On Stabilized Branch-and-Price for Constrained Tree Problems

Markus Leitner and Mario Ruthmair and Günther R. Raidl

Forschungsbericht / Technical Report

TR–186–1–11–01

September 30, 2011

On Stabilized Branch-and-Price for Constrained Tree Problems

Markus Leitner Mario Ruthmair Günther R. Raidl

Institute of Computer Graphics and Algorithms

Vienna University of Technology, Austria

{leitner,ruthmair,raidl}@ads.tuwien.ac.at

Abstract

We consider a rather generic class of network design problems in which a set or subset of given terminal nodes must be connected to a dedicated root node by simple paths and a variety of resource and/or quality of service constraints must be respected. These extensions of the classical Steiner tree problem on a graph can be well modeled by a path formulation in which individual variables are used for all feasible paths. To solve this formulation in practice, branch-and-price is used. It turns out, however, that a naive implementation of column generation suffers strongly from certain degeneracies of the pricing subproblem, leading to excessive running times. After analyzing these computational problems, we propose two methods for stabilizing column generation by using alternative dual-optimal solutions. This stabilized branch-and-price is practically tested on the rooted delay-constrained Steiner tree problem and a quota-constrained version of it. Results indicate that the new stabilization methods in general speed up the solution process dramatically, far more than a piecewise linear stabilization to which we compare. Furthermore, our stabilized branch-and-price exhibits on most test instances a better performance than a so far leading mixed integer programming approach based on a layered graph model and branch-and-cut. As the new stabilization technique utilizing alternative dual-optimal solutions is generic in the sense that it easily adapts to the inclusion of a large variety of further constraints and different objective functions, the proposed method is highly promising for a large class of network design problems.

Keywords: branch-and-price, integer linear programming, network design, stabilized column generation, Steiner tree

1 Introduction

Network design problems in which a dedicated root node must be connected to a given set or subset of terminal nodes, possibly via optional intermediate nodes and respecting diverse resource and quality-of-service (QoS) constraints, represent a highly important class of combinatorial optimization problems. Applications appear in various fields, but most prominently and obviously in the construction of communication networks, where the root node represents a central server and terminals potential clients.

In the simplest case, such a problem can be modeled as an efficiently solvable spanning tree problem, but additional options like possibly includable intermediate nodes, delay, length and/or more general resource constraints, and different objectives make these kind of problems most of the time NP-hard, and moderate to large instances are frequently very difficult to solve to proven optimality in practice. As long as aspects like redundant connections to terminals in order to achieve higher connectivity and robustness to failure are excluded, solutions have tree structure, and such problems can be modeled as extensions of the *Steiner tree problem on a graph* (STP).

If considering a problem variant in which all terminals need to be connected obligatorily one usually aims to identify a solution yielding overall minimal costs for establishing the network. On the contrary, in many real world applications the primary goal is to maximize the net profit, which is the profit earned by connecting customers reduced by the investment to build the network. Such scenarios are frequently called prize collecting network design problems.

Over the last decades, many variants of such problems have been studied and a large variety of exact and approximate solution techniques have been suggested. Most of the proposed works, however, are targeted towards very specific problem variants, special features of these are rigorously exploited, and well working solution techniques cannot at all or can only with a high effort be adapted to similar but slightly different problem variants. This is, for example, frequently the case with leading branch-and-cut approaches but also for many implementations of metaheuristics.

An in principle more generally applicable concept comes from the Dantzig-Wolfe decomposition and is based on column generation. Many if not most network design problems of the above described class can be addressed by path models using mixed integer linear programming (MIP), in which any feasible path to a terminal is represented by an own variable (column). In the process of solving such a model, column generation is used to introduce new paths/variables not yet considered in order to possibly improve a current state. In this way, the consideration of many specific constraints and other aspects is effectively delegated to the pricing subproblem, and the main algorithm stays more general.

Such solution approaches based on column generation and branch-and-price are, however, often believed as not competitive to other state-of-the-art methods. This is somewhat in contrast to other domains like cutting and packing, where column generation based methods are applied with much more success. One of the main problems of column generation in the network design domain are diverse sorts of degeneracy, which lead to many iterations and long running times.

In this article we describe the application of column generation stabilization based on alternative dual-optimal solutions to overcome the computational problems of branch-and-price approaches in the domain of network design. Furthermore, we show that the proposed stabilization approach allows for additionally including many real world relevant side constraints. Hence the obtained stabilized branch-and-price approach can be used to model realistic network design problems involving multiple side constraints. Computational experiments document that our stabilization approach yields a substantial improvement and clearly outperforms an alternative stabilization method based on piecewise linear penalty terms. We further show that due to the achieved speed-up the resulting branch-and-price is competitive to a state-of-the-art branch-and-cut approach based on layered graphs.

Problem Definition. More precisely, we consider here the class of rooted prize collecting network design problems where (i) each terminal (customer node) may be connected by installing a single path from the dedicated root node to it, (ii) each such path must respect some QoS constraint(s) according to some resource function(s) defined on potential edges, and (iii) the overall solution must form a tree.

Formally, we start with the following basic scenario. Given are an undirected graph $G = (V, E)$ with node set V, edge set E, and a dedicated root node $s \in V$. Each edge $e \in E$ is associated with costs $c_e \in \mathbb{Z}^+$ and a resource value $r_e \in \mathbb{Z}^+$, respectively. Furthermore, we are given node profits (prizes) $p_i \in \mathbb{N}_0$, $\forall i \in V$, earned when connecting node i to the root node in a feasible way. These profits partition the node set $V \setminus \{s\}$ into terminal nodes $T = \{i \in V \setminus \{s\} : p_i > 0\}$ and Steiner nodes $S = V \setminus (T \cup \{s\})$. Finally, we are given a resource bound $B \geq 0$ and a potentially empty set $T' \subseteq T$ of terminal nodes that need to be connected to the root node obligatorily.

A feasible solution to this problem is a Steiner tree $G_S = (V_S, E_S)$, $V_S \subseteq V$, $E_S \subseteq E$, containing the root node, i.e. $s \in V_S$, and all mandatory terminals, i.e. $T' \subseteq V_S$. Furthermore, the total resource usage along the unique path from the root node to each connected terminal $t \in (T \cap V_S)$ may not exceed the resource bound B. Formally, if $p_S(t) \subseteq E_S$ denotes the edge set of the path from s to terminal $t \in T$, then $\sum_{e \in p_S(t)} r_e \leq B$ must hold for all connected terminals $t \in (T \cap V_S)$. Depending on the concrete problem variant an optimal solution $G_S^* = (V_S^*, E_S^*)$ is a feasible solution with either minimal costs $\sum_{e \in E_S^*} c_e$ or maximal net profit $\sum_{t \in T \cap V_S^*} p_t - \sum_{e \in E_S^*} c_e$, respectively. In the following, we primarily consider the latter, more general prize collecting case and highlight necessary adaptations to the cost minimization variant where necessary. For a more uniform notation, we do, however, use the equivalent minimization form in which we add the sum of all potential profits to ensure non-negativity:

$$p(G_S^*) = \min \sum_{e \in E_S^*} c_e - \sum_{t \in T \cap V_S^*} p_t + \sum_{t \in T} p_t = \min \sum_{e \in E_S^*} c_e + \sum_{t \notin (T \cap V_S^*)} p_t$$

The remainder of this article is organized as follows: After discussing previous and related work in Section 2 we describe the details of our branch-and-price approach in Section 3, including the pricing subproblem and the issue of branching. Section 4 introduces stabilization approaches based on alternative dual-optimal solutions. In Section 5 we discuss four classes of practically important, additional constraints in a general context and show that all main observations from Section 4.1 as well as the proposed stabilization approaches remain valid when including them in our model. Sections 6.1 and 6.2 contain a computational study on the *rooted delay-constrained Steiner tree problem* (RDCSTP) and a variant additionally involving quota constraints, respectively. Finally, we conclude in Section 7 where we also discuss potential aspects that may be considered in future work.

2 Previous and Related Work

The Steiner tree problem on graphs has been introduced by Dreyfus and Wagner in 1971 [14] and since then many variants of this problem with additional constraints emerged in literature. Practical applications, e.g. multimedia content distribution and VoIP, ask for QoS constraints such as limiting the communication delay between server and clients. Therefore, two problem variants particularly increased in popularity, the already mentioned RDCSTP (also known as *multicast routing problem*) and the *hop-constrained Steiner tree problem* (HCSTP) where $r_e = 1, \forall e \in E$, modeling the fact that in many cases only the number of distribution and routing nodes in an end-to-end connection is relevant.

The RDCSTP is introduced and proven to be NP-hard by Kompella et al. [27] who also presented a construction heuristic. Manyem and Stallmann [37] showed that the RDCSTP and HCSTP are not in APX even when considering the spanning tree variants. There are lots of recent publications dealing with these problems and related variants, see e.g. [41, 50, 51] for recent metaheuristic approaches. Metaheuristics for the spanning tree variant with $T' = V \setminus \{s\}$ are presented by Ruthmair and Raidl [44, 45] who also discussed effective preprocessing methods to reduce the size of a given instance graph. The latter are also applied in this work, extended by a simple test removing Steiner nodes that cannot be part of a delay-constrained tree. Construction and local search heuristics for the HCSTP have been described by Voß [48], Fernandes et al. [17], and Gouveia et al. [20].

Exact methods for the RDCSTP are dominated by mixed integer programming (MIP) methods. Leggieri et al. [28] presented a compact extended node-based formulation using lifted Miller-Tucker-Zemlin inequalities yielding rather weak linear programming (LP) relaxation bounds. Hence, they further tightened the formulation adding directed connection inequalities in a typical branch-and-cut way. Further MIP approaches for the spanning tree variant are introduced by Gouveia et al. in [21] first stating a path formulation for the problem and then solving it in three different ways. Unstabilized delayed column generation turned out to be computationally inefficient whereas Lagrangian relaxation dualizing the constraints linking path and edge variables combined with an efficient primal heuristic yields better results.

In the third approach of [21] the constrained shortest path problem for each node is modeled on a layered graph and solved by a multi commodity flow (MCF) formulation. Each layer in this extended graph corresponds to a specific path delay from the root node. Original nodes are then duplicated on each layer modeling the visit of a node exactly at the corresponding path delay. Each edge in G is copied in a similar way skipping a number of layers that corresponds to the edge's delay. Thus, delays are implicitly encoded in the layered structure and therefore do not have to be considered explicitly anymore in a solution approach. Obviously, the size of the layered graph and therefore the efficiency of an according MIP model strongly depends on the number of achievable discrete delay values making this approach only usable for instances in which this number is relatively low. Additionally, MCF models frequently suffer in practice from the huge amount of used flow variables, altogether leading to a slow and memory-intensive solving process. Nevertheless, solving these layered graph models turned out to be highly effective on certain classes of instances. In [46] the approach of [21] is extended such that not just the constrained shortest path problems for each terminal but the whole RDCSTP is modeled on a layered graph which reduces to solving the classical STP on this graph. The definition of the layered graph implies its acyclicity allowing to prevent cycles with a polynomial number of connectivity constraints without additional variables, see [19]. However, additionally including directed cut inequalities yields a tighter or at least equal LP bound than all other known formulations for the RDCSTP. This result was shown by Gouveia et al. [22] for the HCSTP and can be generalized to the RDCSTP in a natural way. To overcome the issue of an excessive number of layers in case of a larger number of achievable delay values, a so-called *Adaptive Layers Framework (ALF)* based on iteratively solving smaller layered graphs is presented in [46] yielding lower and upper bounds to the optimal solution costs. By successively extending these smaller graphs appropriately, the bounds are tightened to finally converge to an optimal solution. In practice, this approach usually yields very small gaps even on instances where the directed cut formulation on the layered graph is not able to derive an optimal LP value.

Recently, we proposed stabilized column generation and branch-and-price approaches for the RDCSTP [32]. The current article significantly extends this publication by formalizing and generalizing these methods to a broader range of constrained tree problems discussing several families of constraints. Furthermore, we provide additional experimental results on the quota-constrained variant of the RDCSTP. To the best of our knowledge no work has been published tackling the RDCSTP in a prize-collecting or quota-constrained fashion. Related work on stabilizing delayed column generation is discussed in Section 4 and further problem variants of the STP are considered in Section 5.

3 Branch-and-Price Approach

In this section we first introduce a branch-and-price approach based on an MIP model utilizing variables corresponding to feasible paths. We further address the pricing subproblem as well as the issue of branching in branch-and-price.

3.1 Path Model

Our path based MIP model is a rather straightforward adaptation of similar models previously proposed, see e.g. [21]. Since directed formulations are usually tighter than undirected ones, it utilizes a directed arc set A containing an arc (s, j) for each edge incident to the root and two oppositely directed arcs for all remaining edges, i.e. $A = \{(s, j) \mid \{s, j\} \in E\} \cup \{(i, j), (j, i) \mid \{i, j\} \in E,\ i, j \neq s\}$.

We assume the edge cost and resource functions to be correspondingly defined on arc set A, i.e. $c_{ij} = c_e$, $r_{ij} = r_e$, $\forall (i, j) \in A,\ e = \{i, j\} \in E$. A solution to the integer master problem (IMP) defined by (1)–(8) is represented by an outgoing arborescence rooted at s. The IMP utilizes decision variables x_{ij}, $\forall (i, j) \in A$, and y_i, $\forall i \in V$, on arcs and nodes, respectively. Furthermore, path variables $\lambda_g \in \{0, 1\}$, $\forall g \in \mathcal{P}$ are used, where $\mathcal{P} = \bigcup_{t \in T} \mathcal{P}_t$, and $\mathcal{P}_t \subseteq 2^A$ is the set of all feasible paths from the root node s to terminals $t \in T$ represented by their set of arcs; $\sum_{(i,j) \in g} r_{ij} \leq B$ must hold for each path $g \in \mathcal{P}$. We also introduce corresponding dual variables in parentheses in formulation (1)–(8) as they will be needed for explaining the pricing problem as well as for discussing the dual problem in Section 4.1.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij}x_{ij} + \sum_{i\in V} p_i(1-y_i) && && (1)\\
\text{s.t.} \quad & \sum_{g\in\mathcal{P}_t} \lambda_g - y_t \geq 0 && (\mu_t) \quad \forall t \in T && (2)\\
& x_{ij} - \sum_{g\in\mathcal{P}_t | (i,j)\in g} \lambda_g \geq 0 && (\pi^t_{ij}) \quad \forall t\in T,\ \forall (i,j)\in A && (3)\\
& y_j - \sum_{(i,j)\in A} x_{ij} = 0 && (\gamma_j) \quad \forall j \in V\setminus\{s\} && (4)\\
& y_t = 1 && (\rho_t) \quad \forall t\in T'\cup\{s\} && (5)\\
& y_i \in \{0,1\} && \forall i \in V\setminus(T'\cup\{s\}) && (6)\\
& x_{ij}\in\{0,1\} && \forall (i,j)\in A && (7)\\
& \lambda_g \geq 0 && \forall g\in\mathcal{P} && (8)
\end{aligned}
$$

As previously defined the objective function (1) minimizes the sum of costs of realized arcs and not gained node profits. Constraints (2) ensure that profits can only be earned if the corresponding terminal node is connected to the root node by a feasible path, while Constraints (3) link path variables to arcs used by them. Equations (4) link arc with node variables and hence ensure that the maximum indegree of each node is one. This together with the fact that the solution obviously is connected since each path contains the root node guarantees that each solution is a tree. Finally, Constraints (5) ensure that all mandatory nodes will be connected; Constraints (6) and (7) are the integrality constraints on node and arc variables. Path variables for which only lower bounds are imposed by Inequalities (8) will automatically become integral due to the other constraints.

Note that variables y_i could be easily removed for Steiner nodes $i \in S$. We do, however, include them since branching on node variables first frequently turns out to yield better performance than branching on arc variables only.

The number of feasible paths and hence the total number of variables of the IMP may be exponentially large for each terminal. Thus, we cannot solve the IMP directly, but apply branch-and-price, i.e. embed delayed column generation in a branch-and-bound approach, cf. [5, 12]. For each node of the branch-and-bound tree we then need to solve the *restricted master problem* (RMP) using delayed column generation. This RMP is defined by replacing the integrality constraints (6)–(7) by (9)–(11) and additionally considering only a small subset $\tilde{\mathcal{P}}_t \subseteq \mathcal{P}_t$, $\forall t \in T$, of path variables, which must not be empty for obligatory terminals, i.e. $\tilde{\mathcal{P}}_t \neq \emptyset$, $\forall t \in T'$.

$$y_i \leq 1 \qquad (\rho_i) \quad \forall t \in V \setminus (T' \cup \{s\}) \tag{9}$$

$$y_i \geq 0 \qquad \forall i \in V \setminus (T' \cup \{s\}) \tag{10}$$

$$x_{ij} \geq 0 \qquad \forall (i,j) \in A \tag{11}$$

In the following we formally introduce the pricing subproblem and discuss the issue of branching in branch-and-price. Details on the algorithm used for solving the pricing subproblem will be given in Section 6.1 where we also describe which variables are added in each iteration.

3.2 Pricing Subproblem

When solving a node of the branch-and-price tree by column generation, we need to repeatedly identify path variables with negative reduced costs and add at least one of them to the RMP, which in turn needs to be resolved. This process is repeated until no more variables with negative reduced costs exist. The reduced costs $\bar{c}_p$ of any not yet included path variable λ_g, $g \in \mathcal{P}_t \setminus \tilde{\mathcal{P}}_t$, $t \in T$, are given by $\bar{c}_g = -\mu_t + \sum_{(i,j)\in g} \pi^t_{ij}$, where $\mu_t \geq 0$, $\forall t \in T$, and $\pi^t_{ij} \geq 0$, $\forall t \in T$, $\forall (i,j) \in A$.

In order to prove that no more negative reduced cost variables do exist, we need to compute the path variable yielding minimal reduced costs. Thus the pricing subproblem is formally defined as

$$(t^*, g^*) = \mathrm{argmin}_{t\in T, g\in\mathcal{P}_t} - \mu_t + \sum_{(i,j)\in g} \pi^t_{ij}. \tag{12}$$

This problem can be solved by computing a cheapest path g from the root node s to each terminal $t \in T$ using arc costs π^t_{ij}, which does not violate the given resource bound, i.e. $\sum_{(i,j)\in g} r_{ij} \leq B$. In case the total costs $\sum_{(i,j)\in g} \pi^t_{ij}$ of such a path are smaller than μ_t, variable λ_g has negative reduced costs and may be added to the current RMP.

It is well known that the problem of finding a minimum cost resource-constrained shortest path with non-negative arc costs is NP-hard in the weak sense, cf. [18], and can be solved in pseudo-polynomial time. Recently, algorithms based on dynamic programming with computational complexity of $O(B \cdot |A|)$ have been described for solving practical instances of this problem quite efficiently, see e.g. [16, 21].

3.3 Branching in Branch-and-Price

Depending on the concrete model used, branching in branch-and-price in a meaningful way may be nontrivial. On the one hand, one should ensure that branching decisions do not change the structure of the pricing subproblem. On the other hand it is important to avoid branching on the exponentially large set of variables, i.e. the path variables λ_g in our case. Fixing such a variable to zero usually has no or little impact while fixing it to one dramatically reduces the search space, cf. [5, 13]. Hence, such branching rules usually yield a highly asymmetric partitioning of the search space leading to rather poor overall performance.

In our case, however, branching can simply be performed on fractional node and arc variables. Note that while it would be sufficient to restrict only arc variables to be integral, we additionally consider branching on node variables since fixing a node influences variables of adjacent arcs and thus often has a stronger impact than just fixing a single arc variable.

4 Column Generation Stabilization

Branch-and-price and column generation algorithms often suffer from computational instabilities leading to excessive runtimes. Following the classification by Vanderbeck [47] these include the generation of irrelevant columns in the beginning due to poor dual information (*heading-in effect*), slow convergence (*tailing-off effect*), and multiple optimal solutions in the dual problem (*primal degeneracy*) leading to relatively slow re-optimization after adding new columns.

Different stabilization techniques have been proposed to overcome these problems by reducing the impacts of these negative effects, see e.g. [35, 36] for recent reviews including many other aspects of column generation. For example, the boxstep method [38] restricts each dual variable to a trust region around a current stability center. Based on this idea several approaches have been introduced that penalize deviations from the current stability center, which is finally updated as long as the obtained solution does not lie within the trust region; cf. [2, 15]. Amor et al. [4] compared various possibilities and concluded that piecewise linear penalty functions generally work well and hence are a good option, in particular as the resulting model remains linear. Quadratic penalty terms as used in bundle methods would in principle be favorable. Hence they may become more attractive in future in case the efficiency of quadratic programming solvers significantly increases [7].

Weighted Dantzig-Wolfe decomposition [40, 49] does not modify the RMP but tries to obtain a better column using a convex combination of current dual prices and those generating the best Lagrangian dual bound so far. Other approaches include trying to obtain solutions inside the dual space [43] or adding valid inequalities to the dual [3, 11].

Stabilization using alternative dual-optimal solutions has been introduced by the current authors in the context of survivable network design [29, 30, 31]. We further showed its applicability to the RDCSTP [32, 33] and highlighted that it usually yields a significant speed-up and reduces the numbers of necessary pricing iterations and finally included variables. In the following, we review the concept of alternative dual-optimal solutions and put the main results on a more formal basis allowing to finally show its general applicability for many network design problems in Section 5.

4.1 Alternative Dual-Optimal Solutions

Stabilization based on alternative dual-optimal solutions exploits the fact that due to primal degeneracy multiple solutions to the dual of the RMP, i.e. the *restricted dual problem* (RDP), exist. Given some optimal solution D^* to the RDP it aims to generate a different optimal solution D' that facilitates the generation of more relevant variables early in the column generation process in order to reduce the heading-in effect. We will further argue that our approach also helps to reduce the tailing-off effect and hence further reduces the total runtime. These arguments are strongly supported by our computational results in Sections 6.1 and 6.2, respectively. Additionally, we consider the fact that these alternative solutions are only used to solve the pricing subproblem and hence we do not need to modify the RMP another main advantage.

Before showing how to construct alternative optimal solutions, we introduce the RDP (13)–(20) and make some important observations; corresponding primal variables are given in parentheses.

$$
\begin{aligned}
\max \quad & \sum_{i \in V} (\rho_i + p_i) & & & (13)\\
\text{s.t.} \quad & \sum_{t \in T} \pi_{ij}^t - \gamma_j \leq c_{ij} & (x_{ij}) \quad & \forall (i,j) \in A & (14)\\
& \mu_t - \sum_{(i,j) \in g} \pi_{ij}^t \leq 0 & (\lambda_g) \quad & \forall t \in T,\ \forall g \in \tilde{\mathcal{P}}_t & (15)\\
& -\mu_t + \gamma_t + \rho_t \leq -p_t & (y_t) \quad & \forall t \in T & (16)\\
& \gamma_i + \rho_i \leq 0 & (y_i) \quad & \forall i \in V \setminus (T \cup \{s\}) & (17)\\
& \mu_t \geq 0 & & \forall t \in T & (18)\\
& \pi_{ij}^t \geq 0 & & \forall t \in T,\ \forall (i,j) \in A & (19)\\
& \rho_i \leq 0 & & \forall i \in V \setminus (T' \cup \{s\}) & (20)
\end{aligned}
$$

Our primary interest concerns variables π_{ij}^t, $\forall t \in T$, $\forall (i,j) \in A$, since they are used as arc costs in the pricing subproblem and hence are of major importance. We observe that Inequalities (14) are capacity constraints imposing upper bounds on the sum of these variables $\sum_{t \in T} \pi_{ij}^t$ for each arc $(i,j) \in A$. These variables are additionally only included in Constraints (15) which ensure that the corresponding sum over all arcs of each path to some terminal t is at least μ_t. These considerations allow to further analyze some properties of optimal solutions more formally.

Theorem 1. *Assume* (13)–(20) *has a feasible solution and let* $(i,j) \in A$ *be an arc that is not contained in any path* $g \in \tilde{\mathcal{P}} = \bigcup_{t \in T} \tilde{\mathcal{P}}_t$*, i.e.* $\nexists g \in \tilde{\mathcal{P}} : (i,j) \in g$*. Then there exists an optimal solution to* (13)–(20) *such that* $\pi^t_{ij} = 0$*,* $\forall t \in T$*.*

Proof. If the RDP has a feasible solution it also has an optimal solution $D^* = (\mu^*, \pi^*, \gamma^*, \rho^*)$. Let $(i,j) \in A$ be an arc not contained in any so far included path and ${\pi^t_{ij}}^* \geq 0$ the corresponding optimal solution value for some arbitrarily chosen terminal $t \in T$. Obviously, setting ${\pi^t_{ij}}^*$ to zero does not violate the capacity constraint (14). Since arc (i,j) is by assumption not contained in any path $g \in \tilde{\mathcal{P}}$, variable π^t_{ij} is not contained in any coupling constraint (15) and hence we obtain another feasible solution. Since the objective value remains constant, this new solution also is optimal. In this way, any ${\pi^t_{ij}}^* > 0$ is independently replaced by zero for all $t \in T$. □

In particular we can state the following corollary which immediately follows from Theorem 1.

Corollary 1. *If* (13)–(20) *has a feasible solution then there also exists an optimal solution such that* $\pi^t_{ij} = 0$*,* $\forall t \in T$*,* $\forall (i,j) \in A' = \{(u,v) \in A \mid \nexists g \in \tilde{\mathcal{P}} : (u,v) \in g\}$*.*

This observation has direct implications in practice, and we can now explain why in particular the heading-in effect is very pronounced and has typically dramatic consequences w.r.t. the number of pricing iterations and thus running time: Most if not all state-of-the-art implementations of LP solvers yield optimal solutions with each variable has a minimal value, i.e., exactly a solution as proven to exist by Corollary 1. Especially in the early iterations of column generations, most arcs (i,j) will not be part of any so far included paths, i.e. $(i,j) \in A'$, and consequently $\pi^t_{ij} = 0$. Hence most arc costs in the pricing subproblem are zero, and there is almost no guiding information for creating meaningful paths. As a consequence, many irrelevant columns are priced in in the beginning until column generation converges to more meaningful dual variable values.

Based on the following definition introducing the concept of dual slack of arcs, we can define an alternative class of always existing optimal solutions in Theorem 2, which appear to be more promising from the beginning on.

Definition 1 (Dual Slack)**.** *Let* $D^* = (\mu^*, \pi^*, \gamma^*, \rho^*)$ *be an optimal solution to the RDP. Then the* dual slack Δ_{ij} *of arc* $(i,j) \in A$ *with respect to* (13)–(20) *is defined as*

$$\Delta_{ij} = c_{ij} + \gamma^*_j - \sum_{t \in T} {\pi^t_{ij}}^*. \qquad (21)$$

Theorem 2. *If* (13)–(20) *has a feasible solution than there exists an optimal solution such that all capacity constraints* (14) *are binding.*

Proof. Let $D^* = (\mu^*, \pi^*, \gamma^*, \rho^*)$ be an optimal solution of the RDP and $\Delta_{ij} \geq 0$, $\forall (i,j) \in A$, be the corresponding dual slack values. We first observe that increasing variable values ${\pi^t_{ij}}^*$ does not change the objective value and since Constraints (15) impose only lower bounds on them may not violate any other constraints than the capacity Constraints (14). Furthermore, since each capacity constraint refers to a different arc we can consider them independently of each other. Hence, by increasing $\sum_{t \in T} \pi^t_{ij}$ by the dual slack Δ_{ij} for all arcs $(i,j) \in A$ we obtain an optimal solution to RDP in which all capacity constraints are binding. □

The following corollary follows from the proof of Theorem 2 and reveals our basic strategy to construct an alternative dual-optimal solution.

Corollary 2. *Given an optimal solution* $D^* = (\mu^*, \pi^*, \gamma^*, \rho^*)$ *to the RDP we can construct a possibly different optimal solution* $\bar{D} = (\bar{\mu}, \bar{\pi}, \bar{\gamma}, \bar{\rho})$ *such that* $\bar{\mu} = \mu^*$, $\bar{\gamma} = \gamma^*$, $\bar{\rho} = \rho^*$, *and* $\bar{\pi}^t_{ij} = {\pi^t_{ij}}^* + \frac{\Delta_{ij}}{|T|}$, $\forall t \in T$, $\forall (i,j) \in A$, *and all capacity constraints* (14) *are binding.*

Corollary 3 finally, introduces the necessary formal basis for a more fine grained approach in which different dual variable values are iteratively tried. It turned out to be beneficial in our previous work [31, 32].

Corollary 3. *Given an optimal solution* $D^* = (\mu^*, \pi^*, \gamma^*, \rho^*)$ *to the RDP, a terminal* t', *an integer* $Q > 1$, *and a parameter* q, $1 \leq q \leq Q$, *respectively, we can construct a possibly different optimal solution* $\hat{D}^{t',q} = (\hat{\mu}^{t',q}, \hat{\pi}^{t',q}, \hat{\gamma}^{t',q}, \hat{\rho}^{t',q})$ *such that* $\hat{\mu}^{t',q} = \mu^*$, $\hat{\gamma}^{t',q} = \gamma^*$, $\hat{\rho}^{t',q} = \rho^*$, *and*

$$\hat{\pi}^{t',q}_{ij} = \begin{cases} {\pi^t_{ij}}^* + \frac{\Delta_{ij}}{|T|} + \frac{Q-q}{Q-1}\left(\Delta_{ij} - \frac{\Delta_{ij}}{|T|}\right) & \textit{if } t = t' \\ {\pi^t_{ij}}^* & \textit{otherwise} \end{cases}, \ \forall t \in T,\ \forall (i,j) \in A. \tag{22}$$

While we simply equally distribute the dual slack of each arc over all relevant variables to obtain an alternative solution $\bar{D}$ in our basic strategy, the approach following Corollary 3 utilizes an exogenous parameter $Q \geq 2$ denoting a total number of major iterations. Parameter q is initially set to one and incremented in case no negative reduced cost path could be found when solving the pricing subproblem using $\hat{D}^{t,q}$, $\forall t \in T$; it thus indicates the current major iteration. The dual solution $\hat{D}^{t,q}$ used for solving the pricing problem now further depends on the terminal $t \in T$ considered and hence we use different dual solutions for different terminals. Note that, while the individual solutions used are optimal as argued in Corollary 3, the dual cost vector formed by the union of actually used arc costs in the pricing subproblems together is infeasible for the RDP as long as $q < Q$. Hence, we increase q and resolve the pricing subproblem using the resulting different dual solutions if no more path variables yielding negative reduced costs with respect to $\hat{D}^{t,q}$ exist for all terminals $t \in T$.

Proposition 1 reveals that we essentially use $\bar{D}$ when $q = Q$ and thus can terminate column generation at the current node of the branch-and-price tree if no negative reduced cost variables do exist for all terminals and when $q = Q$.

Proposition 1. *If $q = Q$, then $\hat{\pi}^{t,q}_{ij} = \bar{\pi}^{t}_{ij}$, $\forall t \in T$, $\forall (i,j) \in A$.*

Informally speaking, the approach divides the interval $\left[\frac{\Delta_{ij}}{|T|}, \Delta_{ij}\right]$ into $Q - 1$ equally sized sub-intervals defining the dual variable values used for each iteration $q = 1, \dots, Q$. In the beginning the whole dual slack is added to the dual variables corresponding to the currently considered terminal. In successive iterations, the relative amount of dual slack added to the current terminal is decreased down to $\frac{\Delta_{ij}}{|T|}$.

To summarize, both approaches generate alternative dual-optimal solutions such that all variable values are frequently greater than or at least equal to the ones of the solution computed by traditional LP solvers, which in particular set $\pi^t_{ij} = 0$, $\forall t \in T$, for all arcs $(i,j) \in A'$ not part of any so far included paths. Especially in the beginning of column generation, our alternative variable values reflect to some degree original edge costs c_{ij} and therefore guide the construction of new paths in much more meaningful ways. More promising columns are consequently priced in from the beginning, reducing the heading-in effect.

Further note that any path variable yielding negative reduced costs with respect to $\bar{D}$ or $\hat{D}^{t,q}$, respectively, would also have negative reduced costs with respect to the dual solution computed by the used LP solver. The opposite is, however, not generally true. Thus, increasing dual variable values should also help to decrease the tailing-off effect.

5 Additional Constraints

Next to Quality-of-Service aspects that can be modeled by independent restrictions on feasible paths such as the resource (length or delay) constraints already considered in our basic model, several other real world relevant constraints have been used in network design models. In the following, we highlight four types of constraints in a general setting and discuss concrete examples of them that have appeared in literature. We then show that all results obtained in Section 4.1 remain valid for path models additionally involving an arbitrary number of these constraints. Hence, we prove that stabilization based on alternative dual-optimal solutions can be applied to realistic network design problems involving multiple side constraints simultaneously, i.e. to rich network design problems. Dual variable values are again annotated in parentheses.

Constraints on Arc Variables. We first consider a set of Constraints (23) that impose upper bounds on used arcs over all paths according to some additional resource functions w^l_{ij}, $\forall l \in \mathcal{R}$, $\forall (i,j) \in A$, and resource bounds w^l, $\forall l \in \mathcal{R}$. Obviously, we can use them to also model similar constraints involving only certain subsets of arcs using a correspondingly defined resource function with zero weights for all not relevant arcs. Such constraints have been used for the weight-constrained STP, cf. [42], as well as the budget-constrained STP, cf. [10, 26], restricting the sum of costs of realized arcs by an upper bound.

$$\sum_{(i,j)\in A} w^l_{ij} x_{ij} \leq w^l \qquad (\zeta_l) \quad \forall l \in \mathcal{R} \tag{23}$$

Constraints on Node Variables. Similarly, we can consider Constraints (24) involving nodes included in a solution. As opposed to arc constraints these constraints impose lower bounds r^q, $\forall q \in \mathcal{Q}$, according to additional revenue functions r^q_i, $\forall q \in \mathcal{Q}$, $\forall i \in V \setminus \{s\}$. We note that these type of constraints has been used to ensure a certain amount of revenues to be collected by any feasible solution in variants of the quota-constrained STP, cf. [23, 24].

$$\sum_{i\in V\setminus\{s\}} r^q_i y_i \geq r^q \qquad (\eta_q) \quad \forall q \in \mathcal{Q} \tag{24}$$

Obviously, we could also impose corresponding upper bounds in the very same way.

Constraints on Node and Arc Variables. Another type of constraints that is frequently encountered in the literature combines variables of nodes with those of incident arcs. They can e.g. be used to include degree constraints restricting the number of links incident to a node, cf. [6, 39]. Since due to the directed tree structure of our solutions there may be at most one ingoing arc in our setting, we restrict corresponding constraints to emanating arcs. Constraints (25) generalize this idea by considering weights $w_{ij} \geq 0$, $\forall (i,j) \in A$, and restricting the total weight of emanating arcs to D_i, $\forall i \in V$.

$$D_i y_i - \sum_{(i,j)\in A} w_{ij} x_{ij} \geq 0 \qquad (\delta_i) \quad \forall i \in V \tag{25}$$

Capacity Constraints. Finally, we consider capacity constraints on arcs which are of particular interest in many real world applications, cf. [9, 25]. Constraints (26) assume that each terminal node $t \in T$ has some demand $d^t \geq 0$ that shall be satisfied by the path connecting the terminal, while the maximum total demand routed along arc $(i,j) \in A$ may not exceed a given capacity $C_{ij} \geq 0$.

$$C_{ij}x_{ij} - \sum_{t \in T} \sum_{g \in \mathcal{P}_t : (i,j) \in g} d^t \lambda_g \geq 0 \qquad (\nu_{ij}) \quad \forall (i,j) \in A \tag{26}$$

Since Constraints (26) involve path variables, we also need to consider potential changes of the pricing subproblem. Formally, the pricing subproblem for the IMP including Constraints (26) is defined as

$$(t^*, g^*) = \operatorname{argmin}_{t \in T, g \in \mathcal{P}_t} - \mu_t + \sum_{(i,j) \in g} \left(\pi^t_{ij} + d^t \nu_{ij} \right). \tag{27}$$

Hence, the general structure remains identical, i.e. we need to solve a minimum cost resource-constrained shortest path problem for each terminal $t \in T$, just the arc costs have changed to $\pi^t_{ij} + d^t \nu_{ij}$, $\forall (i,j) \in A$. As $\nu_{ij} \geq 0$, $\forall (i,j) \in A$, these arc costs are strictly nonnegative and we can use the same pseudo-polynomial algorithms than for solving the original pricing subproblem (12).

Analysis of the Resulting Dual Problem. To show that stabilization using alternative dual-optimal solutions is still possible in the previously described way when including all the additional Constraints (23) to (26) in the IMP, we first derive the correspondingly extended restricted dual problem RDP$^+$:

$$\max \quad \sum_{i\in V}(\rho_i + p_i) + \sum_{l\in\mathcal{R}} w^l \zeta_l + \sum_{q\in\mathcal{Q}} r^q \eta_q \tag{28}$$

$$\text{s.t.} \quad \sum_{t\in T}\pi^t_{ij} - \gamma_j + \sum_{l\in\mathcal{R}} w^l_{ij}\zeta_l - w_{ij}\delta_i + C_{ij}\nu_{ij} \le c_{ij} \qquad (x_{ij}) \quad \forall (i,j)\in A \tag{29}$$

$$\mu_t - \sum_{(i,j)\in g}(\pi^t_{ij} + d^t\nu_{ij}) \le 0 \qquad (\lambda_g) \quad \forall t\in T,\ \forall g\in\tilde{\mathcal{P}}_t \tag{30}$$

$$-\mu_t + \gamma_t + \rho_t + \sum_{q\in\mathcal{Q}} r^q_t\eta_q + D_t\delta_t \le -p_t \qquad (y_t) \quad \forall t\in T \tag{31}$$

$$\gamma_i + \rho_i + \sum_{q\in\mathcal{Q}} r^q_i\eta_q + D_i\delta_i \le 0 \qquad (y_i) \quad \forall i\in V\setminus(T\cup\{s\}) \tag{32}$$

$$\mu_t \ge 0 \qquad \forall t\in T \tag{33}$$

$$\pi^t_{ij} \ge 0 \qquad \forall t\in T,\ \forall (i,j)\in A \tag{34}$$

$$\rho_i \le 0 \qquad \forall i\in V\setminus(T'\cup\{s\}) \tag{35}$$

$$\zeta_l \le 0 \qquad \forall l\in\mathcal{R} \tag{36}$$

$$\eta_q \ge 0 \qquad \forall q\in\mathcal{Q} \tag{37}$$

$$\delta_i \ge 0 \qquad \forall i\in V \tag{38}$$

$$\nu_{ij} \ge 0 \qquad \forall (i,j)\in A \tag{39}$$

We observe that while RDP$^+$ contains more variables than RDP the general structure remains identical. In particular lower bounds greater than zero for variables ν_{ij}, $\forall (i,j)\in A$, are only imposed if the corresponding arc is contained in at least one already included path variable. The latter is important since these variables are additionally included as arc costs in the pricing subproblem. Hence we can easily transfer our results from Section 4.1.

Theorem 3. *Equivalent versions of Theorem 1 and Corollary 1 do hold for variables π^t_{ij}, $\forall t\in T$, $\forall (i,j)\in A$, and ν_{ij}, $\forall (i,j)\in A$, of* RDP$^+$.

Proof. We can prove Theorem 3 using the same arguments as in the proof of Theorem 1. Given some optimal solution to RDP$^+$, we can obviously reduce variable values π^t_{ij} for all arcs that are not contained in any path $g\in\tilde{\mathcal{P}}$ and all terminals since these variables are not contained in further constraints and hence only restricted to be nonnegative. The same arguments hold for variables ν_{ij}. □

Corresponding to Definition 1, we define the dual slack Δ_{ij} of arc $(i, j) \in A$ with respect to RDP$^+$ as

$$\Delta_{ij} = c_{ij} + \gamma_j^* - \sum_{l \in \mathcal{R}} w_{ij}^l \zeta_l^* + w_{ij}\delta_i^* - C_{ij}\nu_{ij}^* - \sum_{t \in T} {\pi_{ij}^t}^*. \tag{40}$$

In the following we assume that D^*, $\bar{D}$, and $\hat{D}^{t,q}$ denote dual solutions of RDP$^+$, i.e. in comparison to those introduced in Section 4.1 they are correspondingly extended to additionally include variables ζ_l, η_q, and ν_{ij}, respectively. It is then easy to see that Theorem 2, Corollaries 2 and 3, as well as Proposition 1 are valid for RDP$^+$, too. Hence we conclude that both approaches introduced in Section 4.1, i.e. equally distributing the slack to obtain alternative dual-optimal solutions as well as initially using different solutions, remain feasible for RMP$^+$, i.e. for RMP additionally including Constraints (23) to (26).

6 Computational Results

We implemented all described variants of the branch-and-price approach in C++ using ZIB SCIP 2.0.2 [1] with IBM CPLEX 12.2 as embedded LP solver. Furthermore, a pure column generation solely using CPLEX has been implemented to analyze the impacts of stabilization on a pure LP basis without influence of other parts of a MIP framework like branching or primal heuristics. In the following subsections we refer by BP to the full branch-and-price approach, while CG denotes the column generation approach solving LP relaxations only.

Each computational experiment has been performed on a single core of an Intel Xeon E5540 processor with 2.53 GHz in a multi-core system where eight cores share 24GB RAM and an absolute limit of 10 000 CPU-seconds has been applied to each experiment. The dual simplex algorithm has been used for solving LPs in CG and BP, since it turned out to significantly outperform other options (primal simplex, barrier) in preliminary tests. For BP, we further deactivated presolving and separation of general purpose cutting planes (as recommended) and set parameter "fastmip" to one. Apart from that default settings and plugins of SCIP have been used.

6.1 Experiments on the Rooted Delay-Constrained Steiner Tree Problem

We first tested our approach on the RDCSTP using benchmark instances originally proposed by Gouveia et al. [21] for the spanning tree variant of the RDCSTP, i.e. $T' = V \setminus \{s\}$, focusing on the subsets C and E with Euclidean costs and the root node placed near the center (C) and near the border (E), respectively. Each instance set consists of five complete input graphs with 41 nodes and a specific range of possible discrete edge delay values, e.g. C100 denotes the set of instances where $r_e \in \{1, \dots, 100\}$, $\forall e \in E$. Note that we do not consider the instance sets from [21] with a very restricted range of delays, i.e. C2, E2, C5, and E5, since most of them can be solved almost instantly by BP as well as by the layered graph approach from [46]. Additionally instance sets Tα [32] consisting of 30 randomly generated complete graphs with $|V| = 100$ and $|T'| = \alpha$ have been used. All delays and costs are uniformly distributed in $\{1, \ldots, 99\}$. For reducing the input graphs we applied the preprocessing methods described in [45] prior to solving. Resulting average numbers of nodes ($|\overline{V}|$) and edges ($|\overline{E}|$) of each instance set are reported in Table 3.

We used a simple heuristic which iteratively adds delay-constrained shortest paths from the root node to terminal nodes while dissolving possible cycles to build an initial set of paths for CG and BP if $T \neq V \backslash \{s\}$. For spanning tree instances the Kruskal-based heuristic followed by variable neighborhood descent as introduced in [44] has been applied.

When solving the pricing subproblem, we potentially add multiple path variables for a single terminal in each iteration following an approach originally proposed by Gouveia et al. [21] since this method turned out to outperform the variant of adding at most one path variable per terminal in our previous work [32]. Their method iterates all nodes $i \in V$ adjacent to the currently considered terminal $t \in T$ and all delay values $b = 0, \ldots, B - r_{it}$ for which a path from s to i in conjunction with arc (i, t) is feasible. In case such a shortest path p to i of total delay b exists and $p' = p \cup \{(i, t)\}$ yields negative reduced costs, the variable corresponding to p' is added to the RMP.

As an additional improvement, we avoid zero arc costs in the pricing subproblem by using arc costs ε instead, where ε corresponds to the numerical precision of the used LP solver. This strategy turned out to be in particular helpful for standard column generation without stabilization.

6.1.1 Stabilization Based on Alternative Dual-Optimal Solutions

In our first set of experiments we aim at analyzing the overall efficiency of stabilization by alternative dual-optimal solutions as well as the influence of parameter Q in the approach initially using different dual-optimal solutions for different terminals. While large values of Q obviously may introduce a significant overhead for relatively easy instances, they turned out to be beneficial for hard instances in our previous work [32, 33].

Table 1 compares standard column generation D^* to stabilized approaches using alternative dual-optimal solutions for different instance sets and delay bounds. Here $\bar{D}$ denotes the approach equally distributing the dual slack over all relevant variables and $\hat{D}$ the more fine grained approach utilizing parameter Q. Next to numbers of solved instances ($\#_{\text{solved}}$) and median CPU-times in seconds (t_{total}) we also report the times needed for finding the correct LP value (t_{best}) in order to analyze a potential tailing-off effect. Best results are marked bold in Table 1 as well as all further ones. Additionally, we use dashes in all tables to indicate when the majority of instances for some setting could not be solved within the given time limit of 10 000 seconds or to denote the maximum optimality gap of 100%.

Table 1: Numbers of solved instances, median CPU-times in seconds, and median CPU-times for reaching the LP value for CG.

		$\#_{\text{solved}}$					t_{total} $[s]$					t_{best} $[s]$				
dual solution		D^*	$\bar{D}$	$\hat{D}$			D^*	$\bar{D}$	$\hat{D}$			D^*	$\bar{D}$	$\hat{D}$		
Set	$B \setminus Q$	-	-	10	20	30	-	-	10	20	30	-	-	10	20	30
C100	100	**5**	**5**	**5**	**5**	**5**	13	**6**	8	9	11	11	4	**3**	**3**	4
	150	**5**	**5**	**5**	**5**	**5**	25	**11**	12	13	15	13	7	3	**2**	**2**
	200	**5**	**5**	**5**	**5**	**5**	62	15	15	**14**	17	41	7	**2**	**2**	**2**
	250	**5**	**5**	**5**	**5**	**5**	344	**11**	14	13	14	124	7	**2**	**2**	**2**
E100	100	**5**	**5**	**5**	**5**	**5**	97	30	**13**	18	17	97	16	**5**	6	6
	150	4	**5**	**5**	**5**	**5**	2350	222	**26**	**26**	28	1380	112	11	11	**10**
	200	1	**5**	**5**	**5**	**5**	-	2407	**73**	80	93	-	2406	**44**	57	80
	250	2	**5**	**5**	**5**	**5**	-	7215	129	**68**	**68**	-	4302	71	53	**51**
C1000	1000	**5**	**5**	**5**	**5**	**5**	9	**5**	11	15	16	5	3	**2**	**2**	**2**
	1500	**5**	**5**	**5**	**5**	**5**	41	**11**	23	22	26	18	6	**2**	**2**	**2**
	2000	**5**	**5**	**5**	**5**	**5**	164	28	**24**	27	32	132	16	**3**	**3**	**3**
	2500	**5**	**5**	**5**	**5**	**5**	374	29	28	**27**	32	355	14	**4**	**4**	**4**
E1000	1000	**5**	**5**	**5**	**5**	**5**	80	27	**18**	23	29	58	15	**5**	**5**	6
	1500	**5**	**5**	**5**	**5**	**5**	1013	151	**40**	41	51	951	131	**10**	12	12
	2000	3	**5**	**5**	**5**	**5**	3395	796	106	91	**60**	2201	286	13	13	**10**
	2500	1	**5**	**5**	**5**	**5**	-	2672	157	81	**67**	-	1535	11	**8**	9
T30	16	**30**	**30**	**30**	**30**	**30**	**1**	**1**	2	3	4	**0**	**0**	1	1	1
	30	**30**	**30**	**30**	**30**	**30**	7	**4**	7	10	12	3	**1**	**1**	2	2
	50	**30**	**30**	**30**	**30**	**30**	36	**11**	15	20	23	20	4	**2**	3	4
	100	**30**	**30**	**30**	**30**	**30**	326	34	**29**	37	43	118	9	**3**	5	5
T50	16	**30**	**30**	**30**	**30**	**30**	**2**	**2**	4	6	8	**1**	**1**	2	2	2
	30	**30**	**30**	**30**	**30**	**30**	19	**9**	15	21	23	11	5	**4**	6	7
	50	**30**	**30**	**30**	**30**	**30**	99	36	**31**	40	49	33	13	**6**	7	8
	100	25	**30**	**30**	**30**	**30**	1305	167	94	**92**	95	1173	54	**9**	12	17
T70	16	**30**	**30**	**30**	**30**	**30**	**3**	**3**	7	10	12	**2**	**2**	**2**	3	4
	30	**30**	**30**	**30**	**30**	**30**	33	**18**	24	31	38	25	13	8	**7**	9
	50	**30**	**30**	**30**	**30**	**30**	186	60	**58**	63	75	129	30	**12**	15	17
	100	23	**30**	**30**	**30**	**30**	2007	318	185	195	**170**	1371	186	**25**	39	37
T99	16	**30**	**30**	**30**	**30**	**30**	6	**4**	12	15	17	4	**3**	4	4	**3**
	30	**30**	**30**	**30**	**30**	**30**	69	**39**	43	50	59	51	24	**15**	**15**	21
	50	**30**	**30**	**30**	**30**	**30**	331	**86**	99	118	120	296	52	**24**	26	29
	100	21	29	**30**	**30**	**30**	4044	880	373	**288**	290	3609	540	60	**55**	61

From Table 1 we conclude that stabilization using alternative dual-optimal solutions significantly outperforms standard column generation for all but a few rather trivial instance sets, i.e. whenever the median time of standard column generation exceeds ten seconds. Moreover, all three reported settings of $\hat{D}$ successfully solved the LP relaxations of all tested instances to optimality within the given time limit. Especially for the hardest instance sets these variants clearly outperform the simpler approach $\bar{D}$. Sometimes the required CPU-time is even reduced by more than one order of magnitude compared to $\bar{D}$, yielding a reduction of two orders of magnitude between $\hat{D}$ and D^*. We further observe that $\hat{D}$ usually needs the smallest relative amount of time $\frac{t_{\text{best}}}{t_{\text{total}}}$ to first reach the final LP value.

Figures 1 and 2 show relative median CPU-times and numbers of average pricing iterations and finally included path variables for $\hat{D}$ and various settings of Q using the corresponding values of $\bar{D}$ as baseline. We report on all instance sets where the median CPU-time exceeded 20 seconds when using $\bar{D}$.

We observe that for the hardest instances larger values of Q are beneficial while they sometimes introduce a moderate overhead for instances that could be solved relatively fast using $\bar{D}$. Since the absolute overhead

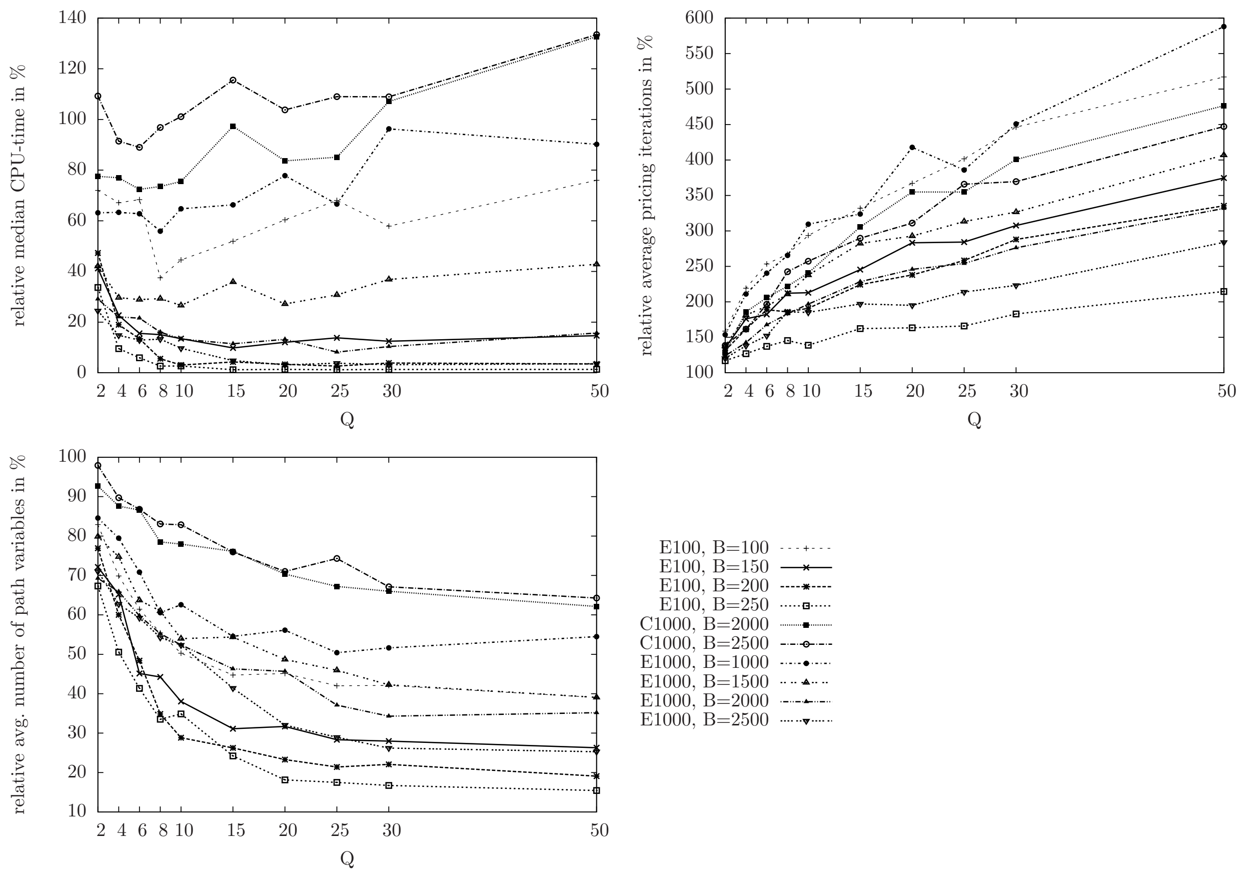

Figure 1: Median CPU-times, average pricing iterations, and average numbers of included path variables for $\hat{D}$ and different values of Q in % relative to $\bar{D}$ on spanning tree instances from [21].

is not too high in these cases and further increasing Q does not seem to have a significant positive impact, we conclude that choosing Q from the interval $[10, 30]$ generally seems to be a good compromise. Interestingly, the number of total pricing iterations, i.e. the number of times we need to resolve the RMP, increases for larger Q while the number of finally added path variables decreases. We conclude that for $\hat{D}$ less variables are included in each iteration and thus resolving the RMP needs less time. Nevertheless, finally necessary variables are typically added already early in the column generation process. Overall, a huge reduction of the needed CPU-time and the size of the final model in terms of included variables could be achieved. This observation is further supported by Figure 3 plotting the relative average gap in percent of the current value of the RMP to the LP solution value of the instance over the used CPU-time. Here, we exemplarily report on D^*, $\bar{D}$, and $\hat{D}$ with $Q = 10$ using the instance set E1000 with $B = 2000$ and $B = 2500$, respectively, as these turned out to be particularly hard.

While $\bar{D}$ and especially D^* need a considerable amount of time to significantly reduce the objective value of the RMP and even exhibit quite long plateaus in doing this, $\hat{D}$ identifies relevant columns early

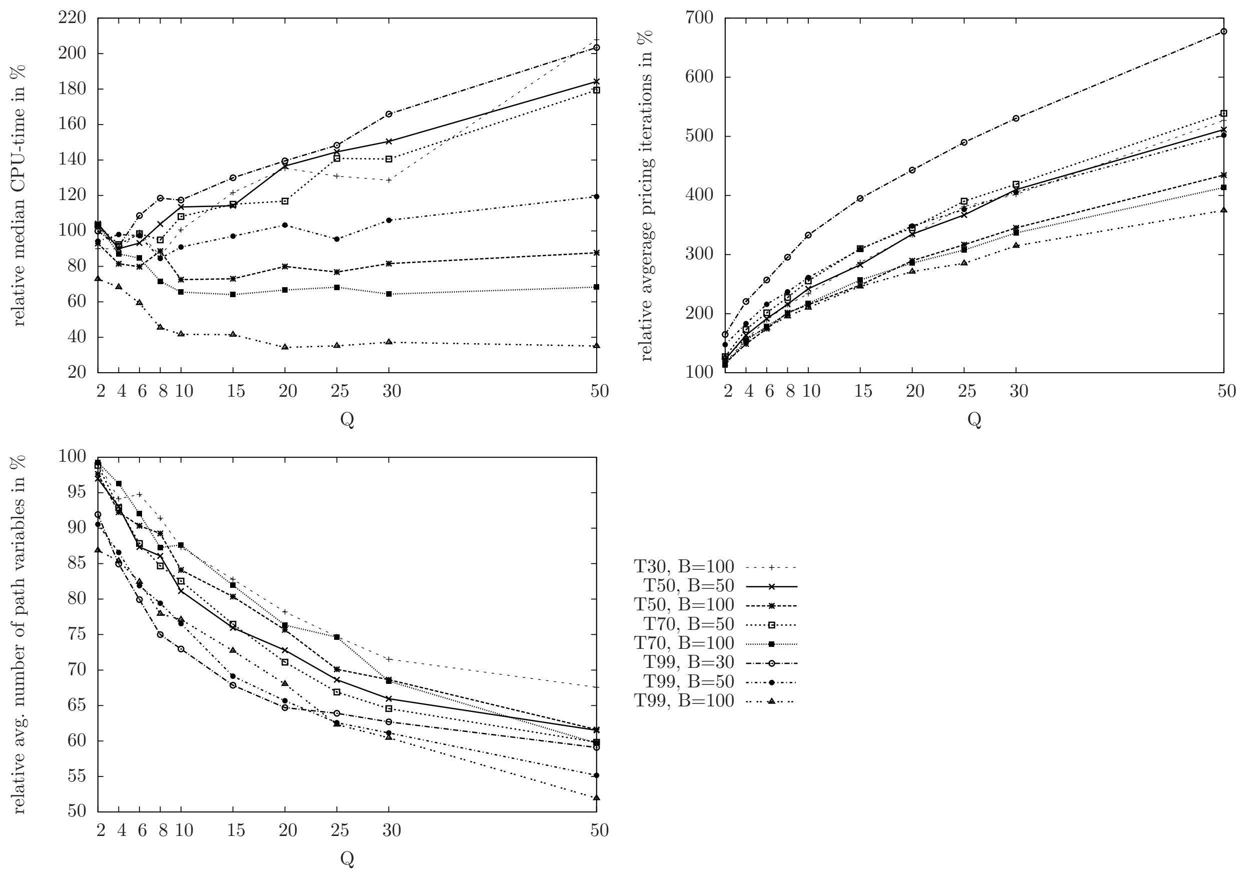

Figure 2: Median CPU-times, average pricing iterations, and average numbers of included path variables for $\hat{D}$ and different values of Q in % relative to $\bar{D}$ on random instances from [32].

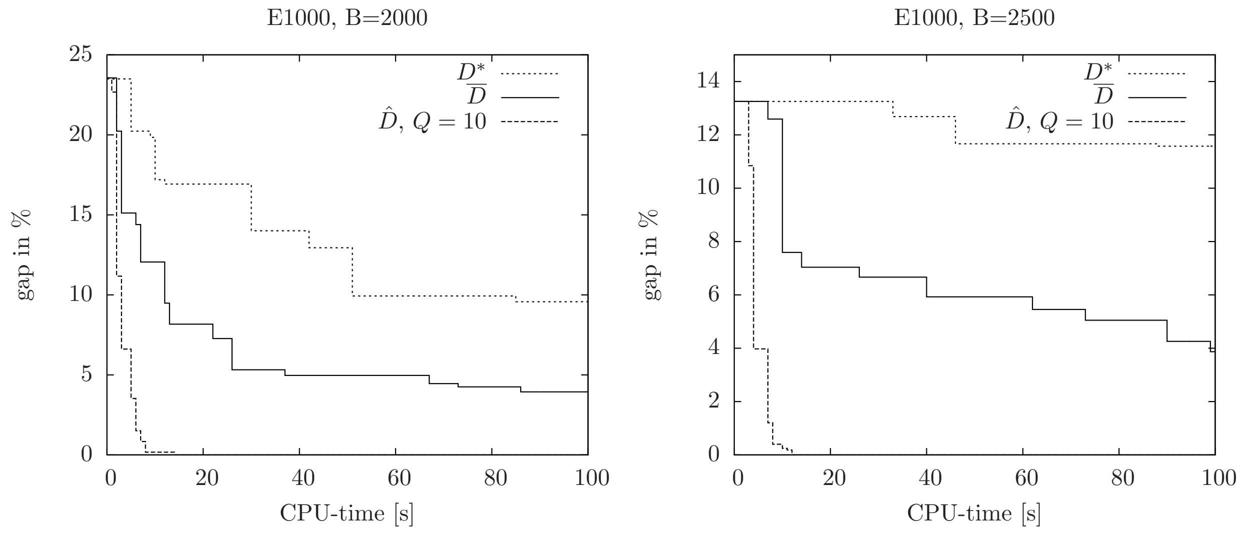

Figure 3: Relative gap in % to final LP value over time.

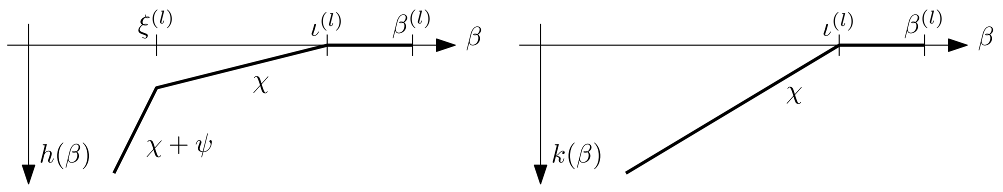

Figure 4: Piecewise linear dual penalty functions $h(\beta)$ and $k(\beta)$.

and since resolving is much faster it rapidly finds the final LP value. We further observed that there is often no significant difference between $\bar{D}$ and $\hat{D}$ when analyzing the current objective values over the so far performed iterations. Thus, the major advantage of $\hat{D}$ over $\bar{D}$ is its success in focusing on a smaller set of really important variables.

6.1.2 Comparison to Stabilization using Piecewise Linear Penalty Functions

We also compare our approaches to an alternative stabilization that penalizes deviations from a current stability center by adding piecewise linear penalty functions to the dual problem, cf. [2]. Our previous results [32, 33] indicate that penalizing only small dual variable values seems to perform better than penalizing deviations in both directions. Since this result is also supported by our analysis of the dual problem in Section 4.1 as well as by the huge reduction of CPU-time achieved using alternative dual-optimal solutions, we will focus on that case in the following. In each major iteration $l \in \mathbb{Z}^+$ we define a current stability center $\beta^{(l)} \in \mathbb{R}^m$, where $m = |T| \cdot (|A| + 1) + |V| - 1$ denotes the total number of constraints, i.e. the number of dual variables, that should be stabilized. Deviations from a current stability center are then penalized by adding a stabilization term to the primal problem, which corresponds to a penalty function in the dual. We tested two approaches using penalty functions $h(\beta)$ and $k(\beta)$, respectively, as shown in Figure 4. Dual variable values outside the current trust region $\left[\iota^{(l)}, \infty\right[$ are penalized according to χ and ψ. If all dual variable values are in the non-penalized region after the column generation process terminates at major iteration l we have solved the LP relaxation of the current node. Otherwise, we update the stability center according to the current dual solution. As has been shown previously [2] this process will terminate after finitely many major iterations.

We used identical penalty slopes and trust region radii for all dimensions and tested different slope values χ and ψ, respectively, as well as various trust region radii $r_\mathrm{i} = \beta^{(l)} - \iota^{(l)}$ and $r_\mathrm{o} = \beta^{(l)} - \xi^{(l)}$. Starting with a good initial stability center in order to avoid too many updates of it is crucial for the overall performance on the one hand. On the other hand, it is not trivial to obtain such a good initial center in advance. We first solved the LP relaxation of each instance and stored the finally obtained dual solution $D' \in \mathbb{R}^m$ and initially set $\beta_i^{(1)} = D'_i + r_\mathrm{i}$, $\forall i = 0, \ldots, m-1$. In this way, we completely avoided updates of the stability center for

Table 2: Median CPU-times in seconds for CG utilizing piecewise linear stabilization, standard CG, and two approaches using alternative dual-optimal solutions.

	stabilization	D^*	$\bar{D}$	$\hat{D}$, $Q = 10$	$h(\beta)$				$k(\beta)$	
	$(r_\mathrm{i}, r_\mathrm{o})$	-	-	-	$(0.1, 0.5)$		$(0.5, 1.0)$		$(0.1, -)$	
Set	$B \setminus (\chi, \psi)$	-	-	-	$(0.3, 1.0)$	$(0.5, 1.5)$	$(0.3, 1.0)$	$(0.5, 1.5)$	$(1.3, -)$	$(2.0, -)$
C1000	1000	9	**5**	11	8	8	9	9	8	8
	1500	41	**11**	23	22	22	26	21	27	28
	2000	164	28	**24**	90	120	142	131	115	125
	2500	374	29	**28**	264	213	239	206	182	191
E1000	1000	80	27	**18**	38	37	70	29	41	33
	1500	1013	151	**40**	211	207	351	213	180	145
	2000	3395	796	**106**	713	407	848	803	390	348
	2500	-	2672	**157**	2040	2030	2869	4268	1616	2594
T30	16	**1**	**1**	2	**1**	**1**	**1**	**1**	**1**	**1**
	30	7	**4**	7	7	7	8	8	6	7
	50	36	**11**	15	33	32	37	38	30	34
	100	326	34	**29**	282	309	270	291	252	230
T50	16	**2**	**2**	4	**2**	**2**	**2**	**2**	**2**	**2**
	30	19	**9**	15	15	14	16	14	12	12
	50	99	36	**31**	70	86	84	71	59	63
	100	1305	167	**94**	673	489	643	527	455	495

all experiments reported in Table 2[1]. Our approach based on piecewise linear penalty functions thus uses an initially optimal stability center, which is in real applications not available. This positive discrimination obviously is unfair, but gives us an idea on how this approach may work in the best case.

From Table 2 we conclude that column generation based on piecewise linear penalty functions is able to successfully reduce the necessary CPU-time compared to standard column generation and using the chosen positive discrimination is partly competitive to alternative dual-optimal solutions when simply equally distributing the dual slack ($\bar{D}$). It is, however, clearly outperformed by the more sophisticated approach initially using different dual-optimal solutions ($\hat{D}$). Since it is generally not possible to know optimal dual values in advance we conclude that stabilization by alternative dual-optimal solutions is better suited for the considered type of optimization problems. Furthermore, additional main advantages of the latter are the facts that one does not need to choose too many parameters and that it does not add further constraints or variables to the RMP.

6.1.3 Comparison to Branch-and-Cut Approach based on Layered Graphs

Table 3 compares the numbers of solved instances ($\#_{\mathrm{solved}}$), average optimality gaps ($\overline{\mathrm{gap}}$) in %, and median CPU-times (t_{total}) in seconds for branch-and-price approaches (BP) utilizing standard column generation as well as different variants of stabilized column generation based on alternative dual-optimal solutions to variants of the branch-and-cut from [46] based on layered graphs. For the latter we introduced variables for each node in the layered graph to be able to model the prize-collecting variant. Since the branch-and-

[1]For two instances an unnecessary update shifting the center by ε has been performed in the last iteration due to numerical issues. The additional time can, however, be neglected.

cut is likely to benefit from the generally better performance of CPLEX compared to SCIP we further re-implemented it using SCIP. In Table 3 $\mathrm{LG}_{\mathrm{C}}^{(+)}$ denotes the pure CPLEX implementation while $\mathrm{LG}_{\mathrm{S}}^{(+)}$ refers to the conceptually identical re-implementation using SCIP with CPLEX as LP solver.

In [32] we observed that the adaptive strategy ALF [46] indeed outperforms the pure layered graph approaches when the set of achievable resource values is large, but it nevertheless is not able to compete with our stabilized branch-and-prize. Furthermore, an improved variant of ALF is still ongoing work for which reasons we excluded it from experimental comparison here.

LG_{C} and LG_{S} use a compact MIP model including a polynomial number of connectivity inequalities, cf. [19], enough to guarantee a feasible solution, while connectivity cuts are additionally separated to strengthen the dual bounds for $\mathrm{LG}_{\mathrm{C}}^{+}$ and $\mathrm{LG}_{\mathrm{S}}^{+}$, respectively. Probing has been deactivated for CPLEX and SCIP since it often needed too much time and memory, slowing down the overall process. As for BP, we further set parameter "fastmip" to one for SCIP. Apart from that standard settings have been used.

For the branch-and-price approach using dual solutions $\hat{D}$ we do not reinitialize parameter q after branching. Reinitializing q did not significantly change the overall performance in our preliminary tests since most variables are usually generated in the root node of the branch-and-price tree. Hence, $\bar{D}$ is used in all further branching nodes.

As anticipated from the results in the previous subsections, we first observe that $\hat{D}$ clearly outperforms $\bar{D}$ and D^* with respect to all measured quantities. Regarding the number of solved instances, we conclude that $\hat{D}$, i.e. the stabilization using different dual-optimal solutions, also outperformed the layered graph approaches for all three tested settings. Irrespective whether $Q = 10$, $Q = 20$, or $Q = 30$, $\hat{D}$ could solve at least as many instances to proven optimality as the best layered graph approach for all but two settings: For set E10 with $B = 25$, $\hat{D}$ with $Q = 30$ could solve only four instances, while $\mathrm{LG}_{\mathrm{C}}^{+}$ and two other settings of $\hat{D}$ could solve all five instances. Furthermore, for instance set T99 with $B = 100$, LG_{C} could solve one more instance than the three best stabilized branch-and-price approaches which in turn solved more instances than $\mathrm{LG}_{\mathrm{C}}^{+}$, LG_{S}, and $\mathrm{LG}_{\mathrm{S}}^{+}$, respectively. The average remaining optimality gap of all three variants using $\hat{D}$, however, is slightly smaller than the one of LG_{C} even for this setting.

We further observe that the SCIP implementation of the layered graph approach is significantly slower than the one solely using CPLEX, often by a factor between five and ten. Despite these advantages of CPLEX, BP outperforms LG_{C} and $\mathrm{LG}_{\mathrm{C}}^{+}$, respectively, with respect to CPU-time on the majority of the spanning tree instances from [21] and several of the larger random instances for the Steiner tree variant. The layered graph approaches have advantages when the delays are distributed in a small interval, e.g. on sets C10 and E10, and sometimes when the delay bound is particularly tight. LG_{S} and $\mathrm{LG}_{\mathrm{S}}^{+}$ are clearly outperformed by BP on all but some rather easy instances.

Overall we conclude that the proposed stabilized branch-and-price approach utilizing alternative dual-optimal solutions outperforms the layered graph approach whenever the delay bound is not too strict and if the number of achievable delays is not too small. Furthermore, even for the few cases that could not be solved to proven optimality by the branch-and-price in the given time the resulting gap is very small while the branch-and-cut approach may yield quite large gaps if too many layers need to be considered.

6.2 Experiments on the Quota-Constrained Rooted Delay-Constrained Steiner Tree Problem

We now consider the *Quota-Constrained Rooted Delay-Constrained Steiner Tree Problem* (QRDCSTP), a variant of the RDCSTP where $T' = \emptyset$, i.e. there are no mandatory terminals, and prizes $p_t \geq 0$ are given for all terminals $t \in T$. The objective is to identify a solution yielding minimum total costs, while the sum of prizes of connected terminal nodes must be at least equal to a given quota value Θ. To the best of our knowledge, this problem variant has not been considered before.

We created sets QC and QD of benchmark instances using sets C and D of preprocessed instances for the prize collecting STP [34] originally created by Canuto et al. [8], respectively. Instances from subset A where terminal prizes have been originally created randomly from the interval $[1, 10]$ have been used.

For each original instance I, three different QRDCSTP instances I-α have been derived with edge delays r_e chosen uniformly at random from $[1, \alpha]$ with $\alpha = 10$, $\alpha = 100$, and $\alpha = 1000$, respectively, for all edges $e \in E$. The Steiner node with the smallest index is used as root node.

Reasonable lower and upper bounds $B_{\min}$ and $B_{\max}$ for B are determined as follows: $B_{\min} = \max\{B_{\min}(t) : t \in T\}$ where $B_{\min}(t)$ is the minimum delay of any path connecting the root node with terminal t. On the contrary, $B_{\max} = \max\{B_{\max}(t) : t \in T\}$ where $B_{\max}(t)$ is the delay of a minimum cost path to terminal t.

Tables 4 and 5 compare median CPU-times in seconds (t_{total}) and relative optimality gaps in percent for branch-and-price with and without stabilization and for branch-and-cut on the layered graph.

Both approaches differ from the ones used for the RDCSTP only by the quota constraint that has been added. As in Section 6.1.3 we report results for CPLEX and SCIP implementations of the branch-and-cut approach with and without additionally separating directed connectivity cuts, respectively. For each instance, we consider two different delay bounds $B = \lceil b_{\text{r}} \cdot (B_{\max} - B_{\min}) + B_{\min} \rceil$ with $b_{\text{r}} \in \{0.3, 0.6\}$, and for each delay bound two different quota values $\Theta = \lceil \theta_{\text{r}} \cdot \sum_{i \in T} p_i \rceil$ with $\theta_{\text{r}} \in \{0.3, 0.6\}$.

From Tables 4 and 5 we first observe that the layered graph approach is rather efficient on instances where the maximum delay of an edge is ten. On these instances, the CPLEX implementation frequently outperforms all branch-and-price variants. Looking at the fairer comparison of $\mathrm{LG_S}$ and $\mathrm{LG_S^+}$ to different variants of BP, the picture is not so clear anymore on these instances where the number of layers is relatively small. The performance of the layered graph approach heavily suffers from an increasing number of layers, i.e. when using instances I-100 and I-1000, respectively. Here, BP clearly outperforms the two branch-and-cut approaches using SCIP and is often competitive or superior to the CPLEX variants.

Comparing the different variants of branch-and-price, we observe that the performance trends between them are not as clear as in our experiments for the RDCSTP anymore. For some settings, BP without stabilization outperforms the other variants. We argue that the number of feasible paths may be quite restricted for particularly these instances and settings, since the corresponding graphs are significantly sparser than the ones used for the RDCSTP. Hence, already the minimal guiding using ε edge costs, i.e. choosing paths with a smaller number of edges, rather than zeros in D^* has a significant positive effect. Another hint supporting this argument is the relatively small difference between the layered graph approaches with and without directed connectivity cuts, respectively. We also note that branch-and-price obviously performs best for problems with rather tight restrictions, i.e. if the number of potential variables is small. In these cases, however, we clearly cannot hope to gain too much using stabilization techniques.

Nevertheless, stabilized branch-and-price utilizing alternative dual-optimal solutions clearly outperforms the standard approach in the majority of cases. Whether $\bar{D}$ or $\hat{D}$ performs better heavily depends on the considered instance and setting. Considering the particularly hard instances, however, the approaches based on $\hat{D}$ were able to solve slightly more instances to proven optimality in the given time. Also the number of cases where the remaining optimality gap exceeds ten percent is smaller for all approaches based on $\hat{D}$ than for $\bar{D}$.

Regarding the instances that could not be solved by all or most approaches within the given time limit, we further observe that the resulting optimality gap of BP is usually rather small. The layered graph approaches often fail to derive any meaningful primal and dual bounds yielding a gap of 100% for many of these cases. Overall we conclude that branch-and-price utilizing alternative dual-optimal solutions usually outperforms standard branch-and-price and outperforms the layered graph approach whenever the delay range of edges is not too restricted.

7 Conclusions and Future Work

In this article we described a branch-and-price approach based on a path formulation for a generic generalization of the Steiner tree problem on graphs with additional quality-of-service requirements for realized connections. As a naive column generation suffers greatly from degeneracy, our primary focus was on stabilizing this approach using alternative dual-optimal solutions. Two concrete variants have been introduced and shown to easily generalize also when considering rich problem variants involving diverse types of side constraints simultaneously. Performing a computational study on different instance sets of the RDCSTP and a quota-constrained variant, the positive impacts of our stabilization technique could be clearly observed. We analyzed and discussed major reasons of the huge speed-up and showed the general superiority over an alternative stabilization technique utilizing piecewise linear penalty functions in the considered domain. We further proved that due to the achieved speed-up, stabilized branch-and-price based on alternative dual-optimal solutions allows to solve hard instances and outperforms a layered graph approach on many occasions. It is thus at least competitive to state-of-the-art methods for the considered type of network design problems.

It should be not too difficult to generalize our results to further classes of network design problems. In particular, problems additionally considering redundancy issues or variants where no fixed root node is given but more general commodities between arbitrary node pairs shall be satisfied appear promising to look at. Utilizing alternative dual-optimal solutions, stabilized branch-and-price approaches may have the potential to yield state-of-the-art results in these areas, too.

Acknowledgements

Günther R. Raidl is supported by the *Wiener Wissenschafts-, Forschungs- und Technologiefonds* (WWTF), project number ICT10-027. Markus Leitner wants to thank Jacques Desrosiers for suggesting to use an optimal initial stability center in the piecewise linear stabilization approach for comparison reasons.

Table 3: Numbers of solved instances, average optimality gaps, and median CPU-times for branch-and-price compare to layered graph approaches.

				$\#_{\text{solved}}$									$\overline{\text{gap}}$ %									t_{total} $[s]$								
				LG_C	LG_C^+	LG_S	LG_S^+	D^*	$\bar{D}$	$\hat{D}$			LG_C	LG_C^+	LG_S	LG_S^+	D^*	$\bar{D}$	$\hat{D}$			LG_C	LG_C^+	LG_S	LG_S^+	D^*	$\bar{D}$	$\hat{D}$		
Set	B	$\lvert\bar{V}\rvert$	$\lvert\bar{E}\rvert \setminus Q$	-	-	-	-	-	-	10	20	30	-	-	-	-	-	-	10	20	30	-	-	-	-	-	-	10	20	30
C10	10	41	431	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	2	1	3	2	4	5	6
	15	41	484	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**1**	2	11	9	10	5	7	7	7
	20	41	484	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**7**	16	79	50	26	**7**	9	11	12
	25	41	484	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	102	32	254	103	62	29	18	**13**	15
E10	10	41	566	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	10	10	8	6	8	10	10
	15	41	670	**5**	**5**	**5**	**5**	4	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**18**	24	112	104	257	76	66	53	59
	20	41	670	**5**	**5**	**5**	**5**	3	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	20.2	**0.0**	**0.0**	**0.0**	**0.0**	803	**257**	2340	562	7527	1574	697	618	461
	25	41	670	3	**5**	1	**5**	1	3	**5**	**5**	4	1.9	**0.0**	3.1	**0.0**	60.3	1.2	**0.0**	**0.0**	0.2	6091	**908**	-	1369	-	3953	1105	1340	1617
C100	100	41	561	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	38	46	200	353	12	**8**	9	11	12
	150	41	572	**5**	**5**	**5**	4	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	0.2	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	379	888	1777	4458	30	**14**	**14**	**14**	16
	200	41	572	3	3	2	1	**5**	**5**	**5**	**5**	**5**	2.8	2.6	2.8	10.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5227	3056	-	-	69	16	17	**14**	20
	250	41	572	1	2	2	0	**5**	**5**	**5**	**5**	**5**	4.8	6.6	4.8	33.2	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	284	26	27	22	**15**
E100	100	41	651	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	394	419	1539	1892	104	55	**16**	23	20
	150	41	672	0	2	0	0	4	**5**	**5**	**5**	**5**	6.9	1.8	8.7	16.3	0.2	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	1184	176	**51**	59	58
	200	41	672	0	0	0	0	1	4	4	**5**	**5**	10.9	16.5	13.7	17.1	80.0	0.3	0.1	**0.0**	**0.0**	-	-	-	-	-	3446	261	196	**146**
	250	41	672	0	0	0	0	1	3	**5**	**5**	**5**	13.0	22.5	12.0	41.2	80.0	20.0	**0.0**	**0.0**	**0.0**	-	-	-	-	-	9322	491	**169**	307
C1000	1000	41	572	**5**	4	4	2	**5**	**5**	**5**	**5**	**5**	**0.0**	0.8	2.1	9.8	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	523	730	6344	-	10	**8**	11	16	15
	1500	41	589	1	0	0	0	**5**	**5**	**5**	**5**	**5**	4.6	12.4	48.7	30.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	51	**22**	23	**22**	24
	2000	41	589	0	0	0	0	**5**	**5**	**5**	**5**	**5**	10.5	14.8	15.7	48.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	180	71	62	**50**	54
	2500	41	589	0	0	0	0	**5**	**5**	**5**	**5**	**5**	67.6	32.9	-	81.7	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	332	64	**29**	31	33
E1000	1000	41	632	2	2	1	0	**5**	**5**	**5**	**5**	**5**	6.5	15.4	6.8	25.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	71	41	**18**	20	26
	1500	41	668	0	0	0	0	4	**5**	**5**	**5**	**5**	12.5	26.4	28.7	28.4	0.1	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	706	137	**34**	40	36
	2000	41	668	0	0	0	0	3	**5**	**5**	**5**	**5**	19.1	25.3	55.2	38.4	40.0	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	3730	631	139	71	**55**
	2500	41	668	0	0	0	0	0	4	**5**	**5**	**5**	-	85.7	85.7	-	80.2	0.2	**0.0**	**0.0**	**0.0**	-	-	-	-	-	4125	1004	**194**	213
T10	16	96	469	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	**0**	**0**	**0**	**0**	1	1	1
	30	100	932	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	9	7	3	2	2	3	4
	50	100	1269	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	8	7	63	58	9	**2**	4	5	6
	100	100	1695	29	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	0.7	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	99	102	667	692	38	**4**	6	7	9
T30	16	98	482	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	**0**	**0**	2	2	3	4	4
	30	100	932	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**2**	**2**	10	9	11	7	11	13	16
	50	100	1269	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**16**	**16**	127	178	54	21	21	23	30
	100	100	1695	28	27	28	24	28	**30**	**30**	**30**	**30**	0.5	2.4	0.8	5.3	0.1	**0.0**	**0.0**	**0.0**	**0.0**	144	442	1300	1519	485	60	**41**	42	56
T50	16	99	486	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	**0**	**0**	4	3	6	6	8
	30	100	932	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**3**	5	21	20	26	15	22	23	38
	50	100	1269	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	20	**14**	170	262	105	56	47	51	66
	100	100	1695	28	22	25	19	22	29	29	**30**	29	0.2	7.0	1.5	11.4	16.9	**0.0**	**0.0**	**0.0**	0.1	300	1129	1590	4110	2711	225	**134**	155	145
T70	16	99	487	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	**0**	1	6	5	8	11	14
	30	100	932	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**3**	5	17	17	36	27	30	37	58
	50	100	1269	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**20**	40	158	208	259	130	75	81	93
	100	100	1695	27	24	26	21	22	27	28	**29**	**29**	4.0	10.3	1.0	8.0	20.2	0.3	**0.1**	**0.1**	**0.1**	**338**	854	1841	5114	1811	464	356	415	446
T99	16	100	490	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0**	**0**	1	1	7	8	12	15	20
	30	100	932	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**3**	6	37	31	83	74	73	77	105
	50	100	1269	**30**	**30**	**30**	**30**	28	**30**	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**	**0.0**	**25**	32	311	428	422	368	110	107	200
	100	100	1695	**29**	25	25	22	19	27	28	28	28	0.3	8.5	1.0	17.8	30.1	3.4	**0.2**	**0.2**	**0.2**	**262**	525	1403	6646	4650	836	685	542	640

Table 4: Optimality gaps and median CPU-times for the QRDCSTP on QC instances.

						gap %									t_{total} $[s]$								
						LG_C	LG_C^+	LG_S	LG_S^+	D^*	$\bar{D}$	$\hat{D}$			LG_C	LG_C^+	LG_S	LG_S^+	D^*	$\bar{D}$	$\hat{D}$		
Instance	B	$\vert V\vert$	$\vert E\vert$	$\vert T\vert$	$\theta_\text{r} \setminus Q$	-	-	-	-	-	-	10	20	30	-	-	-	-	-	-	10	20	30
QC12-10	21	451	1953	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1	**0**	7	4	6	5	6	5	5
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	14	9	13	10	10	11	12
	27	483	2177	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2	**1**	25	31	135	46	67	59	48
					60	**0.0**	**0.0**	**0.0**	10.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**3**	5	93	-	73	20	17	15	19
QC12-100	171	451	1892	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4	**3**	41	37	6	5	5	7	7
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3	**2**	46	61	14	15	15	16	19
	214	479	2143	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	29	**17**	498	514	62	20	29	28	36
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**20**	28	1483	1503	61	30	26	26	40
QC12-1000	1907	472	2107	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	80	58	2129	2214	48	**20**	22	34	24
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	68	68	3053	2677	**12**	**12**	13	17	**12**
	2436	482	2169	9	30	**0.0**	**0.0**	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	877	740	-	-	213	**71**	85	84	86
					60	**0.0**	**0.0**	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1233	1117	-	-	64	**23**	26	26	33
QC13-10	27	471	2108	70	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	16	**10**	271	50	140	141	155	131	184
					60	**0.0**	**0.0**	**0.0**	**0.0**	0.9	0.9	0.9	0.9	0.9	25	**10**	55	120	-	-	-	-	-
	37	472	2112	70	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**39**	72	281	721	9605	234	4561	4568	3911
					60	**0.0**	**0.0**	**0.0**	**0.0**	1.0	1.0	2.0	1.0	1.0	84	**71**	219	561	-	-	-	-	-
QC13-100	228	472	2109	70	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	178	**117**	2270	9940	1981	1134	4875	1089	1457
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	235	268	1089	2497	143	**94**	128	166	175
	348	472	2109	70	30	**0.0**	24.0	9.8	88.4	2.5	**0.0**	**0.0**	2.5	**0.0**	**2449**	-	-	-	-	6895	6039	-	4303
					60	**0.0**	58.1	4.0	70.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3522	-	-	-	585	456	**357**	428	378
QC13-1000	2444	472	2106	70	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	5115	**1831**	4944	9583	9573
					60	68.1	68.1	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	2395	**2204**	7672	7943	3567
	3726	472	2106	70	30	-	-	-	-	**2.6**	**2.6**	**2.6**	**2.6**	**2.6**	-	-	-	-	-	-	-	-	-
					60	-	-	-	-	1.0	**0.0**	**0.0**	1.0	1.0	-	-	-	-	-	9094	**8783**	-	-
QC14-10	23	465	2068	100	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.4	**0.0**	**0.0**	**0.0**	4	**3**	82	108	4997	-	4721	352	5179
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**4**	**4**	68	90	574	142	126	1061	828
	33	466	2074	100	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	25	**18**	73	80	530	205	193	219	220
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	26	**21**	69	337	407	246	248	303	275
QC14-100	203	466	2073	100	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2.4	**0.0**	4.7	**0.0**	40	**22**	827	1439	5669	-	2368	-	6362
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	37	**26**	274	258	105	114	135	144	143
	298	466	2075	100	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	337	328	260	232	978	652	717	527	**184**
					60	**0.0**	**0.0**	5.9	5.9	**0.0**	**0.0**	**0.0**	**0.0**	0.9	725	626	-	-	270	**164**	1502	3877	-
QC14-1000	2088	466	2074	100	30	**0.0**	**0.0**	-	-	**0.0**	2.3	**0.0**	**0.0**	**0.0**	1868	5294	-	-	2043	-	**135**	216	227
					60	**0.0**	**0.0**	-	71.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5084	6969	-	-	231	**202**	205	234	259
	3153	466	2074	100	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	1789	1749	**1129**	2974	2661
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	1213	701	**474**	571	562
QC15-10	23	406	1868	166	30	**0.0**	**0.0**	**0.0**	2.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**6**	8	36	-	1507	1804	1416	1251	922
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**9**	**9**	186	162	1021	1343	165	1626	1458
	29	406	1868	166	30	**0.0**	**0.0**	**0.0**	**0.0**	2.1	**0.0**	2.1	2.1	2.1	48	**28**	190	202	-	6610	-	-	-
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**84**	118	459	632	1063	497	349	4582	3818
QC15-100	218	406	1860	166	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	198	**196**	549	2782	4535	4496	1983	1772	2491
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	108	**62**	373	202	203	426	160	229	1416
	311	406	1860	166	30	**0.0**	**0.0**	**0.0**	91.0	**0.0**	**0.0**	**0.0**	**0.0**	2.2	**1831**	3344	3900	-	3202	2285	8711	9082	-
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	763	1322	5599	4166	628	397	**276**	330	364
QC15-1000	2481	406	1859	166	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	262	**221**	1936	2460	5293
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	288	333	**218**	280	220
	3562	406	1859	166	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	5198	3290	8142	**3140**	3931
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	4817	1193	**544**	622	580
QC17-10	16	494	4351	8	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3	**2**	14	9	114	59	51	60	43
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	6	**2**	36	14	169	187	128	240	92
	23	498	4607	8	30	**0.0**	**0.0**	**0.0**	25.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	79	**20**	764	-	956	249	286	233	217
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	96	**30**	1319	536	1657	1120	761	596	843
QC17-100	128	497	4294	8	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	81	87	1173	1520	241	72	87	**65**	73
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	141	183	1698	2866	598	207	**110**	141	158
	204	498	4607	8	30	**0.0**	**0.0**	-	78.3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2236	1789	-	-	1905	318	**263**	493	530
					60	**0.0**	50.0	**0.0**	60.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1418	-	9673	-	5710	**443**	1140	920	2995
QC17-1000	1201	498	4206	8	30	**0.0**	**0.0**	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2481	2009	-	-	171	59	65	59	**55**
					60	**0.0**	29.4	64.3	64.3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3891	-	-	-	909	357	**302**	426	465
	1997	498	4606	8	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	1203	441	382	487	**311**
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	6111	1423	2899	2345	**1162**
QC18-10	14	469	4251	52	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**2**	**2**	17	17	288	296	258	274	212
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**2**	**2**	14	21	811	504	501	451	431
	19	469	4403	52	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**5**	**5**	68	66	896	429	316	402	403
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	6	**5**	165	193	209	186	170	258	223
QC18-100	154	469	4396	52	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	167	**132**	2699	2582	396	386	268	509	607
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**257**	292	892	8531	2527	746	1868	2144	1059
	234	469	4401	52	30	**0.0**	**0.0**	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**304**	319	-	-	1391	1380	988	1308	358
					60	**0.0**	**0.0**	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2498	3379	-	-	**366**	376	371	419	451
QC18-1000	1269	469	3727	52	30	**0.0**	**0.0**	-	14.3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	462	358	-	-	154	153	**134**	174	176
					60	**0.0**	**0.0**	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1095	1654	-	-	121	**96**	128	164	178
	1816	469	4397	52	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	747	539	426	**366**	462
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	**194**	366	278	402	306
QC19-10	14	429	3522	55	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	2	3	52	46	46	43	35
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**2**	**2**	60	33	109	63	91	108	81
	19	430	3771	55	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5	**4**	9	7	43	50	37	54	44
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	10.0	**0.0**	**0.0**	**0.0**	12	**4**	113	134	4802	-	4209	6733	6099
QC19-100	114	429	3126	55	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	31	**20**	51	50	35	36	38	39	38
					60	**0.0**	**0.0**	**0.0**	14.3	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**157**	399	3147	-	222	244	530	612	516
	164	430	3666	55	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	237	187	385	252	48	68	**40**	62	50
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	683	748	7923	6641	138	154	**137**	143	162
QC19-1000	1141	430	3286	55	30	**0.0**	**0.0**	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1849	2006	-	1536	**38**	52	41	54	39
					60	**0.0**	54.2	-	88.9	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5256	-	-	-	794	1387	1036	**755**	1768
	1788	430	3709	55	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	52	48	**42**	50	61
					60	-	-	-	-	10.0	10.0	**0.0**	10.0	10.0	-	-	-	-	-	-	**7407**	-	-

Table 5: Optimality gaps and median CPU-times for the QRDCSTP on QD instances.

						gap %									t_{total} [s]								
						LG_{C}	LG_{C}^{+}	LG_{S}	LG_{S}^{+}	D^*	$\bar{D}$	$\hat{D}$			LG_{C}	LG_{C}^{+}	LG_{S}	LG_{S}^{+}	D^*	$\bar{D}$	$\hat{D}$		
Instance	B	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	$\theta_{\text{r}} \setminus Q$	-	-	-	-	-	-	10	20	30	-	-	-	-	-	-	10	20	30
QD12-10	23	951	4373	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	2	**1**	31	21	30	15	15	18	23
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	3	**1**	63	89	169	64	76	103	113
	34	991	4631	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	16	**12**	255	465	175	32	28	38	50
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**47**	62	1694	1886	1540	538	199	425	180
QD12-100	214	977	4544	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	22	20	19	26	32	**13**	**13**	**13**	17
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	29	**25**	3642	2652	123	58	44	63	82
	325	991	4633	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	911	748	1139	1244	102	29	**23**	24	40
					60	**0.0**	**0.0**	-	53.2	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	4715	2389	-	-	1788	**107**	614	226	490
QD12-1000	1652	879	3770	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	18	18	266	343	30	21	**13**	19	18
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	30	26	155	177	**17**	20	22	21	26
	2286	987	4611	9	30	**0.0**	**0.0**	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1439	1759	-	-	63	34	**25**	46	36
					60	**0.0**	-	-	26.6	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1250	-	-	-	24	**19**	21	39	21
QD13-10	30	966	4569	143	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	26	**17**	207	360	3860	2982	2790	4322	4533
					60	**0.0**	**0.0**	**0.0**	**0.0**	0.5	0.5	0.5	0.5	0.5	**32**	68	160	470	-	-	-	-	-
	41	966	4569	143	30	**0.0**	**0.0**	**0.0**	**0.0**	2.8	4.2	**0.0**	2.8	2.8	171	**92**	347	3488	-	-	1896	-	-
					60	**0.0**	**0.0**	**0.0**	**0.0**	1.1	1.1	0.6	1.1	**0.0**	117	**97**	1004	1358	-	-	-	-	9598
QD13-100	228	965	4561	143	30	**0.0**	**0.0**	88.8	88.8	1.3	1.3	**0.0**	1.3	**0.0**	**618**	1285	-	-	-	-	2611	-	6750
					60	**0.0**	**0.0**	**0.0**	9.8	**0.0**	0.5	0.5	0.5	**0.0**	**1148**	1686	5819	-	4522	-	-	-	9512
	342	966	4566	143	30	**0.0**	61.5	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	5201	-	-	-	2033	7263	1288	**1166**	1621
					60	70.4	70.4	-	-	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**	-	-	-	-	-	-	-	-	-
QD13-1000	2230	966	4558	143	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	2935	**1611**	1873	2330	2689
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	961	**729**	803	925	1196
	3409	966	4558	143	30	-	-	-	-	**2.8**	**2.8**	**2.8**	**2.8**	**2.8**	-	-	-	-	-	-	-	-	-
					60	-	-	-	-	**0.0**	**0.0**	1.7	**0.0**	**0.0**	-	-	-	-	9183	**2839**	-	3296	9971
QD14-10	29	946	4493	207	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	72	**52**	502	527	1384	1327	998	1251	1573
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**92**	352	327	3540	2986	3090	7907	2470	2577
	41	946	4493	207	30	**0.0**	**0.0**	**0.0**	**0.0**	35.3	**0.0**	**0.0**	**0.0**	**0.0**	**525**	2094	1426	4324	-	4880	4649	3307	4906
					60	**0.0**	**0.0**	**0.0**	**0.0**	-	1.4	1.4	1.4	**0.0**	**281**	327	2872	5224	-	-	-	-	8053
QD14-100	260	946	4492	207	30	**0.0**	59.8	8.0	88.9	**0.0**	**0.0**	0.9	**0.0**	**0.0**	**2930**	-	-	-	7070	7064	-	8492	7816
					60	**0.0**	**0.0**	-	73.4	**0.0**	0.4	**0.0**	**0.0**	**0.0**	**2381**	4844	-	-	7943	-	3515	3777	3944
	381	946	4492	207	30	-	88.8	-	-	**1.1**	**1.1**	**1.1**	**1.1**	**1.1**	-	-	-	-	-	-	-	-	-
					60	-	-	-	-	-	-	**0.4**	-	**0.4**	-	-	-	-	-	-	-	-	-
QD14-1000	2209	946	4492	207	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	**369**	455	702	841	998
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	0.4	-	-	-	-	**1409**	7380	3718	3357	-
	3194	946	4492	207	30	-	-	-	-	**1.1**	**1.1**	8.0	**1.1**	8.0	-	-	-	-	-	-	-	-	-
					60	-	-	-	-	0.9	**0.0**	0.9	0.4	0.9	-	-	-	-	-	**9753**	-	-	-
QD15-10	26	832	4165	348	30	**0.0**	**0.0**	**0.0**	**0.0**	27.2	27.2	27.7	27.2	-	**45**	148	187	1015	-	-	-	-	-
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	6.2	-	**0.0**	3.0	**69**	83	193	539	6291	-	-	1960	-
	37	832	4165	348	30	**0.0**	**0.0**	**0.0**	**0.0**	-	11.8	58.0	**0.0**	**0.0**	**456**	522	1233	5685	-	-	-	4108	3172
					60	**0.0**	18.7	**0.0**	-	-	-	7.3	-	-	**1408**	-	7901	-	-	-	-	-	-
QD15-100	218	832	4161	348	30	**0.0**	48.8	-	91.0	-	3.5	49.5	6.0	6.8	**1587**	-	-	-	-	-	-	-	-
					60	**0.0**	**0.0**	**0.0**	0.7	1.7	7.1	6.1	4.3	4.3	**882**	3559	7280	-	-	-	-	-	-
	323	832	4161	348	30	-	-	-	-	-	**17.1**	-	-	-	-	-	-	-	-	-	-	-	-
					60	-	75.0	-	-	-	-	-	**26.9**	-	-	-	-	-	-	-	-	-	-
QD15-1000	2415	832	4160	348	30	-	-	-	-	**0.0**	17.5	**0.0**	**0.0**	-	-	-	-	-	**1454**	-	2931	1609	-
					60	-	-	-	-	-	6.7	**0.4**	**0.4**	**0.4**	-	-	-	-	-	-	-	-	-
	3385	832	4160	348	30	-	-	-	-	-	-	**0.0**	2.0	**0.0**	-	-	-	-	-	-	6575	-	**4319**
					60	-	-	-	-	-	-	**1.1**	7.4	7.4	-	-	-	-	-	-	-	-	-
QD17-10	14	970	7712	9	30	**0.0**	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**1**	-	6	6	153	88	64	59	82
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**1**	**1**	24	37	122	81	62	61	76
	19	999	10412	9	30	**0.0**	**0.0**	**0.0**	12.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	12	**9**	293	-	1060	224	267	266	428
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	13	**12**	886	342	327	309	90	324	119
QD17-100	96	986	6438	9	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**51**	62	626	873	179	124	158	145	159
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**57**	142	874	967	945	399	458	599	531
	145	999	10379	9	30	**0.0**	**0.0**	30.0	30.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	138	157	-	-	583	396	118	**99**	141
					60	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	510	312	2298	8512	154	**72**	**72**	99	134
QD17-1000	1396	999	10355	9	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	134	**53**	75	88	83
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	336	**69**	105	133	115
	2178	999	10391	9	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	9398	1835	2028	4884	**1251**
					60	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	952	127	**108**	174	170
QD18-10	18	944	9726	111	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**14**	15	1092	1168	751	754	936	1028	862
					60	**0.0**	**0.0**	**0.0**	**0.0**	23.2	23.2	**0.0**	**0.0**	**0.0**	**49**	67	488	1655	-	-	1217	1351	2106
	27	944	9726	111	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	7.1	7.1	7.1	**450**	1719	3042	7056	5734	3913	-	-	-
					60	**0.0**	41.5	**0.0**	20.0	-	-	-	-	-	**584**	-	3376	-	-	-	-	-	-
QD18-100	139	944	9700	111	30	**0.0**	**0.0**	**0.0**	94.3	**0.0**	**0.0**	**0.0**	12.5	**0.0**	**2630**	3728	5684	-	3262	5384	4682	-	4699
					60	**0.0**	42.3	5.2	77.6	-	17.4	1.7	1.7	1.7	**1392**	-	-	-	-	-	-	-	-
	222	944	9711	111	30	-	-	-	-	97.5	7.1	95.5	**0.0**	95.5	-	-	-	-	-	-	-	**9484**	-
					60	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
QD18-1000	1315	944	9706	111	30	-	-	-	-	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	-	-	-	-	**511**	635	756	764	919
					60	-	-	-	-	9.7	**1.8**	**1.8**	**1.8**	**1.8**	-	-	-	-	-	-	-	-	-
	2143	944	9726	111	30	-	-	-	-	**0.0**	23.5	23.5	**0.0**	23.5	-	-	-	-	6462	-	-	**2905**	-
					60	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
QD19-10	17	897	9234	147	30	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	16	**12**	727	2834	7396	9799	7982	6557	7379
					60	**0.0**	**0.0**	**0.0**	**0.0**	62.6	75.8	1.6	1.6	1.6	48	**40**	440	1419	-	-	-	-	-
	25	897	9234	147	30	**0.0**	**0.0**	**0.0**	68.9	**0.0**	88.3	**0.0**	88.4	88.3	**648**	761	1688	-	7249	-	2444	-	-
					60	**0.0**	54.0	**0.0**	77.6	-	-	-	-	-	**1578**	-	6649	-	-	-	-	-	-
QD19-100	169	897	9197	147	30	16.7	51.6	-	-	5.9	**0.0**	93.2	93.8	93.2	-	-	-	-	-	**3106**	-	-	-
					60	65.1	56.4	-	78.9	-	-	-	**8.7**	63.4	-	-	-	-	-	-	-	-	-
	274	897	9197	147	30	-	-	-	-	-	-	-	-	**92.5**	-	-	-	-	-	-	-	-	-
					60	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
QD19-1000	1474	897	9150	147	30	-	-	-	-	88.5	**6.3**	16.7	11.8	16.7	-	-	-	-	-	-	-	-	-
					60	-	-	-	-	-	-	4.3	4.3	**0.0**	-	-	-	-	-	-	-	-	**5440**
	2296	897	9171	147	30	-	-	-	-	-	**6.7**	**6.7**	**6.7**	**6.7**	-	-	-	-	-	-	-	-	-
					60	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

References

[1] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] H. B. Amor and J. Desrosiers. A proximal trust-region algorithm for column generation stabilization. *Computers & Operations Research*, 33:910–927, 2006.

[3] H. B. Amor, J. Desrosiers, and J. M. V. Carvalho. Dual-optimal inequalities for stabilized column generation. *Operations Research*, 54(3):454–463, 2006.

[4] H. B. Amor, J. Desrosiers, and A. Frangioni. On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157:1167–1184, 2009.

[5] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh, and P. H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998.

[6] F. Bauer and A. Varma. Degree-constrained multicasting in point-to-point networks. In *INFOCOM '95. Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Bringing Information to People. Proceedings. IEEE*, pages 369–376 vol.1, 1995.

[7] O. Briant, C. Lemaréchal, P. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck. Comparison of bundle and classical column generation. *Mathematical Programming*, 113(2):299–344, 2008.

[8] 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.

[9] A. Costa, J. Cordeau, and B. Gendron. Benders, metric and cutset inequalities for multicommodity capacitated network design. *Computational Optimization and Applications*, 42:371–392, 2009.

[10] A. M. Costa, J. Cordeau, and G. Laporte. Steiner tree problem with profits. *INFOR*, 44:99–115, 2006.

[11] J. M. V. de Carvalho. Using extra dual cuts to accelerate convergence in column generation. *INFORMS Journal on Computing*, 17(2):175–182, 2005.

[12] G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors. *Column Generation*. Springer, 2005.

[13] J. Desrosiers and M. E. Lübbecke. Branch-price-and-cut algorithms. In J. J. Cochran et al., editors, *Wiley Encyclopedia of Operations Research and Management Science*. Wiley, 2011.

[14] S. E. Dreyfus and R. A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1971.

[15] O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1–3):229–237, 1999.

[16] I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.

[17] M. Fernandes, L. Gouveia, and S. Voß. Determining hop-constrained spanning trees with repetitive heuristics. In *Proceedings of 6th International Conference on Decision Support for Telecommunications ands Information Society*, Warsaw, Poland, 2007.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[19] L. Gouveia. Using hop-indexed models for constrained spanning and Steiner tree models. In B. Sanso and P. Soriano, editors, *Telecommunications Network Planning*, pages 21–32. Kluwer Academic Publishers, 1999.

[20] L. Gouveia, A. Paias, and D. Sharma. Local search heuristics for the hop-constrained minimum spanning tree problem. In B. Fortz, editor, *Proceedings of International Network Optimization Conference*, Spa, Belgium, 2007.

[21] L. Gouveia, A. Paias, and D. Sharma. Modeling and solving the rooted distance-constrained minimum spanning tree problem. *Computers & Operations Research*, 35(2):600–613, 2008.

[22] L. Gouveia, L. Simonetti, and E. Uchoa. Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. *Mathematical Programming*, 128(1):123–148, 2011.

[23] M. Haouari, S. B. Jayeb, and H. D. Sherali. The prize collecting Steiner tree problem: Models and Lagrangian dual optimization approaches. *Computational Optimization and Applications*, 40(1):13–39, 2008.

[24] M. Haouari and J. C. Siala. A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. *Computers & Operations Research*, 33:1274–1288, 2006.

[25] K. Holmberg and D. Yuan. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48(3):461–481, 2000.

[26] D. S. Johnson, M. Minkoff, and S. Phillips. The prize collecting Steiner tree problem: Theory and practice. In *Proceedings of the 11th ACM-SIAM Symposium on Discrete Algorithms*, pages 760–769, 2000.

[27] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos. Multicasting for multimedia applications. In *INFOCOM '92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, pages 2078–2085, 1992.

[28] V. Leggieri, M. Haouari, and C. Triki. An exact algorithm for the Steiner tree problem with delays. *Electronic Notes in Discrete Mathematics*, 36:223–230, 2010.

[29] M. Leitner and G. R. Raidl. Strong lower bounds for a survivable network design problem. *Electronic Notes in Discrete Mathematics*, 36:295–302, 2010.

[30] M. Leitner, G. R. Raidl, and U. Pferschy. Accelerating column generation for a survivable network design problem. In M. G. Scutellà et al., editors, *Proceedings of the International Network Optimization Conference 2009*, Pisa, Italy, 2009.

[31] M. Leitner, G. R. Raidl, and U. Pferschy. Branch-and-price for a survivable network design problem. Technical Report TR 186–1–10–02, Vienna University of Technology, Vienna, Austria, 2010. submitted to Networks.

[32] M. Leitner, M. Ruthmair, and G. R. Raidl. Stabilized branch-and-price for the rooted delay-constrained Steiner tree problem. In J. Pahl et al., editors, *Network Optimization: 5th International Conference, INOC 2011*, volume 6701 of *LNCS*, pages 124–138, Hamburg, Germany, 2011. Springer.

[33] M. Leitner, M. Ruthmair, and G. R. Raidl. Stabilized column generation for the rooted delay-constrained Steiner tree problem. In *Proceedings of the VII ALIO/EURO – Workshop on Applied Combinatorial Optimization*, pages 250–253, Porto, Portugal, 2011.

[34] I. Ljubić, 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.

[35] M. E. Lübbecke. Column generation. In J. J. Cochran et al., editors, *Wiley Encyclopedia of Operations Research and Management Science*. Wiley, 2011.

[36] M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[37] P. Manyem and M. Stallmann. Some approximation results in multicasting. Technical Report TR-96-03, North Carolina State University, 1996.

[38] R. E. Marsten, W. W. Hogan, and J. W. Blankenship. The BOXSTEP method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.

[39] S. C. Narula and C. A. Ho. Degree-constrained minimum spanning tree. *Computers & Operations Research*, 7(4):239–249, 1980.

[40] A. Pessoa, E. Uchoa, M. de Aragão, and R. Rodrigues. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, 2(3):259–290, 2010.

[41] R. Qu, Y. Xu, and G. Kendall. A variable neighborhood descent search algorithm for delay-constrained least-cost multicast routing. In T. Stützle, editor, *Learning and Intelligent Optimization*, volume 5851 of *LNCS*, pages 15–29. Springer, 2009.

[42] M. B. Rosenwein and R. T. Wong. A constrained Steiner tree problem. *European Journal of Operational Research*, 81(2):430–439, 1995.

[43] L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007.

[44] M. Ruthmair and G. R. Raidl. A kruskal-based heuristic for the rooted delay-constrained minimum spanning tree problem. In R. Moreno-Díaz et al., editors, *Proceedings of the 12th International Conference on Computer Aided Systems Theory*, volume 5717 of *LNCS*, pages 713–720. Springer, 2009.

[45] M. Ruthmair and G. R. Raidl. Variable neighborhood search and ant colony optimization for the rooted delay-constrained minimum spanning tree problem. In R. Schaefer et al., editors, *Proceedings of the 11th International Conference on Parallel Problem Solving from Nature: Part II*, volume 6239 of *LNCS*, pages 391–400. Springer, 2010.

[46] M. Ruthmair and G. R. Raidl. A layered graph model and an adaptive layers framework to solve delay-constrained minimum tree problems. In O. Günlük and G. Woeginger, editors, *Proceedings of the 15th Conference on Integer Programming and Combinatorial Optimization (IPCO XV)*, volume 6655 of *LNCS*, pages 376–388. Springer, 2011.

[47] F. Vanderbeck. Implementing mixed integer column generation. In Desaulniers et al. [12], pages 331–358.

[48] S. Voß. The Steiner tree problem with hop constraints. *Annals of Operations Research*, 86(0):321–345, 1999.

[49] P. Wentges. Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming. *International Transactions in Operational Research*, 4(2):151–162, 1997.

[50] Y. Xu and R. Qu. A GRASP approach for the delay-constrained multicast routing problem. In *Proceedings of the 4th Multidisplinary International Scheduling Conference*, pages 93–104, Dublin, Ireland, 2009.

[51] Y. Xu and R. Qu. A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems. *Applied Intelligence*, pages 1–13, 2010.