Stabilized Branch-and-Price for the Rooted Delay-Constrained Steiner Tree Problem

Markus Leitner, Mario Ruthmair, and Günther R. Raidl

Abstract We consider the rooted delay-constrained Steiner tree problem which arises for example in the design of centralized multicasting networks where quality of service constraints are of concern. We present a mixed integer linear programming formulation based on the concept of feasible paths which has already been considered in the literature for the spanning tree variant. Solving its linear relaxation by column generation has, however, been regarded as computationally not competitive. In this work, we study various possibilities to speed-up the solution of our model by stabilization techniques and embed the column generation procedure in a branch-and-price approach in order to compute proven optimal solutions. Computational results show that the best among the resulting stabilized branch-and-price variants outperforms so-far proposed methods.

1 Introduction

When designing a communication network with a central server broadcasting or multicasting information to all or some of the participants of the network, some applications such as video conferences require a limitation of the maximal delay from the server to each client. Beside this delay-constraint minimizing the cost of establishing the network is in most cases an important design criterion. As another example, consider a package shipping organization with a central depot guaranteeing its customers a delivery within a specified time horizon. Naturally the organization aims at minimizing the transportation costs but at the same time has to hold its promise of being in time. Such network design problems can be modeled using an NP-hard combinatorial optimization problem called *rooted delay-constrained Steiner tree problem (RDCSTP)* [14]. The objective is to find a min-

Markus Leitner, Mario Ruthmair, Günther R. Raidl
Institute of Computer Graphics and Algorithms, Vienna University of Technology, Favoritenstraße 9-11, 1040 Vienna, Austria, e-mail: {leitner|ruthmair|raidl}@ads.tuwien.ac.at

imum cost Steiner tree on a given graph with the additional constraint that the total delay along each path from a specified root node to any other required node must not exceed a given delay bound.

More formally, we are given an undirected graph $G = (V,E)$ with node set V, edge set E, a fixed root node $s \in V$, a set $T \subseteq V \setminus \{s\}$ of terminal or required nodes, a set $S = V \setminus (T \cup \{s\})$ of optional Steiner nodes, a cost function $c : E \to \mathbb{Z}^+$, a delay function $d : E \to \mathbb{Z}^+$, and a delay bound $B \in \mathbb{Z}^+$. A feasible solution to the RDCSTP is a Steiner tree $G_S = (V_S, E_S),\ s \in V_S,\ T \subset V_S \subseteq V,\ E_S \subseteq E$, satisfying the constraints $\sum_{e \in p_S(t)} d_e \leq B,\ \forall t \in T$, where $p_S(t) \subseteq E$ denotes the edge set of the unique path from root s to terminal t. An optimal solution $G_S^* = (V_S^*, E_S^*)$ is a feasible solution with minimum costs $c(G_S^*) = \sum_{e \in E_S^*} c_e$.

After discussing existing related work in Section 2 we describe a mixed integer linear programming (MIP) formulation involving exponentially many path variables as well as its solving by branch-and-price in Section 3. Section 4 details two different column generation stabilization techniques. Our computational results in Section 5 show that the best among the resulting stabilized branch-and-price variants outperforms so-far proposed methods. Finally, we conclude in Section 6 and briefly sketch potential future work.

2 Previous & Related Work

Kompella et al. [14] introduced the RDCSTP, proved its NP-hardness and presented a construction heuristic based on the algorithm by Kou et al. [15] for the Steiner tree problem (STP) on graphs. Manyem et al. [21] showed that the problem is not in APX. There are many recent publications dedicated to this problem and its more special variants. Several metaheuristics have been applied to the RDCSTP, such as tabu-search [29], GRASP [30, 33], path-relinking [11], variable neighborhood descent (VND) [24], and variable neighborhood search (VNS) [33]. A hybrid algorithm in [34] combines scatter search with tabu-search, VND, and path-relinking. More heuristic approaches can be found for the spanning tree variant with $T = V \setminus \{s\}$, e.g. a GRASP and a VND in [26] and an ant colony optimization and a VNS in [27]. Furthermore, preprocessing methods are presented in [27] to reduce the size of the input graph significantly in order to speed up the solving process.

Exact methods based on integer linear programming (ILP) have been explored by Leggieri et al. [16] who describe a compact extended node-based formulation using lifted Miller-Tucker-Zemlin inequalities. Since the used Big-M inequalities usually yield rather weak linear programming (LP) relaxation bounds this formulation is improved by directed connection cuts. Several ILP approaches for the spanning tree variant have been examined by Gouveia et al. in [12] based on a path formulation solved by three different methods. Standard column generation turns out to be computationally inefficient while a Lagrangian relaxation approach together with a fast primal heuristic exhibits better performance. The third approach reformulates the constrained shortest path problem for each node on a layered graph and solves it us-

ing a multi commodity flow (MCF) formulation. Since the size of the layered graph and therefore the efficiency of the according model heavily depends on the number of achievable discrete delay values, this approach can in practice only be used for instances in which this number is quite restricted. Additionally an MCF model usually suffers in practice from the huge amount of flow variables used, 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 [28] not just the constrained shortest path problem but the whole RDCSTP is modeled on a layered graph which reduces to solving the classical STP on this graph. The acyclicity of the layered graph allows to eliminate sub-tours using a compact formulation for the STP without additional variables. However, the well-known directed cut formulation on this graph with an exponential number of constraints 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. [13] for the *hop-constrained minimum spanning tree problem* where $d_e = 1, \forall e \in E$, 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 huge set of achievable delay values, a strategy based on iteratively solving smaller layered graphs is presented in [28] obtaining lower and upper bounds to the optimal 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 [20] proposed stabilized column generation approaches for the RDCSTP. The current article significantly extends this work by embedding column generation in a branch-and-bound approach to compute proven optimal solutions, describing an additional pricing strategy and many other aspects in more detail. We also present more extensive results including a comparison to the above mentioned layered graph approaches.

3 Branch-and-Price

In this section we present the details of a branch-and-price approach for solving the RDCSTP, which is based on a MIP formulation utilizing variables corresponding to feasible paths for each terminal. This model is a straightforward modification of the one discussed by Gouveia et al. [12] for the spanning tree variant of the RDCSTP, i.e. for the case of $T = V \setminus \{s\}$. Our directed formulation uses an arc set A containing an arc (s, j) for each edge $\{s, j\} \in E$ incident to the root node and two oppositely directed arcs (i, j), (j, i) for all other edges $\{i, j\} \in E$, $i, j \neq s$. Note that we assume the edge cost and delay functions to be correspondingly defined on the set of arcs, too, i.e. $c_{ij} = c_e$ and $d_{ij} = d_e, \forall (i, j) \in A,\ e = \{i, j\} \in E$.

The *integer master problem* (IMP) defined by (1)–(6) is based on variables $x_{ij} \in \{0, 1\}, \forall (i, j) \in A$, indicating which arcs are included in the directed solution. Each such directed solution must form an outgoing arborescence rooted at node s. We

further use path variables $\lambda_p \in \{0,1\}, \forall p \in P = \bigcup_{t \in T} P_t$, where $P_t \subseteq 2^A$ is the set of feasible paths from the root node s to terminal t. A path $p \in P_t$ to terminal $t \in T$, which is represented by its arc set, is feasible if and only if it satisfies the delay bound, i.e. $\sum_{(i,j) \in p} d_{ij} \leq B$. Variable λ_p is set to one if path $p \in P$ is realized. Dual variables are given in parenthesis in model (1)–(6).

$$\text{(1)} \quad \text{(IMP)} \quad \min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

$$\text{(2)} \quad \text{s.t.} \quad \sum_{p \in P_t} \lambda_p \geq 1 \qquad (\mu_t) \quad \forall t \in T$$

$$\text{(3)} \quad x_{ij} - \sum_{p \in P_t | (i,j) \in p} \lambda_p \geq 0 \qquad (\pi^t_{ij}) \quad \forall t \in T,\ \forall (i,j) \in A$$

$$\text{(4)} \quad \sum_{(i,j) \in A} x_{ij} \leq 1 \qquad (\gamma_j) \quad \forall j \in V$$

$$\text{(5)} \quad x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

$$\text{(6)} \quad \lambda_p \geq 0 \qquad \forall p \in P$$

The convexity constraints (2) ensure that at least one path is realized for each terminal, while the coupling constraints (3) link paths to the corresponding arc variables. Inequalities (4) restrict the in-degree of each node and thus together with inequalities (2) and (3) ensure that the directed solution is an arborescence with root s. Given strictly positive edge costs, removing inequalities (4) would also yield a valid model. We did nevertheless include them to stay consistent with the model by Gouveia et al. [12]. Further note that only lower bounds are given for variables λ_p, $\forall p \in P$, in inequalities (6). These variables will become automatically integral due to the remaining inequalities.

Since the number of feasible paths for each terminal $t \in T$ and thus the total number of variables in the model is in general exponentially large, we cannot solve the IMP directly. Hence we embed delayed column generation – see e.g. [5, 7] – in a branch-and-bound procedure to solve the IMP, i.e. we apply branch-and-price. Branching is performed on arc variables $x_{ij}, \forall (i,j) \in A$. The *restricted master problem* (RMP) which then needs to be solved in each node of the branch-and-bound tree is defined by considering only a subset $\tilde{P}_t \subseteq P_t$, $\tilde{P}_t \neq \emptyset$, $\forall t \in T$, of path variables and by replacing the integrality conditions on arcs (5) by $x_{ij} \geq 0, \forall (i,j) \in A$. Further variables are added on demand according to the solution of the pricing subproblem which will be discussed in the following.

3.1 The Pricing Subproblem

Let $\tilde{P}$ denote the set of paths for which corresponding variables have already been included in the RMP. We further denote by $\mu_t \geq 0$, $\forall t \in T$, the dual variables associated to the convexity constraints (2) and by $\pi^t_{ij} \geq 0$, $\forall t \in T$, $\forall (i,j) \in A$, the dual

variables associated to the coupling constraints (3). In the pricing subproblem, we need to identify at least one path variable λ_p, $p \in P \setminus \tilde{P}$, yielding negative reduced costs $\bar{c}_p = -\mu_t + \sum_{(i,j)\in p} \pi_{ij}^t$ or prove that no such variable exists.

Thus, the pricing subproblem is formally defined as

$$(t^*, p^*) = \text{argmin}_{t\in T, p\in P_t} -\mu_t + \sum_{(i,j)\in p} \pi_{ij}^t. \tag{7}$$

It can be solved by computing a resource constrained shortest path on a graph (V,A) with non-negative arc costs π_{ij}^t, $\forall (i,j) \in A$, for each terminal $t \in T$. Computing a minimum cost resource constrained shortest path between two nodes is NP-hard in the weak sense [10] and can thus be solved in pseudo-polynomial time, see [9] for a survey. We use the dynamic programming based algorithm from [12] in our implementation which has computational complexity $O(B \cdot |A|)$.

For solving the RMP of the currently considered branch-and-bound tree node, we need to add path variables and resolve the RMP as long as at least one path variable $p \in P$ with negative reduced costs $\bar{c}_p$ exists. We compare two pricing strategies, which both require a single run of the dynamic program from [12] for each terminal $t \in T$. In the first approach we add the variable corresponding to the cheapest feasible path for each terminal in case it has negative reduced costs. Thus at most $|T|$ path variables are added in each pricing iteration. The second approach follows [12] and potentially adds multiple path variables for a single terminal in each iteration: We consider all nodes $v \in V$ adjacent to terminal t and all delay values $b = 1, \ldots, B - d_{vt}$ for which a path from s to v in conjunction with arc (v,t) is a feasible path to t. In case a shortest path p to v of total delay b, $0 < b \leq B - d_{vt}$, exists and $p' = p \cup \{(v,t)\}$ yields negative reduced costs, the corresponding variable is added to the RMP.

4 Column Generation Stabilization

It is well known that basic column generation based approaches typically suffer from computational instabilities often leading to long running times. Vanderbeck [31] describes five major causes of these instabilities including primal degeneracy, the tailing-off, and the heading-in effect. In order to improve the efficiency of such methods, several so-called column generation stabilization techniques aiming to reduce the effects of these problems have been proposed. These can be classified into problem specific techniques, such as the usage of dual-optimal inequalities [3, 6] and problem independent approaches, see e.g. [2, 25, 32, 23]. The latter are often based on the concept of stability centers which are current estimates of good dual variable values. The boxstep method [22] restricts each dual variable value to a small trust region around its current stability center. Other methods penalize deviations from the current stability center, e.g. by using piecewise linear penalty functions [3, 8]. Except for the weighted Dantzig-Wolfe decomposition approach, these so far proposed stabilization methods, however, usually need to add additional constraints and

variables to the RMP. Recently, we proposed a stabilization technique [18, 17, 19] which does not modify the RMP, but is based on using alternative dual-optimal solutions within the pricing subproblem. This method turned out to significantly accelerate the column generation process for a survivable network design problem by reducing the necessary number of iterations as well as the total number of included variables.

In the following we will show how this technique can be applied to the RDCSTP before we discuss two alternative stabilization approaches based on piecewise linear penalty functions. The latter two will then be used to compare our method.

4.1 Alternative Dual-Optimal Solutions

In order to describe our stabilization approach, we briefly discuss the dual of the RMP (8)–(13); primal variables are given in parenthesis.

$$\max \sum_{t \in T} \mu_t + \sum_{j \in V} \gamma_j \tag{8}$$

$$\text{s.t.} \quad \sum_{t \in T} \pi_{ij}^t + \gamma_j \leq c_{ij} \qquad (x_{ij}) \quad \forall (i,j) \in A \tag{9}$$

$$\mu_t - \sum_{(i,j) \in p} \pi_{ij}^t \leq 0 \qquad (\lambda_p) \quad \forall t \in T,\ \forall p \in \tilde{P}_t \tag{10}$$

$$\mu_t \geq 0 \qquad \forall t \in T \tag{11}$$

$$\pi_{ij}^t \geq 0 \qquad \forall t \in T,\ \forall (i,j) \in A \tag{12}$$

$$\gamma_j \leq 0 \qquad \forall j \in V \tag{13}$$

Inequalities (9) are capacity constraints imposing upper bounds on the sum of dual values $\sum_{t \in T} \pi_{ij}^t$ for each arc $(i,j) \in A$, while inequalities (10) ensure that the sum of dual arc costs π_{ij}^t along each included path is at least μ_t.

Let (μ^*, π^*, γ^*) denote the current dual solution computed by an LP solver for the RMP and $A' = \{(i,j) \in A \mid \nexists p \in \tilde{P} : (i,j) \in p\}$ be the set of arcs which are not used in any of the so far included path variables. Since only the capacity constraints (9) are relevant for these arcs, any dual variable values ${\pi_{ij}^t}^* \geq 0$ are optimal as long as $\sum_{t \in T} {\pi_{ij}^t}^* \leq c_{ij} - {\gamma_j}^*$, $\forall (i,j) \in A'$, holds. In case the capacity constraints are not binding, it is further possible to increase dual variable values ${\pi_{ij}^t}^*$, $t \in T$, for arcs $(i,j) \in A \setminus A'$ while maintaining dual optimality.

Let $\delta_{ij} = c_{ij} - \gamma_j - \sum_{t \in T} {\pi_{ij}^t}^*$, $\forall (i,j) \in A$, denote the slack of each capacity constraint (9). Then, obviously any values $\pi_{ij}^t \geq {\pi_{ij}^t}^*$, $\forall (i,j) \in A$, $\forall t \in T$, are dual-optimal as long as $\sum_{t \in T} \pi_{ij}^t \leq \sum_{t \in T} {\pi_{ij}^t}^* + \delta_{ij}$, $\forall (i,j) \in A$, holds. Note that state-of-the-art LP solvers usually yield minimal optimal dual variable values, i.e. ${\pi_{ij}^t}^* = 0$, $\forall t \in T$, $\forall (i,j) \in A'$. Based on these observations our stabilization approach aims to choose alternative dual-optimal solutions by distributing the slack δ_{ij} to the rele-

vant dual variables ${\pi_{ij}^t}^*$, $\forall t \in T$. We expect that increasing the dual variable values resulting in higher arc costs in the pricing subproblem facilitates the generation of meaningful path variables. One main advantage of choosing such alternative dual-optimal solutions for solving the pricing subproblem is that on the contrary to most other stabilization approaches we do not modify the RMP or increase its size by adding further variables or constraints.

Our first strategy is based on simply distributing the potential increase δ_{ij} equally among all relevant dual variables, i.e. we use alternative dual variables $\bar{\pi}_{ij}^t = {\pi_{ij}^t}^* + \frac{\delta_{ij}}{|T|}$, $\forall t \in T$, $\forall (i,j) \in A$. In our previous work for a survivable network design problem [18, 17, 19], however, it turned out to be beneficial to initially use different dual-optimal solutions, one for each terminal t, and let them finally converge towards $\bar{\pi}_{ij}^t$, $\forall t \in T$, $\forall (i,j) \in A$. Given an exogenous parameter $Q \geq 2$, denoting a total number of major iterations, the approach is iterated with parameter $q = 1, \ldots, Q$, indicating the current major iteration. Thus, parameter q is initially set to one (first major iteration) and gradually incremented by one in case no negative reduced cost path has been found. Let $t' \in T$ be the terminal currently considered in the pricing subproblem. Then the resulting dual variable values, which are denoted by $\hat{\pi}_{ij}^t$, $\forall t \in T$, $\forall (i,j) \in A$, are defined as follows:

$$\hat{\pi}_{ij}^t = \begin{cases} {\pi_{ij}^t}^* + \frac{\delta_{ij}}{|T|} + \frac{Q-q}{Q-1}\left(\delta_{ij} - \frac{\delta_{ij}}{|T|}\right) & \text{if } t = t' \\ {\pi_{ij}^t}^* & \text{otherwise} \end{cases}, \ \forall t \in T, \ \forall (i,j) \in A \tag{14}$$

This 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 value of q, $1 \leq q \leq Q$. Note that for each terminal $t \in T$ the resulting vector $\hat{\pi}$ is a dual-optimal solution. After Q major iterations, i.e. if $q = Q$, $\hat{\pi}_{ij}^t = \bar{\pi}_{ij}^t$ holds for each terminal t, i.e. we essentially use the same dual solution for all terminals. Thus, we can terminate the column generation process of the current node if $q = Q$ and no path variables have been added. Since most path variables are usually already generated in the root node of the branch-and-bound tree, we do not reinitialize parameter q. Hence, dual variable values $\bar{\pi}_{ij}^t$, $\forall t \in T$, $\forall (i,j) \in A$, are used in all further nodes of the branch-and-bound tree.

4.2 Piecewise Linear Stabilization

As mentioned before, successful stabilization techniques are often based on penalizing deviations from a current stability center by adding a stabilization term to the primal problem, i.e. a penalty function to the dual problem. Amor et al. [4] compared the performance of a Bundle-type approach and k-piecewise linear penalty functions using three and five pieces, respectively. They concluded that using five-piecewise linear penalty functions as originally proposed in [2] yields good results if all parameters are chosen carefully. We also adopted this approach in order to compare its performance to the previously described stabilization technique based

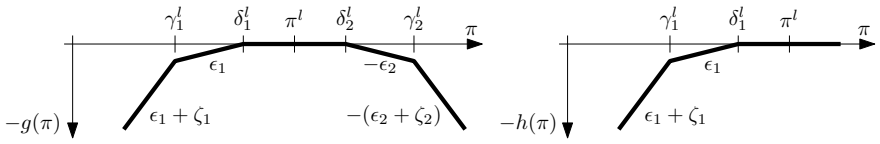

Fig. 1 5-piecewise and 3-piecewise linear dual penalty functions $g(\pi)$ and $h(\pi)$.

on alternative dual-optimal solutions. Given the current stability center $\pi^l \in \mathbb{R}_+^{|T|\cdot|A|}$ of major iteration $l \in \mathbb{N}$, $l \geq 1$, and a correspondingly defined penalty function $g(\pi)$ – see Figure 1 – dual variable values outside the trust region $\left[\delta_1^l, \delta_2^l\right]$ are penalized according to ζ_1, ε_1, ε_2, and ζ_2, respectively.

Let ${\pi_{ij}^t}^*$, $\forall t \in T$, $\forall (i,j) \in A$, denote the dual variable values when the column generation approach on the penalized model at major iteration l terminates. If there exists at least one dual variable value in the penalized region, we need to update the stability center according to the current dual solution, i.e. $\pi^{l+1} = \pi^*$, correspondingly set γ_1^{l+1}, δ_1^{l+1}, δ_2^{l+1}, and γ_2^{l+1}, and continue the column generation process. As has been shown previously [2] this process, which needs to be repeated until each dual variable value lies within an unpenalized region, terminates after finitely many steps yielding the LP relaxation of the current branch-and-bound node.

In our case, however, preliminary tests with various settings showed that due to a typically large number of relatively time consuming updates of the stability center this concept does not seem to pay off. Since the analysis in Section 4.1 shows that high dual variable values facilitate the generation of reasonable path variables, we further apply a second variant of this concept where only dual variable values smaller than δ_1^l are penalized using the penalty function $h(\pi)$ shown in Figure 1.

5 Computational Results

All described variants of the branch-and-price approach – denoted by BP throughout all tables – have been implemented in C++ using ZIB SCIP 2.0.1 [1] with IBM CPLEX 12.2 as embedded LP solver. We further decided to additionally test corresponding pure column generation – denoted by CG in the following – implemented by solely using CPLEX in order to analyze an eventually existing overhead of SCIP compared to CPLEX. Each computational experiment has been performed on a single core of an Intel Xeon E5540 processor with 2.53 GHz and 3 GB RAM per core. An absolute time limit of 10000 CPU-seconds has been applied to all experiments.

First, we tested our approaches on instances originally proposed by Gouveia et al. [12] for the spanning tree variant of the RDCSTP, i.e. $T = V \setminus \{s\}$. The three main instance sets R, C and E each have different graph structures defined by their edge cost functions: R has random edge costs, C and E both have Euclidean costs fixing the source s near the center and near the border, respectively. Each main instance set consists of different subsets of five complete input graphs with 41 nodes

varying in the number of possible discrete edge delay values; e.g. C100 denotes the set of instances with 100 different integer delay values: $d_e \in \{1,\dots,100\}, \forall e \in E$. Additionally we ran tests on instance sets Tα consisting of 30 randomly generated complete graphs with $|V| = 100$ where α denotes the number of terminal nodes $|T|$. Here all delays and costs are uniformly distributed in $\{1,\dots,99\}$. For each instance set we tested our approaches on different delay bounds B. Since we do consider these larger randomly generated instances Tα and since all instances with random costs from [12] could be solved relatively fast, we do not report here our detailed results for the latter due to space constraints. All preprocessing methods described in [27] are used to reduce the input graphs prior to solving. To build an initial set of paths a simple construction heuristic is applied on Steiner tree instances: the delay constrained shortest paths from the root to all terminal nodes are iteratively added to the tree dissolving possible cycles. On instances where $T = V \setminus \{s\}$ we apply the Kruskal-based heuristic followed by VND as introduced in [26].

Table 1 details median CPU-times in seconds for determining the LP relaxation of the IMP by unstabilized column generation, denoted by π^*, and when using stabilization based on alternative dual-optimal solutions $\bar{\pi}$ and $\hat{\pi}$, respectively. Here, OPT denotes the first described pricing strategy where at most one path per terminal is added in each iteration and MPT the one potentially adding multiple paths for a single terminal, compare Section 3.1. We further report average numbers of nodes $\overline{|V|}$ and edges $\overline{|E|}$ for each instance set after preprocessing. Finally, average CPU-times in seconds for the conceptually identical column generation approach by Gouveia et al. [12] – denoted as CG_G – as well as the Lagrangian approach Lag_G from the same authors are given in Table 1. The results of the latter two have, however, been computed on a different hardware and are thus not directly comparable. We observe that MPT outperforms OPT in almost all cases. Hence, we do not consider OPT in all further experiments. We further conclude that all stabilization strategies based on alternative dual-optimal solutions significantly outperform standard column generation. Note that already our unstabilized column generation variant needs significantly less iterations than the conceptually identical one discussed by Gouveia et al. [12]. We believe that next to different CPLEX versions these differences mainly come from choosing a better set of initial path variables, more sophisticated graph preprocessing, and the fact that we use the dual simplex algorithm which turned out to perform better than the primal one in our case.

Table 2 shows more detailed results for the variants of column generation using alternative dual-optimal solutions. Next to median CPU times in seconds (t_total), numbers of pricing iterations (Iterations) and total number of included path variables (Variables) we also report median CPU times for finding the correct LP value (t_best), i.e. the remaining time is needed for proving this value. We do not report on our experiments for Q=2 in Table 2 since we already observed from Table 1 that Q=2 is not competitive compared to Q=5 and Q=10, respectively. From these results we conclude that using $\hat{\pi}$ clearly outperforms $\bar{\pi}$ regarding the total CPU-time as well as the time for finding the correct LP value. Especially for harder instances – i.e. those requiring more time – Q=10 performs significantly better than the other configurations. Using $\bar{\pi}$ usually leads to a smaller number of total pricing iterations

Table 1 Median CPU-times in seconds for CG with different pricing strategies and stabilization techniques based on alternative dual-optimal solutions.

				OPT					MPT					CG_G	Lag_G
dual solution				π^*	$\bar{\pi}$	$\hat{\pi}$			π^*	$\bar{\pi}$	$\hat{\pi}$			-	-
Set	B	$\lvert\overline{V}\rvert$	$\lvert\overline{E}\rvert$	-	-	Q=2	Q=5	Q=10	-	-	Q=2	Q=5	Q=10	-	-
C2	3	41	279	1	1	1	2	2	**0**	**0**	1	1	1	2	4
	5	41	321	7	4	5	5	6	3	**2**	**2**	3	4	173	46
	7	41	321	25	7	8	6	8	11	**3**	4	4	5	3658	76
	9	41	321	62	10	10	8	9	33	7	6	**5**	6	8367	64
E2	3	41	597	5	5	5	7	8	**2**	**2**	**2**	4	5	13	12
	5	41	680	229	72	93	47	54	166	55	43	**22**	31	10045	208
	7	41	680	10000	983	989	246	243	10000	1600	871	113	**102**	10149	205
	9	41	680	10000	1326	1434	229	131	10000	3119	2142	657	**110**	10162	243
C100	100	41	561	173	36	35	31	31	41	**9**	10	**9**	12	10026	809
	150	41	572	808	61	71	48	46	118	24	**14**	16	16	10034	544
	200	41	572	3245	105	97	62	60	567	32	30	22	**21**	10061	711
	250	41	572	8742	103	113	64	63	3137	40	24	**17**	21	10076	1066
E100	100	41	651	520	82	93	56	54	201	62	26	**17**	19	1033	976
	150	41	672	3814	286	278	170	131	2911	376	227	126	**67**	10106	1817
	200	41	672	10000	1869	1501	325	192	10000	4098	1626	238	**158**	10096	2972
	250	41	672	10000	1589	1851	439	201	10000	10000	3453	734	**159**	10104	4008
C1000	1000	41	572	138	47	38	37	31	17	**7**	9	12	15	8186	668
	1500	41	589	648	74	63	64	60	115	**22**	29	**22**	28	10024	942
	2000	41	589	1730	136	144	131	90	599	80	47	38	**36**	10037	2389
	2500	41	589	6952	141	145	96	91	1336	56	47	**45**	54	10037	1256
E1000	1000	41	632	387	82	74	66	58	183	58	41	28	**27**	10065	2846
	1500	41	668	2830	268	343	268	148	2413	348	154	99	**69**	10031	3041
	2000	41	668	10000	671	1038	265	**177**	10000	1516	765	232	180	10083	5882
	2500	41	668	10000	863	1035	420	316	10000	4121	1365	315	**278**	10070	5726
T10	16	96	469	1	1	1	1	1	**0**	**0**	**0**	**0**	**0**	-	-
	30	100	932	23	6	6	6	6	3	**1**	**1**	**1**	2	-	-
	50	100	1269	188	17	18	18	16	15	**1**	**1**	2	2	-	-
	100	100	1695	2173	38	40	37	31	125	**2**	**2**	3	4	-	-
T30	16	98	482	3	3	3	3	4	**1**	**1**	**1**	2	2	-	-
	30	100	932	68	26	25	24	24	17	**4**	**4**	6	8	-	-
	50	100	1269	663	97	92	88	77	118	13	**12**	14	15	-	-
	100	100	1695	9973	345	370	311	269	1906	32	38	**27**	28	-	-
T50	16	99	486	7	6	5	6	7	3	**2**	**2**	4	5	-	-
	30	100	932	114	45	43	36	40	38	**11**	**11**	13	18	-	-
	50	100	1269	1128	159	167	135	140	249	30	**28**	31	38	-	-
	100	100	1695	10000	693	766	725	511	7739	192	106	112	**92**	-	-
T70	16	99	487	11	8	8	9	12	4	**3**	4	6	8	-	-
	30	100	932	177	60	62	59	56	51	**19**	20	24	28	-	-
	50	100	1269	1706	247	250	193	171	438	**67**	68	70	**67**	-	-
	100	100	1695	10000	1179	1036	1041	822	9017	402	436	**240**	249	-	-
T99	16	100	490	17	11	13	15	19	7	**4**	7	10	12	-	-
	30	100	932	320	108	98	83	93	147	49	**42**	45	50	-	-
	50	100	1269	2952	409	343	292	263	858	159	134	**111**	124	-	-
	100	100	1695	10000	1584	1759	1574	1275	10000	1166	1120	835	**629**	-	-

than all variants of $\hat{\pi}$ while the latter reduce the total number of included variables. Hence, we observe that $\hat{\pi}$ allows for finding more meaningful path variables already in the beginning of the column generation process. Since the best performing variants – i.e. using $\hat{\pi}$ and $Q \in \{5, 10\}$ – exhibit a quite significant tailing-off effect there is potential for further possible improvement e.g. by computing additional (Lagrangian) bounds or performing early branching in branch-and-price.

Table 2 Median CPU-times, median times for reaching the LP value, average pricing iterations, and included path variables for CG and stabilization based on alternative dual-optimal solutions.

		t_{total} [s]			t_{best} [s]			Iterations			Variables		
dual solution		$\bar{\pi}$	$\hat{\pi}$		$\bar{\pi}$	$\hat{\pi}$		$\bar{\pi}$	$\hat{\pi}$		$\bar{\pi}$	$\hat{\pi}$	
Set	B	-	Q=5	Q=10	-	Q=5	Q=10	-	Q=5	Q=10	-	Q=5	Q=10
C2	3	**0**	1	1	**0**	**0**	**0**	**10**	32	48	697	648	**641**
	5	**2**	3	4	**1**	**1**	2	**29**	63	89	2645	2024	**1901**
	7	**3**	4	5	**2**	3	**2**	**58**	82	122	4812	3312	**3014**
	9	7	**5**	6	2	**1**	**1**	**104**	106	130	7252	4700	**3808**
E2	3	**2**	4	5	**1**	2	2	**18**	46	74	1814	1639	**1596**
	5	55	**22**	31	54	**12**	16	**122**	154	203	11413	7740	**7186**
	7	1600	113	**102**	1597	100	**86**	383	**342**	365	36396	17908	**13459**
	9	3119	657	**110**	2746	45	**43**	570	565	**495**	59749	33377	**18123**
C100	100	**9**	**9**	12	5	**2**	**2**	**33**	68	101	16788	12828	**10783**
	150	24	**16**	**16**	16	**3**	**3**	**48**	79	113	33938	28465	**23247**
	200	32	22	**21**	15	**2**	**2**	**65**	104	131	52273	39777	**33140**
	250	40	**17**	21	17	**2**	**2**	**63**	89	132	56258	43697	**40100**
E100	100	62	**17**	19	34	**8**	9	**43**	90	127	27737	15702	**12177**
	150	376	126	**67**	184	25	**9**	**59**	114	147	75041	40916	**28441**
	200	4098	238	**158**	4069	**71**	82	**89**	149	194	142512	58547	**45980**
	250	10000	734	**159**	7804	76	**50**	**98**	194	201	209531	99451	**61488**
C1000	1000	**7**	12	15	6	**2**	3	**24**	60	97	30898	26540	**23374**
	1500	**22**	**22**	28	11	**4**	6	**33**	72	110	69064	59589	**53193**
	2000	80	38	**36**	52	10	**9**	**44**	81	118	103734	86713	**75027**
	2500	56	**45**	54	45	**6**	**6**	**42**	89	124	127440	112045	**100876**
E1000	1000	58	28	**27**	50	**6**	**6**	**31**	83	112	43492	29744	**24004**
	1500	348	99	**69**	303	22	**19**	**54**	119	151	116867	78501	**59886**
	2000	1516	232	**180**	740	**13**	14	**83**	141	162	239176	148387	**113407**
	2500	4121	315	**278**	2128	**18**	19	**92**	136	190	334410	187341	**156620**
T10	16	**0**	**0**	**0**	**0**	**0**	**0**	**11**	28	43	348	306	**290**
	30	**1**	**1**	2	**0**	**0**	**0**	**19**	37	59	1400	1221	**1083**
	50	**1**	2	2	**0**	**0**	**0**	**20**	42	63	2427	2153	**1894**
	100	**2**	3	4	**0**	1	1	**20**	42	66	3456	3249	**2779**
T30	16	**1**	2	2	**0**	**0**	1	**14**	39	61	982	852	**787**
	30	**4**	6	8	2	**1**	**1**	**29**	61	88	4946	4300	**3876**
	50	**13**	14	15	5	**2**	3	**43**	73	108	11489	10370	**9303**
	100	32	**27**	28	13	**4**	5	**44**	81	120	23889	23430	**20989**
T50	16	**2**	4	5	**1**	2	2	**15**	44	69	1458	1239	**1144**
	30	**11**	13	18	6	**5**	**5**	**32**	67	101	7271	6404	**5566**
	50	**30**	31	38	15	**5**	6	**50**	86	122	18202	16611	**15131**
	100	192	112	**92**	87	**11**	13	**63**	107	148	46145	43165	**40350**
T70	16	**3**	6	8	**2**	3	3	**15**	50	76	1864	1611	**1492**
	30	**19**	24	28	14	**7**	10	**30**	74	109	9269	7999	**7012**
	50	**67**	70	**67**	32	**11**	12	**50**	97	140	23640	22124	**19302**
	100	402	**240**	249	263	34	**33**	**74**	127	173	65179	63348	**57621**
T99	16	**4**	10	12	**3**	4	**3**	**16**	52	76	2397	2049	**1887**
	30	49	**45**	50	33	**15**	19	**33**	82	115	12063	9991	**8717**
	50	159	**111**	124	91	**20**	28	**54**	110	153	33106	27500	**24996**
	100	1166	835	**629**	750	73	**67**	**88**	160	202	99801	88283	**78012**

Next, we analyze and compare the performance of the two column generation variants using piecewise linear stabilization terms. Table 3 reports median CPU-times in seconds and performed updates of the stability centers for different instance sets from both types. 3PL denotes the approach penalizing only small values and 5PL the full approach using a 5-piecewise linear penalty function. For both variants the initial stability center is chosen according to the first strategy for using alterna-

Table 3 Median total CPU-times and updates of the stability center for CG and piecewise linear stabilization techniques.

			3PL								5PL							
		t_{total} $[s]$	t_{total} $[s]$				$\overline{\text{Updates}}$				$\overline{t_{\text{total}}}$ $[s]$				$\overline{\text{Updates}}$			
		π^*	sml		lrg		sml		lrg		sml		lrg		sml		lrg	
Set	B	-	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2	S_1	S_2
C100	100	56	204	255	138	173	18	23	9	11	10000	10000	10000	10000	51	78	48	47
	150	133	674	493	575	470	15	20	8	11	10000	10000	10000	10000	39	62	22	33
	200	1056	2482	2554	3008	2728	18	20	10	11	10000	10000	10000	10000	24	40	14	22
	250	3485	10000	9474	10000	7350	16	20	10	12	10000	10000	10000	10000	15	24	9	14
E100	100	463	692	627	522	639	19	24	10	12	10000	10000	10000	10000	35	62	18	29
	150	4018	10000	10000	10000	10000	18	19	9	10	10000	10000	10000	10000	21	35	10	17
	200	9137	10000	10000	10000	10000	7	7	4	4	10000	10000	10000	10000	14	25	6	11
	250	9861	10000	10000	10000	10000	5	6	3	3	10000	10000	10000	10000	9	15	4	7
T10	30	4	6	6	5	5	3	3	2	2	52	79	21	34	26	50	11	20
	50	17	24	28	21	22	3	3	2	2	242	322	102	137	29	56	12	22
T30	30	20	43	51	39	40	12	15	7	8	2281	2797	1250	1525	105	181	50	85
	50	137	353	355	301	325	12	17	6	9	10000	10000	8409	9961	89	113	64	87
T50	30	49	132	135	98	117	22	29	11	16	10000	10000	6586	7294	87	126	84	136
	50	364	986	957	695	856	24	33	12	17	10000	10000	10000	10000	28	47	18	29

tive dual-optimal solutions, i.e. $\pi^1 = \bar{\pi}$, and we tested various settings of the inner trust region radius $\pi^{t\ l}_{ij} - \delta^{t\ l}_{ij1} = \delta^{t\ l}_{ij2} - \pi^{t\ l}_{ij} = r_{\text{i}} \frac{c_{ij}}{|T|}$ and the outer trust region radius $\pi^{t\ l}_{ij} - \gamma^{t\ l}_{ij1} = \gamma^{t\ l}_{ij2} - \pi^{t\ l}_{ij} = r_{\text{o}} \frac{c_{ij}}{|T|}$, $\forall t \in T$, $\forall (i,j) \in A$, respectively. For 5PL we used symmetric penalty functions, i.e. $\varepsilon = \varepsilon_1 = \varepsilon_2$ and $\zeta = \zeta_1 = \zeta_2$, and identical penalty slopes for all dimensions. Among the various tested configurations we report here experiments with smaller (sml) and larger (lrg) trust region radii $r_{\text{i}} = 1$, $r_{\text{o}} = 3$ and $r_{\text{i}} = 2$, $r_{\text{o}} = 6$, respectively. Furthermore, we tested penalty slopes S_1 where $\varepsilon = 0.3$ and $\zeta = 1$, and S_2 where $\varepsilon = 0.5$ and $\zeta = 1.5$.

In Table 3 we observe that 3PL and 5PL are clearly not competitive to all approaches based on alternative dual-optimal solutions. Although the number of updates of the stability center was not too high, the additional overhead due to these updates and the additional variables and constraints in the model lead to long running times which are usually even higher than those of unstabilized column generation. As we could not identify better parameter values in further tests we conclude that using piecewise linear stabilization does not seem to be promising for the RDCSTP.

Table 4 compares number of instances solved to proven optimality, average remaining gaps, and median CPU-times in seconds for the full branch-and-price approach using alternative dual-optimal solutions $\hat{\pi}$ and ten major iterations (Q=10) to the theoretically stronger static (SL) and dynamic (DL) layered graph approach from [28] (rerun with CPLEX 12.2). SL starts with the same primal solution as BP whereas DL does not use any initial heuristics here. Directed connection cuts are separated in SL and DL only for the instance sets from [12]; Tα instances are solved faster when omitting them. We additionally report the average number of considered branch-and-bound nodes, the average integrality gap at the root node and the median time needed for solving the root node of the branch-and-bound tree for BP as well as the corresponding CPU-time of the column generation approach.

Table 4 Number of instances solved to proven optimality, average optimality gap, root node gap, number of branching nodes, median total CPU-time, and median time for solving the root node for BP compared to layered graph approaches SL and DL.

		Σ Opt			$\overline{\text{gap}}$ [%]			$\overline{t_{\text{total}}}$ [s]			$\overline{\text{gap}_{\text{root}}}$ [%]	$\overline{\text{Nodes}}$	$\overline{t_{\text{root}}}$ [s]	
Set	B	SL	DL	BP	SL	DL	BP	SL	DL	BP	BP	BP	BP	CG
C2	3	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0**	**0**	1	0.0	1.0	1	1
	5	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0**	3	5	0.7	5.0	3	4
	7	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**1**	5	9	1.0	13.0	5	5
	9	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**3**	12	8	0.6	5.0	5	6
E2	3	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**0**	1	4	0.0	1.0	4	5
	5	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**3**	31	34	2.5	5.2	25	31
	7	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	**18**	522	743	2.3	14.2	122	102
	9	**5**	**5**	4	**0.0**	**0.0**	0.1	**48**	893	914	2.3	7.6	149	110
C100	100	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	130	33	**9**	0.1	4.2	9	12
	150	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	1383	208	**18**	0.0	1.0	18	16
	200	3	**5**	**5**	0.9	**0.0**	**0.0**	7552	1219	**21**	0.1	3.8	21	21
	250	1	**5**	**5**	10.7	**0.0**	**0.0**	10000	1415	**35**	0.1	2.2	20	21
E100	100	**5**	**5**	**5**	**0.0**	**0.0**	**0.0**	1212	766	**22**	0.9	2.4	20	19
	150	0	3	**5**	6.8	0.3	**0.0**	10000	8299	**122**	0.3	4.6	70	67
	200	0	0	**4**	13.1	2.6	**0.1**	10000	10000	**453**	0.8	9.0	361	158
	250	0	0	**4**	12.2	3.0	**0.1**	10000	10000	**993**	0.9	8.0	303	159
C1000	1000	4	**5**	**5**	1.8	**0.0**	**0.0**	2509	20	**13**	0.0	1.0	13	15
	1500	0	**5**	**5**	12.2	**0.0**	**0.0**	10000	175	**40**	0.2	2.6	33	28
	2000	0	4	**5**	100.0	**0.0**	**0.0**	10000	5203	**75**	0.6	19.8	61	36
	2500	0	**5**	**5**	100.0	**0.0**	**0.0**	10000	2163	**53**	0.0	3.0	53	54
E1000	1000	1	**5**	**5**	11.3	**0.0**	**0.0**	10000	1296	**24**	0.1	1.4	24	27
	1500	0	3	**5**	34.8	0.7	**0.0**	10000	3127	**63**	0.3	4.2	63	69
	2000	0	1	**5**	100.0	1.3	**0.0**	10000	10000	**223**	0.5	2.2	223	180
	2500	0	1	**4**	100.0	2.2	**0.1**	10000	10000	**1588**	0.6	9.4	642	278
T10	16	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0**	**0**	0	0.1	1.3	0	0
	30	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	3	4	**1**	0.1	1.5	1	2
	50	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	16	16	**2**	0.2	1.2	2	2
	100	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	106	44	**3**	0.0	1.1	3	4
T30	16	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0**	1	2	0.2	1.5	2	2
	30	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**4**	8	8	0.0	1.5	8	8
	50	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	25	52	**21**	0.5	2.7	17	15
	100	**30**	29	**30**	**0.0**	0.2	**0.0**	186	103	**43**	0.9	4.2	39	28
T50	16	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0**	1	4	0.1	1.6	5	5
	30	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**5**	15	17	0.5	6.3	15	18
	50	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**34**	67	52	0.8	6.1	39	38
	100	**29**	27	28	0.2	0.2	**0.1**	319	505	**217**	0.9	6.6	151	92
T70	16	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**0**	2	8	0.1	1.8	8	8
	30	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**5**	24	33	0.7	8.1	27	28
	50	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**34**	84	76	0.5	3.5	66	67
	100	**28**	26	27	**0.1**	0.3	0.6	**388**	519	426	1.1	6.9	315	249
T99	16	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**1**	2	11	0.5	4.0	11	12
	30	**30**	**30**	**30**	**0.0**	**0.0**	**0.0**	**7**	26	54	0.9	14.6	42	50
	50	**30**	**30**	29	**0.0**	**0.0**	0.1	**36**	76	167	0.5	7.4	117	124
	100	**28**	**28**	26	0.3	**0.2**	0.3	**298**	675	867	0.8	4.5	516	629

We conclude that due to the stabilization based on alternative dual-optimal solutions the proposed branch-and-price approach outperforms both layered graph approaches in many cases and performs particularly good when the delay bound is not too strict as well as when the relative number of terminal nodes is not too high. Thus we consider it a good complement to the layered graph approaches from [28].

6 Conclusions and Future Work

In this paper we presented a branch-and-price approach for the RDCSTP. Column generation stabilization methods based on alternative dual-optimal solutions and piecewise linear penalty functions have been applied to accelerate the approach. We further compared the performance of two different pricing strategies. We conclude that when using stabilization based on alternative dual-optimal solutions our method outperforms so-far proposed exact methods for the RDCSTP in many cases and allows for computing proven optimal solutions solutions to medium sized instances within reasonable time. In future, we want to compare our approach to further stabilization techniques such as e.g. interior point stabilization [25] or weighted Dantzig-Wolfe decomposition [32, 23] as well as combine promising aspects of different stabilization techniques potentially yielding an additional speed-up. We further want to study the impact of different possibilities for choosing an initial set of columns as well as aim at reducing the tailing-off effect.

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. J. M. V. de Carvalho. Using extra dual cuts to accelerate convergence in column generation. *INFORMS Journal on Computing*, 17(2):175–182, 2005.
7. G. Desaulniers, J. Desrosiers, and M. M. Solomon, editors. *Column Generation*. Springer, 2005.
8. O. du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen. Stabilized column generation. *Discrete Mathematics*, 194(1–3):229–237, 1999.
9. I. Dumitrescu and N. Boland. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks*, 42(3):135–153, 2003.
10. M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.
11. N. Ghaboosi and A. T. Haghighat. A path relinking approach for delay-constrained least-cost multicast routing problem. In *19th IEEE International Conference on Tools with Artificial Intelligence*, pages 383–390, 2007.
12. 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.
13. 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*, pages 1–26, 2010.

14. 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.
15. L. Kou, G. Markowsky, and L. Berman. A fast algorithm for Steiner trees. *Acta Informatica*, 15(2):141–145, 1981.
16. 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.
17. M. Leitner and G. R. Raidl. Strong lower bounds for a survivable network design problem. In M. Haouari and A. R. Mahjoub, editors, *ISCO 2010*, volume 36 of *Electronic Notes in Discrete Mathematics*, pages 295–302. Elsevier, 2010.
18. 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.
19. 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.
20. M. Leitner, M. Ruthmair, and G. R. Raidl. Stabilized column generation for the rooted delay-constrained Steiner tree problem. In *VII ALIO/EURO – Workshop on Applied Combinatorial Optimization*, Porto, Portugal, May 2011.
21. P. Manyem and M. Stallmann. Some approximation results in multicasting. Technical Report TR-96-03, North Carolina State University, 1996.
22. R. E. Marsten, W. W. Hogan, and J. W. Blankenship. The BOXSTEP method for large-scale optimization. *Operations Research*, 23(3):389–405, 1975.
23. 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.
24. R. Qu, Y. Xu, and G. Kendall. A variable neighborhood descent search algorithm for delay-constrained least-cost multicast routing. In *Proceedings of Learning and Intelligent OptimizatioN (LION3)*, pages 15–29. Springer, 2009.
25. L.-M. Rousseau, M. Gendreau, and D. Feillet. Interior point stabilization for column generation. *Operations Research Letters*, 35(5):660–668, 2007.
26. 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, *EUROCAST 2009*, volume 5717 of *LNCS*, pages 713–720. Springer, 2009.
27. 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, *PPSN XI, Part II*, volume 6239 of *LNCS*, pages 391–400. Springer, 2010.
28. M. Ruthmair and G. R. Raidl. A layered graph model and an adaptive layers framework to solve delay-constrained minimum tree problems. In *Fifteenth Conference on Integer Programming and Combinatorial Optimization (IPCO XV)*, 2011. to appear.
29. N. Skorin-Kapov and M. Kos. The application of Steiner trees to delay constrained multicast routing: a tabu search approach. In *Proceedings of the 7th International Conference on Telecommunications*, volume 2, pages 443–448, 2003.
30. N. Skorin-Kapov and M. Kos. A GRASP heuristic for the delay-constrained multicast routing problem. *Telecommunication Systems*, 32(1):55–69, 2006.
31. F. Vanderbeck. Implementing mixed integer column generation. In Desaulniers et al. [7], pages 331–358.
32. P. Wentges. Weighted Dantzig-Wolfe decomposition for linear mixed-integer programming. *International Transactions in Operational Research*, 4(2):151–162, 1997.
33. Y. Xu and R. Qu. A GRASP approach for the delay-constrained multicast routing problem. In *Proceedings of the 4th Multidisplinary International Scheduling Conference (MISTA4)*, pages 93–104, 2009.
34. Y. Xu and R. Qu. A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems. *Applied Intelligence*, pages 1–13, 2010.