UNIVERSITÄT DORTMUND

# A New ILP Formulation for 2-Root-Connected Prize-Collecting Steiner Networks (TR)

Markus Chimani
Maria Kandyba
Petra Mutzel

Algorithm Engineering Report
**TR07-1-001**
April 2007
ISSN 1864-4503


University of Dortmund
Department of Computer Science
Algorithm Engineering (LS 11)
44221 Dortmund / Germany
**http://ls11-www.cs.uni-dortmund.de/**

# A New ILP Formulation for 2-Root-Connected Prize-Collecting Steiner Networks (TR)

Markus Chimani, Maria Kandyba*, and Petra Mutzel**

Chair for Algorithm Engineering, Dep. of CS, University of Dortmund, Germany
{markus.chimani,maria.kandyba,petra.mutzel}@cs.uni-dortmund.de

**Technical Report TR07-1-001**
April 2007

**Abstract.** We consider the real-world problem of extending a given infrastructure network in order to connect new customers. By representing the infrastructure by a single root node, this problem can be formulated as a 2-root-connected prize-collecting Steiner network problem in which certain customer nodes require two node-disjoint paths to the root, and other customers only a simple path. Herein, we present a novel ILP approach to solve this problem to optimality based on directed cuts. This formulation becomes possible by exploiting a certain *orientability* of the given graph. To our knowledge, this is the first time that such an argument is used for a problem with node-disjointness constraints. We prove that this formulation is stronger than the well-known undirected cut approach. Our experiments show its efficiency over the other formulations presented for this problem, i.e., the undirected cut approach and a formulation based on multi-commodity flow.

## 1 Introduction

Extending already existing fiber-optics networks by connecting new customers is an important topic in the design of telecommunication networks. Thereby, we have an existing infrastructure network $I$, a set of potential new customers $C$ and a set of potential new route-segments for laying the fiber cables. As each new customer $v$ will generate a certain assessable profit $p(v) \in \mathbb{R}^+$ and each route-segment $e$ has a certain laying cost $c(v) \in \mathbb{R}^+$, the main task is to connect a subset of $C$ with $I$ such that the overall profit is maximized. In this paper we consider the real world problem [1], where some of the customers, if added to the network, require two node-disjoint connections to $I$ to increase reliability. We denote these customers with the set $C_2$, and the other customers with $C_1$.

By representing the infrastructure network by a single root node $r$ (for the details of such transformation see, e.g., [2]), we obtain a rooted Prize-Collecting Steiner Network problem where certain nodes are required to be (nodewise) 2-connected with the root. Formally, we are given an undirected graph $G = (V, E)$, a root node

* Supported by the German Research Foundation (DFG) through the Collaborative Research Center "Computational Intelligence" (SFB 531)
** Partially supported by the Austrian Research Promotion Agency (FFG) under grant 811378 (NetQuest project)

$r \in V$, a set of customer nodes $C = C_1 \dot\cup C_2 \subset V$, a prize function $p : V \to \mathbb{R}^+$, and a cost function $c : E \to \mathbb{R}^+$. Find a subgraph $N = (V_N, E_N)$ of $G$ with $r \in V_N$ which minimizes $\sum_{e \in E_N} c(e) - \sum_{v \in V_N} p(v)$ and satisfies the following connectivity property: for every node $v \in C_k \cap V_N$ ($k \in \{1, 2\}$), $N$ contains at least $k$ node-disjoint paths connecting $v$ to $r$.

We call such a problem a *2-Root-connected Prize-Collecting Steiner Network* problem (2RPCSN). If we require all customers to be included into the solution network, the resulting problem is called *2-Root-connected Steiner Network* problem (2RSN). Both 2RPCSN and 2RSN are NP-hard, as they contain the Steiner tree problem as a special case. While our paper centers on the investigation of 2RPCSN, all results clearly also hold for 2RSN. Furthermore, our approach can be used for the relaxed version where $C_2$ customers are only required to be 2-edge-connected with the root.

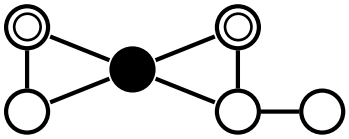
2RPCSN was already studied in [20, 21], where two different ILP formulations for this problem were suggested: one based on multi-commodity flow, similar to [13], the other one using undirected cut inequalities[1]. In this paper, we tranform 2RPCSN into the problem of finding an optimal subgraph in a related directed graph and give a new ILP formulation which uses directed cut inequalities. To our knowledge, our formulation is the first which applies such an approach to a node-disjoint connectivity problem, cf. Section 1.1. Furthermore, we study the polyhedral properties of our ILP and show that our formulation is stronger than the undirected cut formulation. We solve 2RPCSN using this new formulation within a Branch-and-Cut framework, utilizing an LP-based heuristic also presented herein. Our experimental results in Section 3 show that our approach is superior to those of [20, 21] for nearly all test instances.

### 1.1 Basics and Related Work

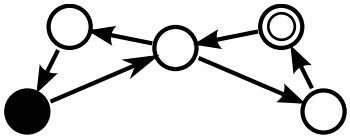
For a set $W \subset V$ of an undirected graph $G = (V, E)$ we denote the set of edges which separate $W$ from $V \setminus W$ by $\delta_G(W) = \{\{u, v\} \in E \mid u \in W, v \in V \setminus W\}$. For a directed graph $G'$, we distinguish between $\delta^-_{G'}(W)$ and $\delta^+_{G'}(W)$, i.e., the set of cut edges having a source or a target node in $W$, respectively. We may drop the index specifying the graph, if the graph is clear from the context.

2RSN is a special class of survivable network design problems (SNDP) [18]; see, e.g., [8, 22] for surveys. In [19] the following variant of SNDP is considered: each node $v$ of the given graph is associated with a nonnegative integer $r_v \in \{0, \dots, k\}$. For each pair of nodes $u, v \in V$ the connectivity requirement is then defined by $r_{uv} = \min\{r_u, r_v\}$, i.e., there should be at least $r_{uv}$ edge- or node-disjoint paths between those nodes. These problems are called $k$ECON and $k$NCON, respectively. In general, 2RSN and 2NCON are not equivalent, cf. Fig. 1(a).

[1] Although the paper's title uses the term "directed cut", it turns out to be equivalent to the traditional undirected approach discussed in Section 2.3.



(a) This network is infeasible for 2NCON, but feasible for 2RSN.



(b) This orientation satisfies the properties of Robbins' proof, but there are no two node-disjoint paths between the root and the $C_2$ customer.

**Fig. 1.** The root node is denoted by the black circle, $C_1$ and $C_2$ customers are denoted by simple and double circles, respectively.

For $k$ECON and $k$NCON, Grötschel, Monma, and Stoer [6] described integer linear programs and investigated their polyhedral structure. The central idea is to express the connectivity requirements by undirected cuts: for every non-empty set of nodes $W \subset V$ the number of edges in $\delta(W)$ should be at least $max\{r_{uv} \mid u \in W, v \in V \setminus W\}$. Wagner et al. [21] formulated their ILP for 2RSN and 2RPCSN using basically the same idea.

An *orientation* of an undirected graph $G$ is a directed graph $G'$ which is obtained by transforming each edge of $G$ into a directed edge. Robbins [17] (for the special case of $k = 1$) and Nash-Williams [15] showed that for any graph $G$ there exists an orientation $G'$ with the following property: for every pair of nodes $u, v$ which is $2k$-connected in $G$, there exist $k$ pairwise edge-disjoint directed paths $(u \to v)$ and $k$ pairwise edge-disjoint directed paths $(v \to u)$ in $G'$.

This fact has been exploited by Chopra [4] for solving 2ECON via directed graphs, who proved his formulation to be superior to the undirected formulation. Goemans [5] and Stoer [19] extended this formulation to $k$ECON for the case that all connectivity requirements are 0, 1, or even; later Magnanti and Raghavan [13] extended it for general $k$. It has been an open problem [19] if a similar orientation technique can be used for $k$NCON-type problems, i.e., when we require node-disjointness. Considering 2RSN and 2RPCSN, where we require nodewise 2-connectedness with a special root node, we will show that this is indeed the case.

Note that the above described approaches considered Steiner networks, and did not consider the prize-collecting variants. If $C_2 = \emptyset$, 2RPCSN becomes a Rooted Prize-Collecting Steiner Tree (RPCST) problem which has been investigated, e.g., by [10, 11]. Therein, the authors presented several ILP models, including a directed cut approach, which turned out to be the most successful one.

## 2 Investigating 2RPCSN

### 2.1 Transformation into a directed problem

The central idea of our formulation is that we can transform the undirected 2RPCSN problem $(G = (V, E), r, c, p)$ into a directed variant $(G' = (V, A), r, c', p)$ as follows: for each edge $\{u, v\} \in E$ there are two directed edges $(u, v)$ and $(v, u)$ in $A$, with $c'((u, v)) = c'((v, u)) = c(\{u, v\})$. An optimal solution of this directed problem

(D2RPCSN) is a subgraph $D = (V_D \subseteq V, A_D \subseteq A)$ with $r \in V_D$, which minimizes $\sum_{e \in A_D} c'(e) - \sum_{v \in V_D} p(v)$ and satisfies:

(D1) For each edge $\{u, v\} \in E$, $A_D$ may include at most one of the arcs $(u, v)$ and $(v, u)$.
(D2) For each customer node $v \in C_1 \cap V_D$ there is a directed path $(r \to v)$ in $D$.
(D3) For each customer node $v \in C_2 \cap V_D$ there is a path $(r \to v)$ and a path $(v \to r)$ in $D$, which are node-disjoint.

We show that D2RPCSN is equivalent to 2RPCSN. Clearly, every feasible solution of D2RPCSN can be interpreted as a feasible solution of 2RPCSN with the same objective value straightforwardly. Hence we have to focus on the reverse transformation from 2RPCSN to D2RPCSN. As we can see for the single $C_2$ node in Figure 1(b), the existence of node-disjoint paths does not follow from the orientability theorem [17] exploited by Chopra [4]. However note that, in this example, a reorientation of the 3-cycle containing this customer would result in a valid orientation for D2RPCSN.

**Theorem 1.** *Any optimal solution for 2RPCSN can be transformed into a corresponding feasible solution for D2RPCSN with the same objective value.*

*Proof.* Let $N = (V_N, E_N)$ be an optimal solution for 2RPCSN. We will show that there exists an orientation $D$ of $N$ which is a feasible solution for the corresponding D2RPCSN problem.

We first shrink $N$ by orienting all *attached trees*, i.e., we iteratively find an edge $\{u, v\}$ where $v$ has degree 1; we orient the edge from $u$ to $v$, and remove it temporarily. The remaining graph structure of undirected edges consists of one or more at least 2-connected components attached to $r$. It is clear that by orienting each one of them separately, we obtain a valid orientation for the complete structure. Hence we can restrict ourselves to a 2-connected undirected graph $B = (V_B, E_B)$ which contains $r$.

We use $\ell : V_B \to \mathbb{R} \cup \{\textit{undefined}\}$ as a labeling function; initially we have $\ell(v) := \textit{undefined}$ for all $v \in V_B$. We start by identifying a simple cycle $Z$ in $B$ containing $r$, and orient its edges consistently in one of the two possible directions. We then label each node on $Z$ with increasing fractional numbers between 0 and 1, according to this orientation, starting with $\ell(r) := 0$. Hence, all edges of $Z$ (except its last edge $\hat{e}$) are oriented from the smaller towards the larger label number. We will now orient the remaining undirected edges in such a way that this invariant is valid for all oriented edges:

We define an *augmenting path* $P = (a \to b)$ as a simple path of unoriented edges where only the disjoint start and end nodes are labeled, and $\ell(a) < \ell(b)$.

To orient $B$, we repeatedly find an augmenting path $P = (a \to b)$ and orient it from $a$ to $b$, labeling all inner nodes with increasing numbers greater than $\ell(a)$ but smaller than $\ell(b)$; these labels are to be unique over all labelings so far. By this construction, we guarantee that each labeled node has at least one incoming and one

outgoing edge. Furthermore, each oriented edge is oriented from the smaller towards the larger label number. Hence, each oriented path will always contain monotonously increasing label numbers (with the exception of $\hat{e}$). This means that any directed circle starting from $r$ and going through any labeled node $v$ will be simple, and we therefore have node-disjoint paths $(r \to v)$ and $(v \to r)$.

It remains to show that every edge gets oriented by this process. Assume that at some point there is at least one unoriented edge $e$ left, but we cannot find any augmenting path. Clearly, $e$ has to be part of some shortest path $Q = (c \to d)$ of unoriented edges with labeled nodes $c$ and $d$. Since neither $Q$ nor its reversal is an augmenting path, we have $\ell(c) = \ell(d)$ and therefore $c = d$, i.e., $Q$ is a cycle of unoriented edges, and none of its nodes except for $c$ are labeled. Since $B$ is 2-connected, there has to be an additional unoriented path from some node $q \in Q$ to some labeled node $p$ ($p, q \neq c$). But then, the path $(p \to q \to c)$ (or its reversal) would be an augmenting path, which is a contradiction. Hence, the above algorithm correctly orients any optimal solution of 2RPCSN. □

### 2.2 ILP for D2RPCSN

To model D2RPCSN on $G'$ we introduce two sets of binary variables

$$x_e, y_v \in \{0, 1\} \quad \forall e \in A, \forall v \in V.$$

The variables are 1, if the corresponding node or edge is in the solution network $D$, and 0 otherwise. We therefore obtain the objective funtion:

$$\min \sum_{e \in A} c(e) \cdot x_e - \sum_{v \in V} p(v) \cdot y_v. \tag{1}$$

In a feasible solution of D2RPCSN, at most one of the edges corresponding to an undirected edge $\{u, v\}$ can be selected. Furthermore, selecting an edge requires both incident nodes to be selected as well:

$$x_{uv} + x_{vu} \leq y_v \qquad \forall v \in V, \forall (v, u) \in A. \tag{2}$$

The *forward-cut* constraints are traditional cut-constraints requiring the selected customer nodes to be reachable from the root.

$$\sum_{e \in \delta^+(S)} x_e \geq y_v \qquad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap C. \tag{3}$$

For customers requiring 2-connectedness we have analogous *backward-cut* constraints:

$$\sum_{e \in \delta^-(S)} x_e \geq y_v \qquad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap C_2. \tag{4}$$

Finally, we have to assure that the 2-connected customer nodes are connected via node-disjoint forward- and backward-paths, i.e., we require each node $w$ to be part of at most one of these paths. This is equivalent to require that at least one of these paths does not contain $w$. Let $G'_w$ denote the graph $G'$ without the node $w$ and its incident edges. Then we have $\forall S_1, S_2 \subseteq V \setminus \{r\}, \forall v \in S_1 \cap S_2 \cap C_2, \forall w \in V \setminus \{r, v\}$:

$$\sum_{e \in \delta^+_{G'_w}(S_1)} x_e + \sum_{e \in \delta^-_{G'_w}(S_2)} x_e \geq y_v. \tag{5}$$

### 2.3 Polyhedral Comparison

We compare our ILP formulation to the common and straightforward formulation of 2RPCSN based on the undirected graph and undirected cuts. This formulation was, e.g., used in [21], although in a slightly more redundant form. Thereby, we have the characteristic vector $z \in \{0,1\}^{|E|}$ specifying the selected edges, and the vector $y$ analogous to the definition in Section 2.2. We then have:

$$\min \sum_{e \in E} c(e) \cdot z_e - \sum_{v \in V} p(v) \cdot y_v \tag{6}$$

$$\sum_{e \in \delta(S)} z_e \geq y_v \qquad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap C_1 \tag{7}$$

$$\sum_{e \in \delta(S)} z_e \geq 2y_v \qquad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap C_2 \tag{8}$$

$$\sum_{e \in \delta_{G_w}(S)} z_e \geq y_v \qquad \forall S \subseteq V \setminus \{r\}, \forall v \in S \cap C_2, \forall w \in V \setminus \{r, v\} \tag{9}$$

$$z_e, y_v \in \{0,1\} \qquad \forall e \in E, \forall v \in V \tag{10}$$

Let us consider any 2RPCSN problem and its corresponding D2RPCSN counterpart. For corresponding solutions, we clearly have the projection $z_{uv} = x_{uv} + x_{vu}$.

Let $\mathcal{P}_U$ and $\mathcal{P}_D$ be the polyhedrons corresponding to feasible LP relaxations, i.e., feasible solutions for the ILP without integrality constraints, for 2RPCSN and D2RPCSN, respectively. We show that $\mathcal{P}_D \subset \mathcal{P}_U$, i.e., the lower bounds obtained by the LP relaxations of our new formulation will in general be tighter than for the undirected formulation. The proof technique is based on [4, 6], but had to be extended for the prize-collecting setting.

**Observation 1** $\mathcal{P}_D \neq \mathcal{P}_U$.

*Proof.* Consider a triangle graph with the root node and two high-profit $C_1$ customer nodes. For 2RPCSN the fractional solution of 0.5 on all edges will satisfy all edge constraints. For D2RPCSN any solution corresponding to this undirected solution would be infeasible. □

**Theorem 2.** *The directed cut formulation is stronger than the undirected cut formulation. I.e.,* $\mathcal{P}_D \subset \mathcal{P}_U$.

*Proof.* Due to Observation 1, it is enough to show $\mathcal{P}_D \subseteq \mathcal{P}_U$. Hence, we have to show that the undirected cut inequalities can be generated from their directed counterparts, based on $z_{uv} = x_{uv} + x_{vu}$.
Consider any set $S \subseteq V \setminus \{r\}$. For $v \in C_2 \cap S$ we have:

$$\sum_{(u,v)\in\delta^+(S)} x_{uv} + \sum_{(v,u)\in\delta^-(S)} x_{vu} = \sum_{\{u,v\}\in\delta(S)} x_{uv} + x_{vu} = \sum_{\{u,v\}\in\delta(S)} z_{uv} \geq 2y_v.$$

For $v \in C_1 \cap S$ we have:

$$\sum_{\{u,v\}\in\delta(S)} z_{uv} = \sum_{\{u,v\}\in\delta(S)} x_{uv} + x_{vu} = \sum_{(u,v)\in\delta^+(S)} x_{uv} + \sum_{(v,u)\in\delta^-(S)} x_{vu} \geq y_v.$$

Analogously, we generate the undirected node-disjointness inequalities for any node $v \in S \cap C_2$ and $w \in V \setminus \{r, v\}$:

$$\sum_{\substack{\{u,v\}\\ \in\delta_{G_w}(S)}} z_{uv} = \sum_{\substack{\{u,v\}\\ \in\delta_{G_w}(S)}} x_{uv} + x_{vu} = \sum_{\substack{(u,v)\\ \in\delta^+_{G'_w}(S)}} x_{uv} + \sum_{\substack{(v,u)\\ \in\delta^-_{G'_w}(S)}} x_{vu} \geq y_v. \quad \square$$

### 2.4 Polynomial Separation and Branch-and-Cut

Based on our D2RPCSN ILP formulation, we developed a Branch-and-Cut code. For a general description of the Branch-and-Cut scheme see, e.g., [23]. Generally, such algorithms start with solving the *LP relaxation*, i.e., the ILP without the integrality property, only considering a certain subset of all constraints. Given the *fractional solution* of this partial LP, we perform a *separation routine*, i.e., identify constraints of the full constraint set which the current solution violates. We then add these constraints to our current LP and reiterate these steps. If at some point we cannot find any violated constraints, we have to resort to *branching*, i.e., we generate two disjoint subproblems, e.g., by fixing a variable to 0 or 1. By using the LP relaxation as a lower bound, and our heuristic solution (cf. Section 2.5) as an upper bound, we can prune irrelevant subproblems.

In our case, we start with the constraints (2) and the subset of the constraints (3) for $|S| = 1$. In the optimal solution, the root is the only node which may have only outgoing but no incoming edges. Analogously, no node, except for $C_1$ customers, will

ever have only incoming edges. Hence, we can use *flow-preservation* constraints:

$$x_{vu} \leq \sum_{\substack{(w,v)\in A \\ w\neq u}} x_{wv} \qquad \forall (v,u) \in A, v \neq r \tag{11}$$

$$x_{uv} \leq \sum_{\substack{(v,w)\in A \\ w\neq u}} x_{vw} \qquad \forall (u,v) \in A, v \notin C_1 \tag{12}$$

These constraints do not affect the strength of the formulation, as similar constraints do for the Steiner tree problem [9]. But adding all of these constraints to the initial constraint set, can help to increase the efficiency of our Branch-and-Cut approach. In our experiments we added all constraints (12) for the *ClgM* and *ClgM*$^+$ instances, cf. Section 3.

The cut constraints (3) can be separated in polynomial time via the traditional max-flow separation scheme: after obtaining some LP relaxation for our partial ILP, we compute the maximum flow from $r$ to each $v \in C$ in $G$ using the edge values of the current solution as capacities. If the resulting value is less then $y_v$, we extract one or more of the induced minimum $r$-$v$-cuts and add the corresponding constraint(s) to our ILP model. The cut constraints (4) can be separated analogously.

If there are no violated constraints of type (3) or (4), we solve the separation problem for the constraints of type (5) in an analogous way: for each node $v \in C_2$ and for each node $w \in V$, $w \neq v$ we compute both the $v$-$r$ and $r$-$v$ maximal flows in $G'_w$. If the sum of these flows is less than $y_v$, we add the corresponding inequalities. Actually, we do not need to perform the separation routine for each node $w$: let us consider an integer solution where the constraints (3) and (4) are valid, i.e., we have edge-disjoint paths $(r \to v)$ and $(v \to r)$ for any $v \in C_2$. Assume these paths have a common node $w$, then there are at least two incoming and two outgoing edges at $w$. More general, this means that in our fractional solution, we have to consider only nodes $w$ satisfying $\sum_{e\in\delta^-(w)} x_e > 1$ and $\sum_{e\in\delta^+(w)} x_e > 1$.

Hence all constraints can be separated in polynomial time and, unless $\mathcal{P} = \mathcal{NP}$, we cannot assume that the LP-relaxation will always be integer. Therefore, if the solution is fractional and there are no separable cuts, we have to revert to branching techniques.

### 2.5 Primal Heuristic

A fractional solution of the LP relaxation is used to construct a feasible solution, thus obtaining upper bounds for the optimal solution. Our heuristic first finds a feasible solution $T$ of the RPCST problem, which we obtain by interpreting all customers as $C_1$ customers. This solution is then extended to a feasible solution $S$ of 2RPCSN by adding additional paths for the $C_2$ customers of $T$. Finally, we shrink $S$ by deleting some nodes and edges from it without losing feasiblity.

*Construct $T$.* We construct $T$ using the LP-based heuristic by Ljubic [10]: according to the $y$-values of the fractional solution, the algorithm first computes the set of customer nodes to be included in $T$. Using the edge costs $1 - x_e$ for $e \in E$, we construct a Steiner tree $T$ with the Steiner tree heuristic by Mehlhorn [14].

*Assure 2-connectivity.* The idea is to successively extend $T$ by adding shortest $(v \rightarrow r)$-paths for the customer nodes $v \in C_2$ for which the 2-connectivity requirement is not satisfied yet. Let $v$ be such a node and let $P$ be the unique $(r \rightarrow v)$-path in the original $T$. To find the missing backward-path from $v$ to $r$, we apply Dijkstra's algorithm on $G \setminus P$; the costs of the edges already in the solution are 0, for the other edges we use the given cost function $c$. Clearly, we have to perform this operations only for nodes $v \in C_2$ which do not contain any further $C_2$ customers in their subtrees of $T$: 2-connecting $v$ results in proper 2-connectedness for all nodes $w \in P$. In the resulting subgraph $S$, all connectivity requirements are satisfied.

*Shrinking.* In general, $S$ can be further optimized by deleting some nodes and edges from $S$ without losing feasiblity. We know that $S$ consists of one or more non-trivial 2-connected components, which have only the root node in common. All other components of the graph form trees, which are attached to some 2-connected component. Having this decomposition in mind we can optimize $S$ in two steps:

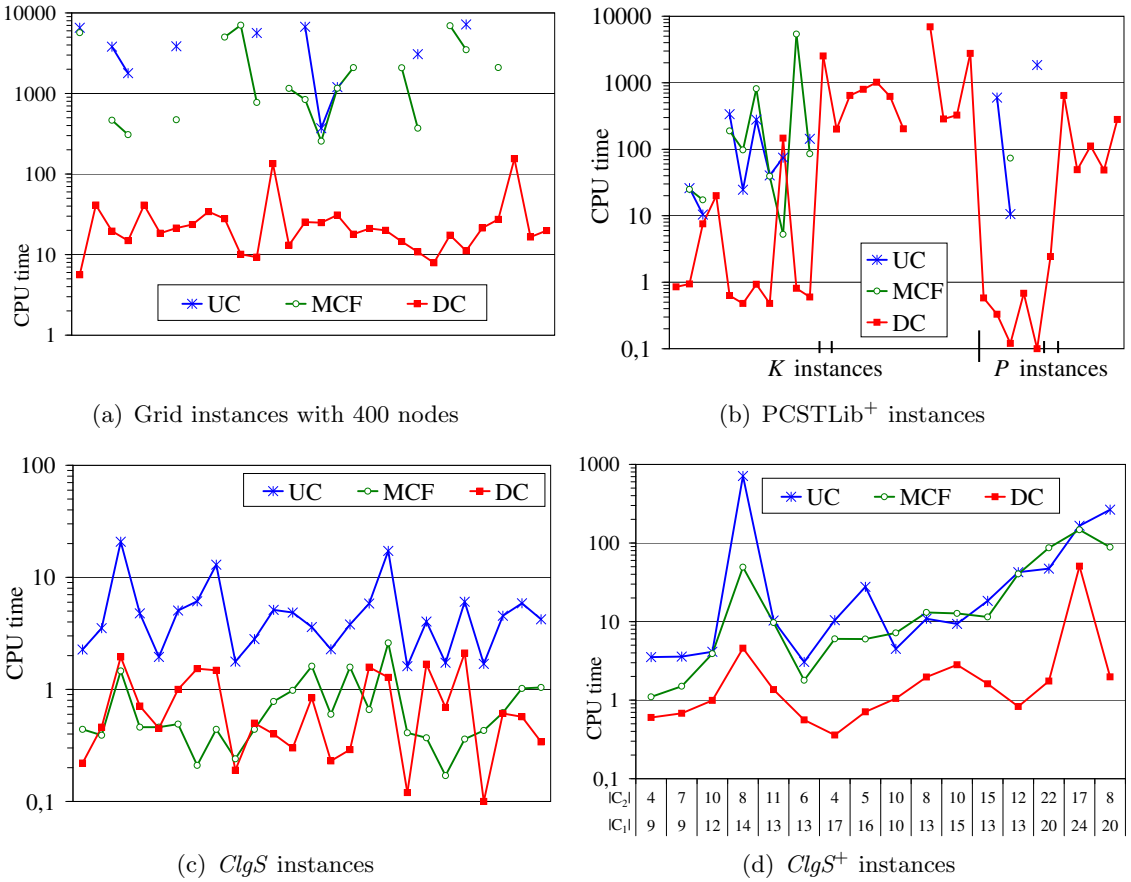
As described in [23], the rooted PCST problem can be solved in linear time, when applied to trees. We use this algorithm to optimize all attached trees, using the attachment node as its root. For the next step, these root nodes will be considered to be $C_1$ customers with corresponding prizes.

For every block $B$ of $S$ we compute its *core graph* $\tilde{B}$. Thereby, every chain of edges only containing nodes $v \in V \setminus (C_2 \cup \{r\})$ are replaced by a single edge. We remove an edge $e$ from $\tilde{B}$ if all connectivity requirements are still satisfied for $\tilde{B} - e$. The corresponding path $P_e$ is also removed from the original subgraph $B$ if no inner node of $P_e$ is a customer node. Otherwise, we can determine in linear time whether it is worth to include such nodes into the solution and, if this is the case, which edges of $P_e$ are superfluous for the 1-connectedness of these.

*Use within Branch-and-Cut.* We run this heuristic after every 10th computation of an LP-relaxation. Furthermore, we use the heuristic to generate an initial solution, choosing $y_v := 1$ for all $v \in C$ and using $c(.)$ as edge costs for the initial Steiner tree $T$.

## 3 Experiments

We implemented our Branch-and-Cut algorithm in C++, using CPLEX 9.0 and LEDA 5.0.1. The tests were run on a 2.4 GHz AMD Opteron with 2GB of RAM per process. For the experiments we used three different sets of instances presented in [1],




(a) Grid instances with 400 nodes

(b) PCSTLib$^+$ instances

(c) *ClgS* instances

(d) *ClgS*$^+$ instances

**Fig. 2.** Diagrams comparing the undirected cut (UC) and the multi-commodity flow (MFC) approach to our directed cut (DC) formulation.

and compared the results of our directed cut (DC) approach with those of the multi-commodity flow (MCF) and undirected cut (UC) formulations, which were partially published in [20, 21][2]. As it was the case in these publications, we applied a time limit of 2 hours per problem instance. See Figure 2 for diagrams corresponding to the experiments described below: generally, the vertical axis shows the required CPU time in seconds on a logarithmic scale, whereas the horizontal axis corresponds to the instances (in lexicographical order). When an instance could not be solved to provable optimality, there is no corresponding data point for the according formulation.

*Grid Instances.* We used the artificial grid instances with 100, 400, 900, 1600, 2500 nodes [1]. There are two different infrastructure layings $I$ per graph size; for each such $I$, there are 15 instances with different sets of customer nodes. The instances have

[2] The algorithms by Wagner et al. were run by Wagner on a stronger Intel Xeon 3.6GHz with CPLEX 10.0.1 and LEDA 5.1.

| # nodes | 100 | 400 | 900 | 1600 | 2500 |
|---|---|---|---|---|---|
| **UC** | 9.36/41.3 | 3834/4021 (33.3%) | (0%) | (0%) | (0%) |
| **MCF** | 5.15/25.6 | 1161/2638 (56.7%) | (0%) | (0%) | (0%) |
| **DC** | 0.10/0.25 | 20.0/28.6 | 214/423 | 1615/1840 | 1856/2238 (73.3%) |

**Table 1.** Median/average CPU time for the grid instances. The percentage of successful instances is given in brackets if not 100%.

5–13 $C_1$ and 3–8 $C_2$ customers. Our algorithm is able to solve all of these instances to optimality, except for 8 instances with 2500 nodes. The largest instances solvable by the previous approaches contained 400 nodes and the running times are much longer, cf. Table 1 and Figure 2(a). Interestingly, our approach required no branching for the instances with 900–2500 nodes.

*PCSTLib*$^+$. This set of instances is based on the instances of PCSTLib [7], also used, e.g., in [3, 10, 12], and were extended in [1] for 2RPCSN. We used the instances of the groups $K$ and $P$; each contains graphs with 100, 200, and 400 nodes. The set $K$ consists of random geometric instances which were designed to have a structure similar to street maps. While, for the $K$ instances, 15%–27% percent of the nodes are customers, the $P$ instances have 34%–50% customers; in all instances there are roughly twice as many $C_1$ nodes as $C_2$ nodes. As shown in Figure 2(b), we solve all $P$ and $K$ instances to optimality, except for a single $K$ instance with 400 nodes. MCF can solve most $K$ instances with 100 nodes, but no larger instances. It solves only a single 100 node $P$ instance, which seems to be due to the high number of customers. This is the only case, where UC is comparable and sometimes even stronger than MCF, but still it is much weaker than DC. These modified PCSTLib instances are the only ones, where DC regularly has to branch, requiring 24 branch nodes on average (the median is 2).

*Cologne Instances.* These instances use the real-world access net data of the city district Cologne-Ossendorf. For our experiments we took the small (*ClgS*) and medium (*ClgM*) sets of instances [1]: 20 instances with 190 nodes and 377 edges, and 25 instances with 1757 nodes and 3877 edges, respectively. Since these instances have a quite small number of customers (3–6 $C_1$ and 2–3 $C_2$ customers), we generated additional sets of instances for both sizes by choosing additional customers, resulting in 16 ClgS$^+$ and 6 ClgM$^+$ instances.

Figure 2(c) gives the results for *ClgS* and shows that MCF and DC are competitive and both clearly outperform the undirected cut approach. As soon as there are more customers, as it is the case for *ClgS*$^+$ depicted in Figure 2(d), DC is again superior to two both other approaches.

For the *ClgM* instances with very few customers, MCF is stronger than the others, being able to solve 14 instances, whereas DC solves 9. DC only solves 5 when we run it without the flow-preservation constraints, and UC does not solve

any instance. This seems to be due to the fact that in these instances the solutions contain very long paths between the customer nodes, giving the flow formulation an advantage over the cut approaches. This is supported by the observed gain by using the flow-preservation constraints. When there are more customers, which is the case for *ClgM*$^+$, the average path length is reduced and DC becomes dominant again: while MCF and UC are unable to solve any instance, DC can solve 1. Furthermore, after two hours, DC obtains better lower bounds than MCF and UC in 4 out of 6 cases, and always has a better feasible solution.

## 4 Additional Remarks

Even without our Theorem 1, we can improve the multi-commodity flow approach presented by Wagner et al. [20] and Raghavan's approach for a related problem where at most two-connectedness is required (cf. page 188 of [16]): We know that 2-connectedness between two nodes $r$ and $v$ is equivalent to having a simple cycle through $r$ and $v$. Instead of introducing two commodities, and sending either both of them from $r$ to $v$ ([20]), or one from $r$ to $v$ and the other backwards from $v$ to $r$ ([16]), the ILP can use one commodity per pair $r, v$ and compute a cyclic flow through both of them. The resulting ILP is intuitively equivalent to the original variant, but the number of variables is greatly decreased.

In [20, 21], the 2RPCSN problem is also considered with a special relaxation for the $C_2$ customers: these customers are not required to be 2-connected with the root, but to be *near* to a node with such a property. Therefore, each edge $e$ has a certain *distance* $d(e)$, and each selected $C_2$ customer $v$ is allowed to have a *connection path* of length up to $k(v)$, if it itself is not 2-connected. Analogously to [21], we can add these constraints to our ILP, using additional $y'$ variables and backward-cuts to decide whether a (non-$C_2$) node is 2-connected. Anyhow, we recommend to only start with the constraints

$$\sum_{w \in N(v)} y'(w) \geq y(v) \qquad \forall v \in C_2 \tag{13}$$

where $N(v)$ is the set of nodes in the neighborhood of $v$, i.e., there exists a path of length at most $k(v)$ to each of them in $G$. Although this constraint is not sufficient, it allows us to introduce a column-generation scheme, where the stricter constraints and corresponding variables for a $C_2$ node are only introduced if the above constraint does not lead to a short enough connection path.

**Acknowledgments.** We are deeply indepted to Ivana Ljubic for gratefully allowing us to use her PCST Branch-and-Cut code [10] as a basis of our implementation and for helpful discussions. We also thank Daniel Wagner for conducting additional

experiments with his undirected cut and multi-commodity flow implementations [20, 21] for our comparison, and the anonymous reviewer for a hint to simplify our orientability proof.

## References

1. P. Bachhiesl. The OPT- and the SST-problems for real world access network design – basic definitions and test instances. Working Report NetQuest 01/2005, Carinthia Tech Institute, Klagenfurt, Austria, 2005.
2. P. Bachhiesl. The OPT- standard problem, solvers, and results. Working Report NetQuest 02/2005, Carinthia Tech Institute, Klagenfurt, Austria, 2005.
3. S. A. Canuto, M. G. C. Resende, and C. C. Ribeiro. Local search with perturbations for the prize-collecting steiner tree problem in graphs. *Networks*, 38(1):50–58, 2001.
4. S. Chopra. The equivalent subgraph and directed cut polyhedra on series-parallel graphs. *SIAM J. Discrete Math.*, 5(4):475–490, 1992.
5. M. X. Goemans. *Analysis of linear programming relaxations for a class of connectivity problems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1990.
6. M. Grötschel, C. L. Monma, and M. Stoer. Polyhedral Approaches to Network Survivability. In *Reliability of Computer and Communication Networks, Proc. Workshop 1989*, volume 5 of *Discrete Mathematics and Theoretical Computer Science*, pages 121–141. American Mathematical Society, 1991.
7. D. S. Johnson, M. Minkoff, and S. Phillips. The prize-collecting steiner tree problem: Theory and practice. In *Proceedings of 11th ACM-SIAM Symposium on Distcrete Algorithms*, pages 760–769, San Fransisco, CA, 2000.
8. H. Kerivin and A. R. Mahjoub. Design of survivable networks: A survey. *Networks*, 46(1):1–21, 2005.
9. T. Koch and A. Martin. Solving steiner tree problems in graphs to optimality. *Networks*, 32(3):207–232, 1998.
10. I. Ljubic. *Exact and Memetic Algorithms for Two Network Design Problems*. PhD thesis, Technische Universität Wien, 2004.
11. I. Ljubic, R. Weiskircher, U. Pferschy, G. Klau, P. Mutzel, and M. Fischetti. An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. *Mathematical Programming, Series B*, 105(2–3):427–449, 2006.
12. A. Lucena and M. G. C. Resende. Strong lower bounds for the prize-collecting steiner problem in graphs. *Discrete Applied Mathematics*, 141(1-3):277–294, 2003.
13. T. L. Magnanti and S. Raghavan. Strong formulations for network design problems with connectivity requirements. *Networks*, 45(2):61–79, 2005.
14. K. Mehlhorn. A faster approximation for the steiner problem in graphs. *Information Processing Letters*, 27:125–128, 1988.
15. C. Nash-Williams. On orientations, connectivity and odd-vertex pairings in finite graphs. *Canad. J. Math.*, 12:555–567, 1960.
16. S. Raghavan. *Formulations and Algorithms for the Network Design Problems with Connectivity Requirements*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.
17. H.E. Robbins. A theorem on graphs with an application to a problem of traffic control. *American Mathematical Monthly*, 46:281–283, 1939.
18. K. Steiglitz, P. Weigner, and D. J. Kleitman. The design of minimum-cost survivable networks. *IEEE Trans Circuit Theory*, 16:455–460, 1969.
19. M. Stoer. *Design of Survivable Networks*. Springer-Verlag, 1992. volume 1531 of Lecture Notes in Mathematics.
20. D. Wagner, G. R. Raidl, U. Pferschy, P. Mutzel, and P. Bachhiesl. A multi-commodity flow approach for the design of the last mile in real-world fiber optic networks. In *Operations Research Proceedings 2006*. Springer-Verlag, 2006.
21. D. Wagner, G. R. Raidl, U. Pferschy, P. Mutzel, and P. Bachhiesl. A directed cut for the design of the last mile in real-world fiber optic networks. In *Proceedings of the International Network Optimization Conference 2007*, 2007.

22. P. Winter. Steiner problem in networks: A survey. *Networks*, 17(2):129–167, 1987.
23. L. A. Wolsey. *Integer Programming*. Wiley-Interscience, 1998.