $\hat{y} := y(F')$ and $\hat{x^k} := \pi^k - x^k(F)$ and define the polyhedron associated with the flow-partition $(\{V_1, \ldots, V_p\}, F)$

$$\mathcal{P}_Y(V_1, \ldots, V_p, F) := conv \left\{ (\hat{y}, \hat{x}) \in \mathbb{N}^{1+|\mathcal{N}|} \mid \begin{array}{l} \sum_{k \in \mathcal{N}} d^k(\pi^k - \hat{x^k}) \leq \hat{y} \\ \hat{x^k} \leq \pi^k \text{ f.a. } k \in \mathcal{N} \end{array} \right\}.$$

Analogously to the partition-knapsack polyhedron, one can show how valid inequalities for the flow-partition polyhedron translate to valid inequalities for $\mathcal{P}_u$.

Proposition 7.5 *Let $\hat{y}$ and $\hat{x^k}$, $k \in \mathcal{N}$, be defined as above and let $\hat{\alpha}\hat{y} + \hat{\beta}\hat{x} \leq \hat{\gamma}$ be a valid inequality for $\mathcal{P}_Y(V_1, \ldots, V_p, F)$. Then $\alpha y + \beta x \leq \gamma$ is a valid inequality for $\mathcal{P}_u$, where*

$$\begin{aligned} \alpha_e &:= \begin{cases} \hat{\alpha} & e \in F' \\ 0 & else \end{cases} \\ \beta_e^k &:= \begin{cases} -\hat{\beta^k} & e \in F \\ 0 & else, \end{cases} \\ \gamma &:= \hat{\gamma} - \sum_{k \in \mathcal{N}} \pi^k \hat{\beta^k} \end{aligned}$$

Note that, in contrast to the the single edge polyhedron and the partition knapsack polyhedron, the $\hat{x}$-variables in the knapsack induced by a flow-partition may obtain values greater than one.

Nevertheless, it can be shown that the following slightly modified version of the lifted cover inequalities for the single edge capacity polyhedron is valid and, under some conditions, even facet defining for the flow-partition polyhedron $\mathcal{P}_Y(V_1, \ldots, V_p, F)$:

$$\hat{y} \geq \sum_{k \in I} \lceil d^k \rceil \hat{x^k} - \sum_{k \in I} \lceil d^k \rceil \pi^k + \lceil \sum_{k \in I} d^k \pi^k \rceil, \tag{25}$$

where $I \subseteq \mathcal{N}$.

Applying the substitution described in Proposition 7.5 to inequalities (25) we derive the **flow-cut inequalities** or **flow-partition inequalities**:

$$\sum_{e \in F'} y_e \geq \left\lceil \sum_{k \in I} d^k \pi^k \right\rceil - \sum_{k \in I} \sum_{e \in F} \lceil d^k \rceil x_e^k \tag{26}$$

where $I \subseteq \mathcal{N}$ and $F \subseteq \delta(V_1, \ldots, V_p)$. For $I = \mathcal{N}$ and $F = \emptyset$ the flow-cut or flow-partition inequality (26) reduces to the the cut-set or graph-partition inequality (23).

8 Computational Issues

To evaluate how strong the models and inequalities introduced in the previous sections are in practice, we implemented a cutting plane algorithm and some LP based rounding heuristics to solve the Steiner tree packing network design problem. In this section we describe the main components of our current implementation: the preprocessing of the initial problem instance, the separation routines for lifted cover inequalities in general and its application to single edge knapsack, partition-knapsack and flow-cut inequalities, and the rounding heuristics.

Before we describe how these components have been implemented we want to mention two points: First, the main focus of this study was to obtain good lower bounds to the optimum solution value. It was not our goal to solve the problem instances to optimality as fast as possible but to get an idea of how far the LP lower bound can be pushed using cutting planes to tighten the relaxation and without branching. The LP rounding heuristics are used only to get reasonable upper bounds for the optimal solution value.

Since our aim was to derive good lower bounds, we implemented only the two "strong" formulations: the directed cut formulation 3.4 and the multi-commodity flow formulation 3.5. The discussion below applies to both models, except for the separation of Steiner cut inequalities, which is not necessary for the multi-commodity flow formulation.

8.1 Initial Formulation

For both formulations 3.4 and 3.5 we start with the entire set of variables in the initial LP relaxation.

The directed cut formulation 3.4 contains the constraints (6), (7), and (8) in the initial formulation. The Steiner cut inequalities (5) are separated during the cutting plane phase of our algorithm. Only a very small subset of these inequalities, namely those for single vertices, is added to the initial formulation:

$$\sum_{a \in \delta^-(v)} x_a^k \geq 1 \qquad k \in \mathcal{N},\, v \in \bar{T}^k.$$

To strengthen the initial LP relaxation without adding more Steiner cut inequalities, we add some further inequalities which ensure that, if some arc $a_1 \in \delta^+(v)$ is in the spanner for terminal set T^k, $v \neq r^k$, then some arc $a_2 \in \delta^-(v)$ (with a_2 and a_1 not anti-parallel) is in the spanner, too:

$$\sum_{(w,v) \in \delta^-(v), w \neq u} x_{(w,v)}^k \geq x_{(v,u)}^k \qquad k \in \mathcal{N},\, v \notin T^k,\, (u,v) \in A.$$

In the multi-commodity flow formulation 3.5 all constraints (9) to (14) are contained in the initial formulation.

8.2 Preprocessing

For a terminal set T^k with a large demand value d^k it is sometimes clear a priori which edges cannot and which edges must be contained in the Steiner tree for T^k

in any optimal solution. In a preprocessing stage of our algorithm we try to fix as many of the corresponding tree variables x_e^k as possible.

For $k \in \mathcal{N}$ and $f \in E$ we denote

$$w(T^k \mid_{x_f^k=0}) := \min\{ \sum_{e \in S^k} w_e \mid \ S^k \text{ Steiner tree w.r.t. } T^k,\ f \notin S^k \ \},$$

and

$$w(T^k \mid_{x_f^k=1}) := \min\{ \sum_{e \in S^k} w_e \mid S^k \text{ Steiner tree w.r.t. } T^k,\ f \in S^k \}.$$

Observation 8.1 *Let $k \in \mathcal{N}$ and $f \in E$.*

(i) If $w(T^k \mid_{x_f^k=0}) \lceil d^k \rceil \leq w(T^k \mid_{x_f^k=1}) \lfloor d^k \rfloor$, then there is an optimal solution $(\hat{y}, \hat{x})$ for SPN with $\hat{x_f^k} = 0$.

(ii) If $w(T^k \mid_{x_f^k=1}) \lceil d^k \rceil \leq w(T^k \mid_{x_f^k=0}) \lfloor d^k \rfloor$, then there is an optimal solution $(\hat{y}, \hat{x})$ for SPN with $\hat{x_f^k} = 1$.

From this observation we derive a simple preprocessing algorithm: For each $k \in \mathcal{N}$ with $d^k \geq 1$ and $f \in E$ we try to compute the minimum cost values $w(T^k \mid_{x_f^k=1})$ and $w(T^k \mid_{x_f^k=0})$ of the Steiner trees for T^k containing f and not containing f, respectively. If 8.1(i) applies, we can fix $x_f^k = 0$, if 8.1(ii) applies, we can fix $x_f^k = 1$ already in the initial formulation.

In order to compute these minimum cost Steiner trees we solve the multi-commodity flow LP relaxation of the (single) Steiner tree problem for T^k with x_f^k fixed to 1 and 0, respectively. If both LP solutions (not only the optimal objective values) are integral we can apply both rules and, if the conditions are satisfied, fix some tree variables. If the solution of the Steiner tree LP is integral for $x_f^k = 0$ but fractional for $x_f^k = 1$, we still can apply rule 8.1(i) and maybe fix the tree variable x_f^k to zero. Analogously, if the Steiner tree LP has an integer solution for $x_f^k = 1$ but not for $x_f^k = 0$, rule 8.1(ii) can be applied. Clearly, another feasible approach would be to apply branch and bound to solve the single Steiner tree problems for T^k with x_f^k fixed to 1 and 0 to integrality. But in order to spend not too much time in the preprocessing we do not branch here.

8.3 Cutting Plane Algorithm

In the cutting plane phase of our algorithm we iteratively solve the current linear programming relaxation, apply our separation algorithms to find violated valid inequalities, and add these inequalities to the relaxation.

We maintain a global pool containing cut-sets and graph-partitions. Only cut-sets and graph-partitions from this pool are considered when we try to separate cut-set, graph-partition, cut-knapsack, partition-knapsack, or flow-cut inequalities. Initially, all cuts whose smaller shore does not contain more that a given number n_{max}^{cut} of vertices are enumerated. Analogously, for given numbers p_{max}^{part} and n_{max}^{part} all partitions into $p \leq p_{max}^{part}$ shores whose smaller $p-1$ shores do not contain more than v_{max}^{part} vertices are enumerated.

During the cutting plane phase, this pool is only modified if the directed cut formulation is used. In this case, we add all violated Steiner cuts which have been separated during the cutting plane phase to the pool. In a future implementation,

we also plan to generate cut-sets and graph-partitions during the cutting plane algorithm based on the dual values of the capacity constraints (6) or (11).

In each iteration the algorithm first (re-) solves the current LP relaxation and then applies one or more separation algorithms. These separation algorithms are combined into four modules for the following types of inequalities:

SEP1 Steiner cut inequalities,

SEP2 Strengthened metric inequalities,

SEP3 Lifted cover inequalities for the single edge capacity knapsack, cut-knapsack inequalities, and partition-knapsack inequalities, and

SEP4 Flow-cut and flow-partition inequalities.

These separation modules are used in a hierarchical manner: a module is only executed if the preceding modules did not find a user specified minimum number of violated inequalities. If the required number of violated inequalities was found the current iteration finishes and the LP is re-optimized. In our tests, we set this number to 20.

8.4 Steiner cuts

When working with the directed cut formulation 3.4, all violated directed Steiner cut inequalities (5) need to be separated. This can be done by solving a sequence of $|T^k| - 1$ maximum flow problems for each commodity $k \in \mathcal{N}$.

We use a special variant of this separation method which was described in [KM98] for the (single) Steiner tree problem: Let $(\bar{y}, \bar{x})$ be a fractional point. For some tiny value ϵ (we used $\epsilon = 10^-6$) we compute the minimal (r^k, t)-dicut for all $t \in \bar{T}^k$ with respect to the capacities $\bar{x^k} + \epsilon$. The advantage of adding this so-called creep-flow to the fractional Steiner tree variables for T^k is that we will obtain a violated Steiner cut which is not only most violated but also contains as few arcs as possible among all maximally violated Steiner cuts. In [KM98] is was shown that this creep-flow approach reduces the number of iterations and separated cuts by two to three orders of magnitude for some large Steiner tree problems. Furthermore, we use the cuts found here also to separate violated cut-knapsack and flow-cut inequalities. Hence, reducing the number of cuts that are considered there significantly speeds up our algorithm. Besides that, the partition-knapsack and flow-cut inequalities are weaker the more edges are in the partition or in the cut.

8.5 Strengthened Metric Inequalities

In order to separate violated cut-set and graph-partition inequalities we simply check these inequalities for all cut-sets and graph-partitions in the global pool.

8.6 Lifted Cover Inequalities

In this section we describe the heuristic we use to find violated lifted cover inequalities (15).

Remember that these inequalities have the form

$$y \geq \sum_{k \in I} \lceil d^k \rceil x^k + \lceil \sum_{k \in I} d^k \rceil - \sum_{k \in I} \lceil d^k \rceil$$

for some $I \subseteq \mathcal{N}$. A fractional point $(\bar{y}, \bar{x})$ violates such an inequality if and only if

$$0 > \min\{\sum_{k\in I} \lceil d^k \rceil (1 - \bar{x^k}) - \lceil \sum_{k\in I} d^k \rceil + \bar{y} \mid I \subseteq \mathcal{N}\}.$$

Since $\bar{x^k} \leq 1$, this can only be the case if $\lceil \sum_{k\in I'} d^k \rceil > \bar{y}$ for the set $I' \subseteq \mathcal{N}$ for which this minimum is attained. Let $c := \lceil \sum_{k\in I'} d^k \rceil - \bar{y}$. Then finding this set $I' \subseteq \mathcal{N}$ is equivalent to solving the knapsack problem

$$\min\{\sum_{k\in\mathcal{N}} (\lceil d^k \rceil (1 - \bar{x^k}))\mu^k \mid \sum_{k\in\mathcal{N}} d^k \mu^k > \lceil \bar{y} \rceil + c,\ \mu \in \{0,1\}^{\mathcal{N}}\}. \tag{27}$$

In this knapsack μ is the characteristic vector of the unknown set. If we can solve this problem and the value of the optimum solution μ' is negative, then the lifted cover inequality given by the set $I' := \{k \in \mathcal{N} \mid \mu'^k = 1\}$ is a maximally violated lifted cover inequality for the point $(\bar{y}, \bar{x})$.

For a given value c the knapsack problem (27) is solved heuristically using the standard greedy approach. We apply the greedy for four different orderings of the elements in $\mathcal{N}$:

- non-decreasing order of $(1 - \bar{x^k})\lceil d^k \rceil / d^k$,
- non-increasing order of $\bar{x^k} d^k$,
- non-increasing order of d^k, and
- non-decreasing order of $(1 - \bar{x^k})\lceil d^k \rceil$.

Since the value c is not known a priori, we run these greedy algorithms for each value $c = 0, \ldots, 3$, if necessary.

In each iteration of the cutting plane phase of our algorithm this separation heuristic is applied to the knapsack $\mathcal{P}_Y(e)$ defined in Section 5 for each edge e in the graph G.

8.7 Partition-knapsack and flow-cut inequalities

In order to separate partition-knapsack inequalities (25) for some graph-partition $\{V_1, \ldots, V_p\}$ (or a cut-set $\{V_1, V_2\}$) we apply the heuristic described for lifted cover inequalities to the knapsack $\mathcal{P}_Y(V_1, \ldots, V_p)$ defined in Section 7.1. Given a fractional point $(\bar{y}, \bar{x})$, the variables $\hat{x^k}$ are chosen from

$$\hat{x^k} := \begin{cases} \bar{x^k_{(u,v)}} + \bar{x^k_{(v,u)}} & \text{for } uv \in [V_i, V_k],\ V_i \cap T^k = \emptyset \text{ or } V_j \cap T^k = \emptyset, \\ \sum_{uv\in F^k} (\bar{x^k_{(u,v)}} + \bar{x^k_{(v,u)}}) - (|F^k| - 1) & \text{for } F^k \subseteq [V_i, V_j], \end{cases}$$

such that $\hat{x^k}$ is maximized.

For a given cut $\{V_1, V_2\}$ and $F \subset [V_1, V_2]$, violated flow-cut inequalities (26) are separated applying the above heuristic to the knapsack $\mathcal{P}_Y(V_1, V_2, F)$ defined in Section 7.1. Given a fractional point $(\bar{y}, \bar{x})$, the variables $\hat{x^k}$ are chosen to be maximal from

$$\hat{x^k} := \begin{cases} 1 - \sum_{uv\in F, u\in V_1, v\in V_2} \bar{x^k_{(u,v)}} & \text{if } r^k \in V_1 \text{ and } T^k \cap V_2 \neq \emptyset, \\ 1 - \sum_{uv\in F, u\in V_1, v\in V_2} \bar{x^k_{(v,u)}} & \text{if } r^k \in V_2 \text{ and } T^k \cap V_1 \neq \emptyset, \\ \max\{\bar{x^k_{(u,v)}} + \bar{x^k_{(v,u)}} \mid uv \in \delta(V_1) \setminus F\} & \text{otherwise.} \end{cases}$$

In the cutting plane algorithm, we apply these separation methods only for those cut-sets and graph-partitions in the pool which are tight. A cut-set or a graph-partition is tight if the slack of its associated metric inequality is not more than a specified percentage of its right hand side. In our implementation the default value for this ratio is 10%. When separating flow-cut inequalities, we simply try all possible subsets $F \subset \delta(V_1)$ for all tight cuts-set. Furthermore, for each cut-set and graph-partition in the global pool we keep track of how often it was used to separate a flow-cut or partition-knapsack inequality and how many of those attempts failed. If the number of unsuccessful attempts reaches a user specified bound, the separators will revoke this cut-set or graph-partition in future separation attempts.

8.8 Rounding Heuristics

The cutting plane algorithm finishes either if it cannot find further violated inequalities or if the improvement rate of the LP value drops below a specified threshold.

We implemented a set of rounding heuristics which then try to find a good primal (i.e. integer) solution, iteratively bounding or fixing some variables and re-applying the cutting plane algorithm to the restricted LP formulation. In each iteration these heuristics either bound some capacity variable y_e to be greater or equal to its current fractional value rounded up or or fix some tree variable x_e^k to one.

Let $(\bar{y}, \bar{x})$ be the current fractional solution. We implemented the following bounding and fixing strategies:

MinCostInc
: Set $y_f \geq \lceil \bar{y_f} \rceil$ where $f = argmin\{(\lceil \bar{y_e} \rceil - \bar{y_e})w_e \mid e \in E, \bar{y_e} \notin \mathbb{Z}\}$.

MinRelCostInc
: Set $y_f \geq \lceil \bar{y_f} \rceil$ where $f = argmax\{\bar{y_e}/\lceil \bar{y_e} \rceil \mid e \in E, \bar{y_e} \notin \mathbb{Z}\}$.

MinCapInc
: Set $y_f \geq \lceil \bar{y_f} \rceil$ where $f = argmin\{\lceil \bar{y_e} \rceil - \bar{y_e} \mid e \in E, \bar{y_e} \notin \mathbb{Z}\}$.

MaxFracTree
: Set $x_f^k = 1$ where $(f, k) = argmax\{\bar{x_e^k} \mid e \in E, k \in \mathcal{N}, \bar{x_e^k} \notin \mathbb{Z}\}$.

If one of the strategies MinCostInc, MinRelCostInc, or MinCapInc is used and there is no more fractional capacity variable $\bar{y_e}$, then the rounding heuristic switches to the MaxFracTree strategy to fix fractional tree variables. If a capacity variable becomes fractional again the heuristic switches back to the original strategy. If the MaxFracTree strategy is used, the heuristic need not switch to another strategy. The lifted cover inequalities that are separated in each iteration are strong enough to push all capacity variables up to next feasible integer value.

8.9 Results

Finally, we want to report on some computational experiments. Unfortunately, we were not able to get hold of reasonable real-life data of design problems with multicast traffic. We used several data sets of real-world unicast network design problems and aggregated the unicast commodities to multicast commodities. Whenever some unicast commodities' origins did match and their demand values were equal up to some specified tolerance, then those unicast commodities were joined to a multicast commodity with demand value equal to the average of the unicast commodities' demand values. Varying the allowed tolerance of the unicast demand values which

Instance	$\|V\|$	$\|E\|$	$\|\mathcal{N}\|$	avg. $\|T^k\|$
15-21	15	22	21	11.1
15-32	15	22	32	6.0
15-70	15	22	70	4.0
15-141	15	22	141	3.1
27-65	27	51	65	11.5
27-97	27	51	97	8.4
27-147	27	51	147	6.2
27-323	27	51	323	3.1

Table 1: Problem instances

Instance	$\lceil w(ILP) \rceil$	LB	UB	gap %
15-21	183,685	191,622	195,110	1.8
15-32	178,427	185,866	189,670	2.0
15-70	186,963	194,092	198,100	2.1
15-141	215,312	224,596	227,98	1.5
27-65	8,940	10,048	10,814	7.6
27-97	11,222	12,833	13,854	8.0
27-147	12,952	15,830	17,131	8.2
27-323	16,651	21,696	23,058	6.3

Table 2: Best lower and upper bounds computed

are aggregated, we are able to generate several practical-looking instances of the Steiner tree packing network design problem.

Based on two basic single-cast network design problems, we created a number of test instances on which we will report here. The first basic instance contains 15 nodes and 22 edges. Aggregating the single-cast commodities of this instance, we created four instances of the Steiner tree packing network design problem. The second basic instance contains 27 nodes and 51 edges and we aggregated the single-cast commodities with four different tolerances. The instances are summarized in Table 1.

These problem sizes might seem very small. But even for the smallest problem instance 15-21 CPLEX 6.5 cannot solve the final LP relaxation generated by our cutting plane algorithm to integrality within one week on a SUN Ultra10, neither if the directed cut formulation nor if the multi-commodity flow formulation is used.

In Table 2 we present the results for the directed cut formulation: the values of the initial LP relaxations, the best lower bounds, i.e., the LP values at the end of the cutting plane algorithms, and the best upper bounds obtained by the rounding heuristics. As we already mentioned in the paper, both the directed cut and the multi-commodity flow based formulation are theoretically equivalent in terms of the lower bound they provide. This also holds if we add all violated inequalities of the classes discussed in this paper. But, since most of these inequalities are separated heuristically, in practice these formulation can lead to slightly different results.

Non-surprisingly, in our tests we observed that with the multi-commodity flow formulation the cutting plane algorithm needs less iterations than with the directed cut formulation. On the other hand, re-solving the multi-commodity flow formulation takes much longer than re-solving the directed cut formulation. Even though we need more iterations with the directed cut formulation, the size of the LP remains small enough to be resolved efficiently. Furthermore, the Steiner cut separator,

Instance	DCUT			MCF		
	value	iter	time	value	iter	time
15-21	191,622	56	21	191,585	44	1:47
15-32	185,866	58	28	185,871	34	2:16
15-70	194,092	85	48	194,122	30	4:03
15-141	224,596	76	1:40	224,303	47	6:10
27-65	10,047	109	2:16	10,048	28	1:32:47
27-97	12,832	131	3:56	12,832	22	1:35:46
27-147	15,830	131	3:56	15,829	19	1:51:44
27-323	21,696	146	19:30	21,695	17	2:56:17

Table 3: Cutting plane values, iterations, and times (h:mm:ss)

which is used only for the directed cut formulation, usually finds many graph-cuts that are not in the initial cut pool. Having these graph-cuts in the pool may turn out to be extremely important, because they are used in the cut-knapsack and flow-cut separators, too. Table 3 shows the values and running times of the cutting plane algorithm for the two different formulations. The computations were done on an Intel Pentium 3, 800MHz machine with 768MB, running the Linux operating system. CPLEX 6.5 was used to solve linear programs.

In conclusion, we think that these tests support our opinion that the two formulations are appropriate to compute good lower bounds for the Steiner tree packing network design problem. With some simple separation heuristics that focus on one type of inequalities, the lifted cover inequalities and its variants as partition-knapsack and flow-cut inequalities, we were able to close a large fraction of the integrality gap. There is still quite some potential in using these inequalities more efficiently or other slightly stronger variants. Nevertheless, we think that it is more promising to study the polyhedra for smaller problem instances in order to derive new types of valid inequalities.

References

[Bea84] J. E. Beasley, *An algorithm for the Steiner problem in graphs*, Networks **14** (1984), 147 – 159.

[BGW96] B.Brockmüller, O. Günlük, and L. A. Wolsey, *Designing private line networks - polyhedral analysis and computation*, Tech. Report CORE 9647, Université Catholique de Louvain, Louvain-la-Neuve, Belgium, 1996.

[BP] M. Bern and P. Plassmann, *The steiner problem with edge lengths 1 and 2*, Inform. Process. Lett., vol. 32.

[BZ78] E. Balas and E. Zemel, *Facets of the knapsack polytope from minimal covers*, SIAM Journal on Applied Mathematics **34** (1978), 119 – 148.

[Cho92] Sunil Chopra, *Packing steiner trees in a graph*, unpublished report, 1992.

[CR94a] S. Chopra and M. R. Rao, *The Steiner tree problem I: Formulations, compositions and extension of facets*, Mathematical Programming **64** (1994), 209 – 229.

[CR94b] S. Chopra and M. R. Rao, *The Steiner tree problem II: Properties and classes of facets*, Mathematical Programming **64** (1994), 231 – 246.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, W. H. Freeman and Company, New York, 1979.

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

[GM93] M. X. Goemans and Y. Myung, *A catalog of Steiner tree formulations*, Networks **23** (1993), 19 – 28.

[GMW96a] Martin Grötschel, A. Martin, and R. Weismantel, *Packing Steiner trees: A cutting plane algorithm and computational results*, Mathematical Programming **72** (1996), 125 – 145.

[GMW96b] Martin Grötschel, A. Martin, and R. Weismantel, *Packing Steiner trees: Further facets*, European Journal on Combinatorics **17** (1996), 39 – 52.

[GMW96c] Martin Grötschel, A. Martin, and R. Weismantel, *Packing Steiner trees: Polyhedral investigations*, Mathematical Programming **72** (1996), 101 – 123.

[GMW96d] Martin Grötschel, A. Martin, and R. Weismantel, *Packing Steiner trees: Separation algorithms*, SIAM Journal on Discrete Mathematics **9** (1996), 233 – 257.

[Goe94] M. X. Goemans, *The Steiner tree polytope and related polyhedra*, Mathematical Programming **63** (1994), 157 – 182.

[KM98] T. Koch and A. Martin, *Solving Steiner tree problems in graphs to optimality*, Networks **32** (1998), 207 – 232.

[Mar92] A. Martin, *Packen von Steinerbäumen: Polyedrische studien und anwendungen*, Ph.D. thesis, Technische Universität Berlin, 1992.

[MMV95] T. L. Magnanti, P. Mirchandani, and R. Vachani, *Modeling and solving the two–facility capacitated network loading problem*, Operations Research **43** (1995), 142–157.

[MW99] H. Marchand and L. A. Wolsey, *The* $0-1$ *knapsack problem with a single continuous variable*, Mathematical Programming **85** (1999), no. 1, 15–34.

[Pap94] C. H. Papadimitriou, *Computational complexity*, Addison Wesley, 1994.

[vdL99] R. L. M. J. van de Leensel, *Models and algorithms for telecommunication network design*, Ph.D. thesis, Faculty of Economics and Business Administration, Maastricht University, 1999.

[vHKvdLS99] C. P. M. van Hoesel, A. M. C. A. Koster, R. L. M. J. van de Leensel, and M. W. P. Savelsbergh, *Polyhedral results for the edge capacity polytop*, Tech. report, Maastricht University, 1999.