A Layered Graph Model and an Adaptive Layers Framework to Solve Delay-Constrained Minimum Tree Problems

Mario Ruthmair and Günther R. Raidl

Vienna University of Technology, Vienna, Austria
Institute of Computer Graphics and Algorithms
{ruthmair,raidl}@ads.tuwien.ac.at

Abstract. We present a layered graph model for delay-constrained minimum tree problems with a polynomial number of constraints which can be solved well for instances with low- to medium-sized sets of achievable delay values and not too high bounds. Layered graph models have been recently shown to frequently yield tight bounds in the context of hop- or delay-constrained network design problems. However, since the size of the layered graph heavily depends on the size of the set of achievable delay values and the corresponding delay bound the practical applicability of these models is limited. To overcome this problem we introduce an iterative strategy in which an initially small layered graph is successively extended in order to tighten lower and upper bounds until convergence to the optimal solution. Computational results show the synergetic effectiveness of both approaches outperforming existing models in nearly all cases.

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. In another example we 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. These network design problems can be modeled using an $\mathcal{NP}$-hard combinatorial optimization problem called *delay-constrained minimum tree (DCMT) problem* [8]. The objective is to find a minimum cost Steiner tree on a given graph with the additional constraint that the sum of delays 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, a fixed root node $s \in V$, set $R \subseteq V \setminus \{s\}$ of terminal or required nodes,

O. Günlük and G.J. Woeginger (Eds.): IPCO 2011, LNCS 6655, pp. 376–388, 2011.

set $S = V \setminus (R \cup \{s\})$ of optional Steiner nodes, edge set E, a cost function $c : E \to \mathbb{Z}^+$, a delay function $d : E \to \mathbb{Z}^+$, and a delay bound $B \in \mathbb{Z}^+$. An optimal solution to the DCMT problem is a Steiner tree $T = (V^T, E^T)$, $s \in V^T$, $R \subset V^T \subseteq V$, $E^T \subseteq E$, with minimum cost $c(T) = \sum_{e \in E^T} c_e$, satisfying the constraints $d_v = \sum_{e \in P(s,v)} d_e \leq B$, $\forall v \in R$, where $P(s, v)$ denotes the unique path from root s to node v.

There are many recent publications dedicated to the DCMT problem and its more special variants. Manyem et al. [12] showed that the problem is not in APX. Several metaheuristics have been presented, such as GRASP [17], variable neighborhood search [17,18], and path-relinking in a hybrid scatter search [18]. More heuristic approaches can be found for the variant with $R = V \setminus \{s\}$, e.g. a GRASP and a variable neighborhood descent in [15] and ant colony optimization and a variable neighborhood search in [16]. Furthermore, preprocessing methods are presented in [16] reducing the size of the input graph significantly.

Exact methods for the DCMT problem based on integer linear programming (ILP) have been explored by Leggieri et al. [9] who describe a compact extended node-based formulation using lifted Miller-Tucker-Zemlin inequalities. Since these Big-M formulations usually yield rather weak linear programming (LP) bounds they improve it by adding directed connection cuts. In [3] Gouveia et al. transform a DCMT problem variant called *Hop-Constrained Minimum Spanning Tree (HCMST) Problem* where $d_e = 1$, $\forall e \in E$, and $R = V \setminus \{s\}$, to an equivalent Steiner tree problem (STP) [2] on an appropriate layered graph without additional constraints. The intensively studied STP can then be solved by any existing approach for directed graphs. In [3] a classic directed connection cut formulation on this layered graph is solved by an efficient branch-and-cut algorithm. This formulation has been shown to be stronger than the HCMST formulation in [4] only modeling the constrained shortest path subproblems on a layered graph. ILP approaches for the DCMT problem with $R = V \setminus \{s\}$ have been examined by Gouveia et al. in [6] based on the concept of constrained shortest paths utilized in column generation and Lagrangian relaxation methods. Similarly to [4] a third approach reformulates the constrained shortest path subproblems on a layered graph and solves them using 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 delay values (see Section 2) this approach can in practice only be used for instances with a reasonably small set of delay values and rather low bounds. Additionally, MCF models usually suffer from the huge amount of flow variables used in the ILP formulation altogether leading to a slow and memory-intensive solving process. Nevertheless solving these layered graph models turned out to be very effective on certain classes of instances, not only for DCMT problems, but e.g. for the hop-constrained connected facility location problem as well, see [10].

The success of the layered graph transformation for some special variants of the DCMT problem leads us to a further investigation of this approach. First, we introduce an efficient ILP model utilizing the special structure of the layered graph and improving the computational performance compared to existing

models. However, the problems with huge sets of achievable delay values and high bounds still exist although much harder instances can now be tackled. To substantially improve this situation we present a new iterative strategy based on solving the problem on smaller layered graphs yielding lower and upper bounds to the optimal costs. By extending these simplified graphs appropriately the bounds are tightened to finally converge to an optimal solution. Compared to our first approach the iterative framework consumes substantially less memory. More generally, our strategy can in principle also be applied to other problems with delay or weight constraints that can be modeled on layered graphs.

The rest of the article is organized as follows: Section 2 describes the transformation to the layered digraph, Section 3 presents the ILP model on this graph and Section 4 shows some theoretical results utilized in the adaptive layers framework in Section 5. Section 6 discusses computational results and Section 7 concludes the article and sketches future work.

2 Transformation to the Steiner Arborescence Problem on Layered Digraphs

Similarly to [6,3] we transform the original graph $G = (V, E)$ to a layered digraph $G_L = (V_L, A_L)$. The node set $V_L = \{s\} \cup S_L \cup R_L$ includes Steiner nodes $S_L = \{i_l : i \in R \cup S,\ 1 \leq l \leq (B-1)\}$ and required nodes $R_L = \{i_B : i \in R\}$. The arc set $A_L = A_s \cup A_g \cup A_z$ consists of root arcs $A_s = \{(s, i_{d_{si}}) : \{s, i\} \in E\}$, general arcs $A_g = \{(i_l, j_{l+d_{ij}}), (j_l, i_{l+d_{ij}}) : \{i, j\} \in E,\ i, j \neq s,\ 1 \leq l \leq (B - d_{ij})\}$ and zero arcs $A_z = \{(i_l, i_B) : i \in R,\ 1 \leq l \leq (B-1)\}$. Arc delays d_{ij} are not needed in G_L since they are implicitly contained in the layered structure: node i_l in G_L represents node i in G with $d_i = l$. The arc costs in A_s and A_g equal the costs of corresponding edges in E, arcs A_z get assigned zero costs. Fig. 1(a) and 1(b) demonstrate the transformation. Usually, G_L can be reduced by the following preprocessing rule: if a Steiner node $v \in S_L$ has no incoming or no outgoing arcs it is removed together with all incident arcs. This preprocessing is able to reduce the number of Steiner nodes and arcs significantly especially for instances with a broad range of delay values, see Fig. 1(c) and Table 1. Further preprocessing methods for Steiner trees can be found in [7,11].

The *Steiner Arborescence Problem* on G_L is to find a Steiner tree $T_L = (V_L^T, A_L^T)$ rooted in $s \in V_L^T$ with $R_L \subset V_L^T \subseteq V_L$, $A_L^T \subseteq A_L$ and minimal arc costs $c_L^T = \sum_{a \in A_L^T} c_a$. An optimal Steiner arborescence $T_{L,\text{opt}}$ on G_L corresponds to an optimal Steiner tree T_{opt} on G, moreover $c(T_{L,\text{opt}}) = c(T_{\text{opt}})$. This has been shown in [3] for the HCMST problem and can be generalized to the DCMT problem in a natural way. We simply transform $T_{L,\text{opt}}$ to T_{opt} by removing all zero arcs $(i_l, i_B) \in A_L^T$ together with their target nodes and rename all nodes $i_l \in V_L^T$ to i. Fig. 1(c) and 1(d) show the optimal solution to the DCMT problem on the example graph in Fig. 1(a).

There are many existing approaches for efficiently solving the Steiner tree problem on graphs, e.g. [7,14,1]. All general ILP tree models either need additional variables or an exponential number of constraints. In our case we are

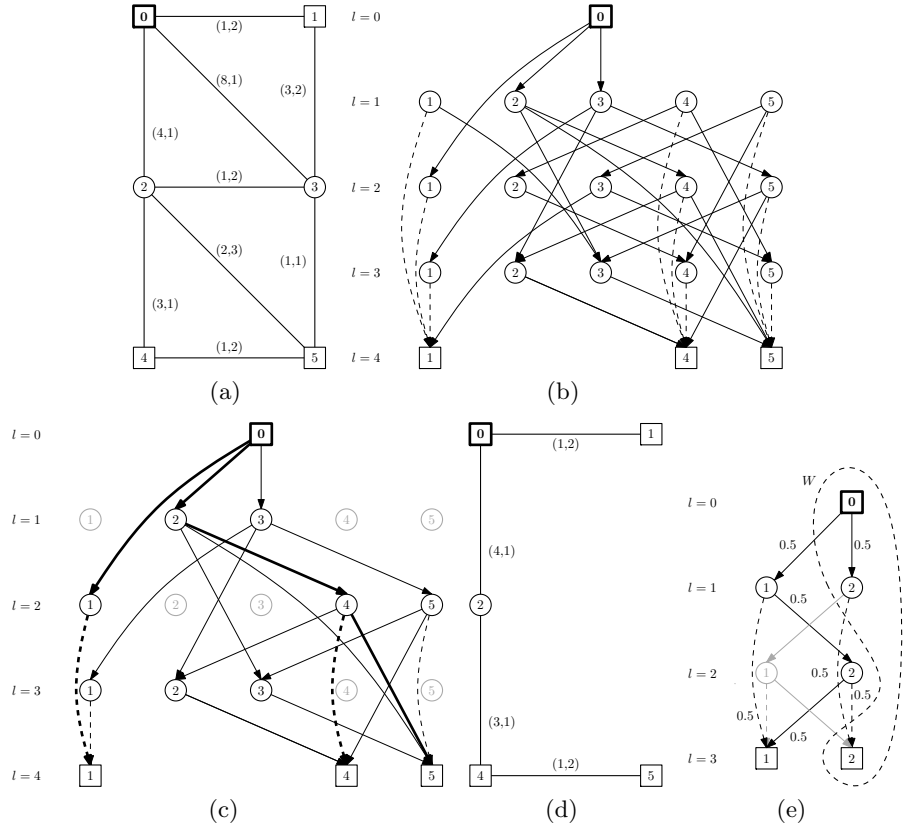

Fig. 1. Example graph in (a) with edge labels (*cost*, *delay*) and root node 0. Squared nodes denote terminal nodes. Corresponding layered digraph in (b) for $B = 4$ (arc costs are omitted). Preprocessed graph G_L in (c) with optimal solution denoted by bold arcs. Optimal tree T in G in (d) with $c(T) = 9$. Example of an LP solution in (e) where a directed connection cut inequality based on set W tightens the LP relaxation (arc labels denote the corresponding y-values, grayed out arcs mean $y = 0$).

lucky to work on a special graph structure: the layered digraph is acyclic. This property makes it possible to model the problem effectively with a polynomial number of constraints without additional variables, see [13,5] and Section 3. Zelikovsky et al. [19] present approximation algorithms for the Steiner tree problem in acyclic digraphs.

3 ILP Model on the Layered Digraph

We use binary variables x_e as design variables for edges $e \in E$ indicating whether the edge is included in the solution T ($x_e = 1$) or not ($x_e = 0$). Similarly, we

use non-negative variables $y_{i_l j_k}$ for arcs $(i_l, j_k) \in A_L$ in T_L. Adapting the hop-indexed model for the HCMST problem in [5] to the DCMT problem leads to the Steiner Arborescence Layered (SAL) model:

$$\min \quad \sum_{e \in E} c_e x_e \tag{1}$$

$$\text{s.t.} \quad \sum_{(i_l, j_B) \in A_L} y_{i_l j_B} = 1 \quad \forall j \in R \tag{2}$$

$$\sum_{(k_{l-d_{ki}}, i_l) \in A_L, k \neq j} y_{k_{l-d_{ki}} i_l} \geq y_{i_l j_{l+d_{ij}}} \quad \forall (i_l, j_{l+d_{ij}}) \in A_g \tag{3}$$

$$\sum_{(k_{l-d_{ki}}, i_l) \in A_L} y_{k_{l-d_{ki}} i_l} = y_{i_l i_B} \quad \forall (i_l, i_B) \in A_z \tag{4}$$

$$y_{s i_{d_{si}}} = x_e \quad \forall e = \{s, i\} \in E \tag{5}$$

$$\sum_{(i_l, j_{l+d_{ij}}) \in A_L} y_{i_l j_{l+d_{ij}}} + \sum_{(j_l, i_{l+d_{ij}}) \in A_L} y_{j_l i_{l+d_{ij}}} = x_e \quad \forall e = \{i, j\} \in E,\ i, j \neq s \tag{6}$$

$$y_{i_l j_k} \geq 0 \quad \forall (i_l, j_l) \in A_L \tag{7}$$

$$x_e \in \{0, 1\} \quad \forall e \in E \tag{8}$$

Constraints (2) ensure that each terminal node in G_L has exactly one incoming arc. The connectivity constraints (3) make sure that if there is an arc going out of a Steiner node there has to be an incoming arc, too. Constraints (4) force the use of the zero arc if there is an incoming arc. Together with equalities (2) this leads to the use of at most one Steiner node of $\{i_l : 1 \leq l \leq (B-1)\},\ \forall i \in R$. Equalities (5) and (6) link the directed arc variables y in G_L to the corresponding undirected edge variables x in G. By relaxing the integrality constraints (8) we obtain the corresponding linear program SAL_{LP}.

Theorem 1. *Model SAL can be used to solve the DCMT problem.*

Proof. Constraints (2) force exactly one incoming arc for each terminal node $j \in R_L$ in layer B of G_L. We have to show that all terminal nodes are connected to the root node s. If the incoming arc (i, j) originates in s we are done. Otherwise constraints (3) and (4) ensure an incoming arc to i. Due to the acyclicity and the layered structure of G_L the source of an arc can only be in a lower layer than the target. Repeating this argumentation for node i extends the path in a backtracking way to the root node in layer 0. The union of all such paths forms a connected acyclic graph including all terminal nodes. □

We optionally add directed connection cut inequalities

$$\sum_{(i_l, j_k) \in A_L,\ i_l \in W,\ j_k \notin W} y_{i_l j_k} \geq 1 \qquad \forall W \subset V_L,\ s \in W,\ (V_L \setminus W) \cap R_L \neq \emptyset \tag{9}$$

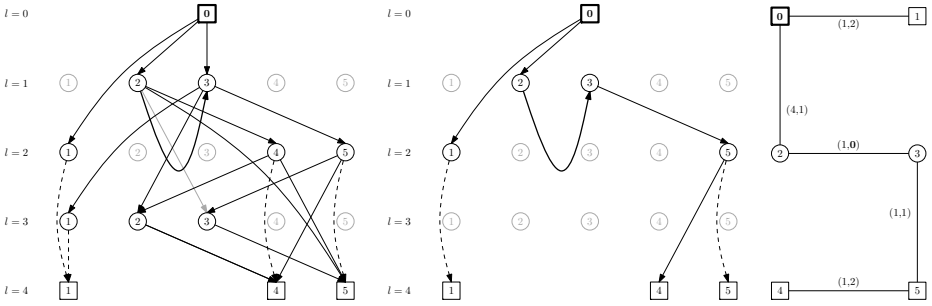

Fig. 2. A layered graph G'_L is shown on the left derived from G_L by redirecting arc $(2_1, 3_3)$ to node 3 on layer 1. In the middle the optimal solution $T'_{L,\text{opt}}$ in G'_L and on the right the reverse transformed infeasible solution T'_{opt} in G with $c(T'_{\text{opt}}) = 8$ is shown. The delay of edge $(2, 3)$ in G is decreased to 0.

to model *SAL* to further tighten the LP relaxation, see Fig. 1(e) for an example, and denote this extended model SAL^{dcut}. The optimal LP value of this model is at least as high as the one of the MCF model in [6] and there are cases in which the LP value is strictly better. This result has been shown by Gouveia et al. [3] for the HCMST problem, and the proof can be trivially adapted for our DCMT case.

The number of variables and constraints of model *SAL* can be estimated by Equations (10) and (11) showing the high dependency on the delay bound B. Therefore B is crucial for the performance and memory consumption of this model which can be clearly observed in the experimental results, see Section 6.

$$|\mathit{variables}| = |E| + |A_L| = \mathcal{O}(|E| \cdot B) \tag{10}$$

$$|\mathit{constraints}| = |R| + |A_g| + |A_z| + 2|E| + |A_L| = \mathcal{O}(|V| + |E| \cdot B) \tag{11}$$

To partly overcome this drawback we introduce the *Adaptive Layers Framework (ALF)* in Section 5 based on the theoretical results presented in the following. In principle *ALF* calculates lower and upper bounds to the optimal costs in reduced layered graphs and iteratively closes the gap by extending these graphs appropriately until the two bounds are equal.

4 Lower and Upper Bounds by Redirecting Arcs

We define the length of the shortest delay path to each node in the original graph G:

$$d_v^{\min} := \min_{P(s,v)} \sum_{e \in P(s,v)} d_e, \quad \forall v \in V \setminus \{s\}$$

In G_L we now consider an arc $(u_{l-d_{uv}}, v_l) \in A_s \cup A_g$, $u_{l-d_{uv}} \in V_L$, $v_l \in S_L$, $d_v^{\min} < l < B$ and redirect its target to a node $v_k \in S_L$ on a lower layer $k < l$. Since $l > d_v^{\min}$ there always exists a feasible node v_k with $k < l$. We denote the resulting graph G'_L.

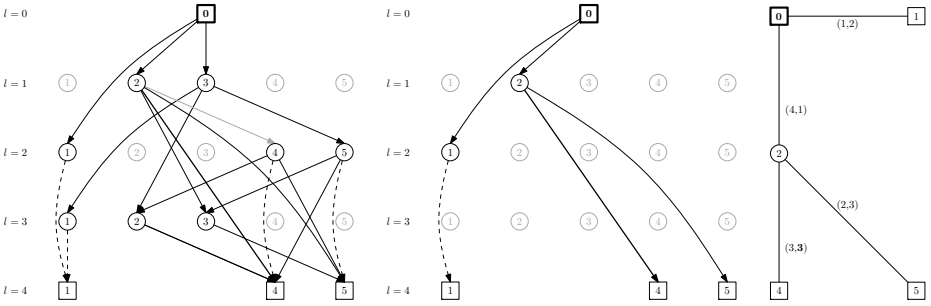

Fig. 3. A layered graph G''_L is shown on the left derived from G_L by redirecting arc $(2_1, 4_2)$ to node 4 on layer 4. In the middle the optimal solution $T''_{L,\text{opt}}$ in G''_L and on the right the reverse transformed solution T''_{opt} in G with $c(T''_{\text{opt}}) = 10$ is shown. The delay of edge $(2, 4)$ in G is increased to 3.

Lemma 1. *Let $T_{L,\text{opt}}$ and $T'_{L,\text{opt}}$ be optimal solutions to the Steiner arborescence problem on G_L and G'_L, respectively. Furthermore we denote by T_{opt} and T'_{opt} the corresponding reverse transformed trees in G, respectively. Then $c(T'_{\text{opt}}) \leq c(T_{\text{opt}})$.*

Proof. Redirecting an arc in G_L from target v_l to node v_k on a lower layer $k < l$ corresponds to a decrease of the related edge delay in G. Therefore, the solution space may be extended since it may now be easier to satisfy the delay bound. The optimal solution T_{opt} still stays feasible but one or more of the new solutions may have less cost than T_{opt}, so $c(T'_{\text{opt}}) \leq c(T_{\text{opt}})$. □

Fig. 2 shows layered graph G'_L derived from the previous example graph in Fig. 1(c). One arc is redirected to a new target on a lower layer. The optimal solution $T'_{L,\text{opt}}$ in G'_L has less cost than the optimal solution $T_{L,\text{opt}}$ in G_L. Therefore T'_{opt} cannot be feasible for the DCMT problem on G otherwise T_{opt} would not be optimal. On the other hand if T'_{opt} would be feasible Lemma 2 applies.

Lemma 2. *If T'_{opt} is feasible for the DCMT problem on G then it is optimal.*

Proof. According to Lemma 1, $c(T'_{\text{opt}}) \leq c(T_{\text{opt}})$. If T'_{opt} is feasible for the DCMT problem on G then $c(T'_{\text{opt}}) = c(T_{\text{opt}})$. Therefore T'_{opt} is an optimal solution to the original problem. □

If we redirect an arc to a target node on a higher layer instead of a lower one and denote the resulting graph G''_L, Lemma 3 holds.

Lemma 3. *Let T''_L be any feasible solution in G''_L and T'' the corresponding reverse transformed tree in G. Then T'' is feasible for the DCMT problem on G. Furthermore, $c(T''_{\text{opt}}) \geq c(T_{\text{opt}})$.*

Proof. Redirecting an arc in G_L to a new target node on a higher layer corresponds to increasing the respective edge delay in G. Therefore it may be harder

to satisfy the delay bound and the solution space may be pruned. Nevertheless all feasible solutions T'' stay feasible for the original DCMT problem on G since replacing the modified edge delay by their original smaller one cannot violate the delay bound. However, former optimal solutions may now be infeasible in G''_L, so $c(T''_{\text{opt}}) \geq c(T_{\text{opt}})$. □

Figure 3 shows an example to this case. By redirecting arc $(2_1, 4_2)$ to a higher layer the former optimal solution $T_{L,\text{opt}}$ now is not valid anymore in G''_L. Here the new optimal solution T''_{opt} in G has higher cost and therefore provides an upper bound to T_{opt}.

5 Adaptive Layers Framework (ALF)

To reduce the size of a layered graph G_L we consider a node $v_l \in S_L,\ d_v^{\min} < l < B$ and redirect all incoming arcs $(u_{l-d_{uv}}, v_l) \in A_L,\ u_{l-d_{uv}} \in V_L$, to a node v_k on a different layer $k \neq l$. Then we can safely remove v_l together with all outgoing arcs from G_L since it cannot be reached from the root s anymore and therefore cannot be part of a solution. If we want to obtain a lower bound the layer k of the new target node v_k is set to $\max_{d_v^{\min} \leq i < l}\{i : v_i \in V_L\}$, for an upper bound $k = \min_{l < i \leq B}\{i : v_i \in V_L\}$. In other words, we want to minimize the difference between the original and the redirected target layer. Repeating this redirection process for further Steiner nodes results in a sequence of layered graphs with monotonically decreasing size. In contrast to this reduction process *ALF* goes the other way and starts with the smallest possible layered graph providing a feasible solution to the model *SAL* and extends it iteratively to tighten the bounds. Node set V_L^0 of the initial layered graph G_L^0 just contains root node s, terminal nodes R_L, Steiner nodes $v_{d_v^{\min}},\ v \in R \cup S$, on the lowest feasible layer and Steiner nodes $v_{B-1},\ v \in S$. If necessary, arcs are redirected to the next available layers depending on the desired bound.

The next step is to compute optimal LP and integer solutions $T^i_{L,\text{LP}}$ and $T^i_{L,\text{opt}}$ of model $SAL_{(\text{LP})}$ on the current layered graph G_L^i. For all redirected arcs $(u_l, v_k) \in A_L^i$ with $y_{u_l v_k} > 0$ in $T^i_{L,\text{LP}}$ or $T^i_{L,\text{opt}}$ we extend G_L^i by adding the node $v_{l+d_{uv}}$ together with related outgoing arcs. If necessary, existing arcs pointing to a node $v_l,\ 1 \leq l \leq B$, are modified to either prevent a former redirection or to reduce the difference between the original and the current target layer. The resulting graph is denoted G_L^{i+1}. Applying Lemma 1 and 3 we know that $c(T^{i+1}_{L,\text{opt}}) \geq c(T^i_{L,\text{opt}})$ if redirecting to lower layers and $c(T^{i+1}_{L,\text{opt}}) \leq c(T^i_{L,\text{opt}})$ otherwise. These steps are now repeated on G_L^{i+1} and further graphs until the two bounds match. Algorithm 1 shows this iterative solving process.

When redirecting to lower layers we have to consider that the resulting graph G_L^i does not necessarily need to be acyclic anymore. Therefore $T^i_{L,\text{LP}}$ or $T^i_{L,\text{opt}}$ of model *SAL* may be unconnected and contain cycles. Adding violated directed connection cut inequalities prevents these cycles and therefore may lift the lower bound. When redirecting the arcs to higher layers we are not faced with this

Algorithm 1. Adaptive Layers Framework (*ALF*)

Input: graph $G = (V, E)$
Output: an optimal solution T_{opt} to the DCMT problem

$V_L^0 = s \cup \{v_{d_v^{\min}} : v \in R \cup S\} \cup \{v_{B-1} : v \in S\} \cup R_L$ `// initial node set`
$LB = 0,\ UB = \infty$ `// lower and upper bounds`
redirect = down `// arc redirection` $\in \{\text{down}, \text{up}\}$
$i = 0$
while $LB \neq UB$ **do**
  build G_L^i depending on V_L^i and *redirect*
  $T_{L,\text{LP}}^i = solve(SAL_{\text{LP}}^{(dcut)}(G_L^i))$ `// solve LP`
  $T_{L,\text{opt}}^i = solve(SAL^{(dcut)}(G_L^i))$ `// solve ILP`
  transform $T_{L,\text{opt}}^i$ to T_{opt}^i on G
  if (T_{opt}^i *is feasible*) $\wedge$ (*redirect* == up) **then** $T_{\text{opt}}^i = improve(T_{\text{opt}}^i)$
  if (T_{opt}^i *is feasible*) $\wedge\ (c(T_{\text{opt}}^i) < UB)$ **then** $UB = c(T_{\text{opt}}^i),\ T_{\text{opt}} = T_{\text{opt}}^i$
  if *redirect* == down **then** $LB = c(T_{\text{opt}}^i)$
  $V_L^{i+1} = V_L^i$ `// extend layered graph` G_L^i
  forall $(u_l, v_k) \in T_{L,\text{LP}}^i \cup T_{L,\text{opt}}^i$ **do**
    if $(y_{u_l v_k} > 0) \wedge (k - l \neq d_{uv})$ **then** $V_L^{i+1} = V_L^{i+1} \cup \{v_{l+d_{uv}}\}$
  switch *redirect*, $i = i + 1$
return T_{opt}

problem, so connection cuts cannot improve generated upper bounds in this case. Additionally, every time we obtain a solution feasible in G we try to improve it by heuristic methods to further tighten the global upper bound.

Theorem 2. *Algorithm 1 terminates.*

Proof. As long as the optimal solution $T_{L,\text{opt}}^i$ in graph G_L^i is infeasible for the DCMT problem when calculating a lower bound, $T_{L,\text{opt}}^i$ must contain a redirected arc. Adding an appropriate node v_l to G_L^{i+1} prevents the redirection leading to a new solution $T_{L,\text{opt}}^{i+1}$. In worst case all nodes $v_l \in S_L$ are added to the layered graph resulting in the original graph G_L with optimal solution $T_{L,\text{opt}}$. Since $|S_L|$ is finite the number of iterations is bounded. □

More generally, *ALF* can be (easily) adapted to work for other problems with delay or weight constraints that can be modeled on a directed layered graph. One just has to replace model $SAL^{(dcut)}$ by an appropriate model for the considered problem on the layered graph. To enhance upper bounds some problem specific heuristics could be provided, too.

6 Computational Results

We compared our model $SAL^{(dcut)}$ and framework $ALF^{(dcut)}$ to three existing approaches: the Miller-Tucker-Zemlin based model (*MTZ*), its variant with additional connection cuts from [9] (MTZ^{dcut}) and the MCF formulation on layered

Table 1. Average sizes of preprocessed graphs G, full layered graphs G_L and final layered graphs G_L^I in $ALF^{(dcut)}$ after I iterations (B: delay bound)

						$\overline{I}$		$\overline{V_L^I}$		$\overline{A_L^I}$	
Set	B	$\overline{\lvert V\rvert}$	$\overline{\lvert E\rvert}$	$\overline{\lvert V_L\rvert}$	$\overline{\lvert A_L\rvert}$	ALF	ALF^{dcut}	ALF	ALF^{dcut}	ALF	ALF^{dcut}
R1000	1000	41	401	34219	433632	15	13	681	642	7565	7167
	1500	41	414	54219	843142	16	14	895	852	12363	11909
	2000	41	414	74219	1256542	15	14	760	687	12274	11019
	2500	41	414	94219	1669942	12	11	617	565	10634	9693
C1000	1000	41	572	34221	537767	16	13	831	725	11683	10256
	1500	41	589	54221	1106116	15	13	1384	1170	25110	21519
	2000	41	589	74221	1679516	17	17	1687	1731	34899	35896
	2500	41	589	94221	2252916	13	12	1762	1510	40291	34531
E1000	1000	41	632	34220	565509	18	17	1298	1171	19472	17379
	1500	41	668	54220	1215032	15	13	1685	1453	33291	28787
	2000	41	668	74220	1874432	11	11	1785	1890	42255	44663
	2500	41	668	94220	2533832	10	11	1829	1932	49036	51358

graphs from [6] (*MCFL*). Tests were performed on complete spanning tree instances with 41 nodes introduced in [6]. 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 input graphs varying in the number of possible discrete edge delay values, e.g. C5 denotes the set of instances with five different integer delay values $d_e \in \{1, ..., 5\},\ \forall e \in E$. All tests have been executed on a single core of a multicore system consisting of Intel Xeon E5540 processors with 2.53 GHz and about 3 GB RAM per core. We used IBM ILOG CPLEX 12.1 to solve the (I)LP models. The y variables in model $SAL^{(dcut)}$ and the flow variables in model *MCFL* are declared Boolean since the CPLEX presolver benefits from integrality of these variables and therefore can significantly reduce the model. To reduce the size of the input graphs all preprocessing methods presented in [16] are applied. Solutions obtained in ALF which are feasible in G are improved by a variable neighborhood descent introduced in [15] to further tighten upper bounds. Table 1 exemplarily shows graph sizes of the instance sets with the largest sets of achievable delay values. We list average sizes of preprocessed input graphs, full layered graphs used in model $SAL^{(dcut)}$, and final layered graphs in iteration I of $ALF^{(dcut)}$ when either an optimal solution is found or the time limit is reached.

When iteratively solving (I)LP models in the ALF framework we can provide tight upper bounds to CPLEX obtained in previous calculations supporting the presolving phase and pruning of the branch-and-bound tree. According to Lemma 2 it would be enough to iteratively compute lower bounds in ALF. But repeated switching between lower and upper bound turned out to work well in practice speeding up the convergence. Finally, in case of limited runtime ALF usually yields small gaps and obtains feasible solutions by computing both bounds. Some tests were performed to initialize the first layered graph in a more sophisticated way, e.g. based on heuristic solutions, but following the proposed trivial way mostly yields the best results.

Table 2. Comparison of different ILP models on test sets from [6] (B: delay bound, O: number of optimal solutions (out of 5), *gap*: average gap in percent, *time*: median CPU time in seconds; time limit: 10000 seconds; best results are printed bold)

		MTZ			*MTZ*^{dcut}			*MCFL*			*SAL*			*SAL*^{dcut}			*ALF*			*ALF*^{dcut}		
Set	B	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*	O	*gap*	*time*
R2	3	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**
	5	**5**	**0.0**	11	**5**	**0.0**	13	**5**	**0.0**	19	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1
	7	**5**	**0.0**	7	**5**	**0.0**	6	**5**	**0.0**	91	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	9	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	677	**5**	**0.0**	1	**5**	**0.0**	1	**5**	**0.0**	2	**5**	**0.0**	2
C2	3	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**
	5	3	1.1	6403	3	1.1	1627	**5**	**0.0**	9	**5**	**0.0**	2	**5**	**0.0**	**0**	**5**	**0.0**	5	**5**	**0.0**	3
	7	4	0.9	3487	**5**	**0.0**	335	**5**	**0.0**	891	**5**	**0.0**	5	**5**	**0.0**	**2**	**5**	**0.0**	10	**5**	**0.0**	6
	9	**5**	**0.0**	1074	**5**	**0.0**	31	**5**	**0.0**	1794	**5**	**0.0**	7	**5**	**0.0**	**3**	**5**	**0.0**	20	**5**	**0.0**	16
E2	3	**5**	**0.0**	10	**5**	**0.0**	11	**5**	**0.0**	1	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	5	0	9.9	10000	0	11.6	10000	**5**	**0.0**	274	**5**	**0.0**	19	**5**	**0.0**	**4**	**5**	**0.0**	29	**5**	**0.0**	34
	7	0	10.6	10000	0	9.7	10000	0	82.1	10000	**5**	**0.0**	375	**5**	**0.0**	**19**	**5**	**0.0**	1512	**5**	**0.0**	715
	9	0	7.4	10000	0	5.4	10000	0	100.0	10000	**5**	**0.0**	1155	**5**	**0.0**	**44**	4	0.3	1944	**5**	**0.0**	843
R5	6	**5**	**0.0**	15	**5**	**0.0**	17	**5**	**0.0**	17	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	**0**
	8	**5**	**0.0**	44	**5**	**0.0**	28	**5**	**0.0**	195	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	10	**5**	**0.0**	49	4	1.1	30	**5**	**0.0**	561	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	2	**5**	**0.0**	2
	12	**5**	**0.0**	24	**5**	**0.0**	16	**5**	**0.0**	1119	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	2	**5**	**0.0**	2
C5	6	**5**	**0.0**	24	**5**	**0.0**	34	**5**	**0.0**	8	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	8	4	1.1	714	4	0.7	564	**5**	**0.0**	109	**5**	**0.0**	1	**5**	**0.0**	**0**	**5**	**0.0**	2	**5**	**0.0**	1
	10	2	2.4	10000	2	1.5	10000	3	40.0	3269	**5**	**0.0**	4	**5**	**0.0**	**2**	**5**	**0.0**	15	**5**	**0.0**	10
	12	1	2.2	10000	**5**	**0.0**	1416	4	0.2	6723	**5**	**0.0**	9	**5**	**0.0**	**6**	**5**	**0.0**	36	**5**	**0.0**	32
E5	6	**5**	**0.0**	2437	**5**	**0.0**	7258	**5**	**0.0**	77	**5**	**0.0**	2	**5**	**0.0**	**1**	**5**	**0.0**	7	**5**	**0.0**	7
	8	0	7.5	10000	0	7.0	10000	**5**	**0.0**	897	**5**	**0.0**	6	**5**	**0.0**	**3**	**5**	**0.0**	29	**5**	**0.0**	14
	10	0	9.1	10000	0	8.5	10000	3	40.0	4067	**5**	**0.0**	18	**5**	**0.0**	**13**	**5**	**0.0**	62	**5**	**0.0**	54
	12	0	9.3	10000	0	8.0	10000	1	80.0	10000	**5**	**0.0**	125	**5**	**0.0**	**52**	**5**	**0.0**	686	**5**	**0.0**	335
R10	10	**5**	**0.0**	45	**5**	**0.0**	77	**5**	**0.0**	149	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	1	**5**	**0.0**	1
	15	**5**	**0.0**	129	**5**	**0.0**	322	4	20.0	2291	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	3	**5**	**0.0**	3
	20	**5**	**0.0**	63	**5**	**0.0**	90	3	40.0	8539	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	5	**5**	**0.0**	4
	25	**5**	**0.0**	28	4	0.6	56	0	100.0	10000	**5**	**0.0**	**3**	**5**	**0.0**	**3**	**5**	**0.0**	8	**5**	**0.0**	5
C10	10	**5**	**0.0**	70	**5**	**0.0**	45	**5**	**0.0**	220	**5**	**0.0**	**0**	**5**	**0.0**	**0**	**5**	**0.0**	2	**5**	**0.0**	2
	15	4	0.8	1034	**5**	**0.0**	481	3	40.0	3212	**5**	**0.0**	**2**	**5**	**0.0**	**2**	**5**	**0.0**	7	**5**	**0.0**	5
	20	2	2.6	10000	3	0.8	3485	0	100.0	10000	**5**	**0.0**	25	**5**	**0.0**	**18**	**5**	**0.0**	37	**5**	**0.0**	32
	25	1	2.2	10000	**5**	**0.0**	2801	0	100.0	10000	**5**	**0.0**	141	**5**	**0.0**	**84**	**5**	**0.0**	200	**5**	**0.0**	167
E10	10	4	0.5	2590	**5**	**0.0**	2918	**5**	**0.0**	301	**5**	**0.0**	**1**	**5**	**0.0**	**1**	**5**	**0.0**	5	**5**	**0.0**	6
	15	0	10.0	10000	0	11.0	10000	0	68.0	10000	**5**	**0.0**	**14**	**5**	**0.0**	24	**5**	**0.0**	140	**5**	**0.0**	112
	20	0	10.1	10000	0	10.0	10000	0	100.0	10000	**5**	**0.0**	590	**5**	**0.0**	**267**	**5**	**0.0**	3320	**5**	**0.0**	2089
	25	0	9.1	10000	0	7.1	10000	0	100.0	10000	2	1.5	10000	**5**	**0.0**	**1105**	2	1.6	10000	3	1.9	9043
R100	100	4	4.5	129	4	4.8	177	0	100.0	10000	**5**	**0.0**	10	**5**	**0.0**	**8**	**5**	**0.0**	14	**5**	**0.0**	11
	150	4	5.6	56	4	6.5	131	0	100.0	10000	**5**	**0.0**	14	**5**	**0.0**	15	**5**	**0.0**	**5**	**5**	**0.0**	6
	200	4	1.3	34	4	2.6	143	0	100.0	10000	**5**	**0.0**	27	**5**	**0.0**	34	**5**	**0.0**	**10**	**5**	**0.0**	11
	250	**5**	**0.0**	9	**5**	**0.0**	9	0	100.0	10000	**5**	**0.0**	70	**5**	**0.0**	66	**5**	**0.0**	12	**5**	**0.0**	**6**
C100	100	2	3.0	10000	3	2.2	8651	0	100.0	10000	**5**	**0.0**	**67**	**5**	**0.0**	122	**5**	**0.0**	89	**5**	**0.0**	87
	150	0	4.1	10000	2	1.2	10000	0	100.0	10000	**5**	**0.0**	1027	4	11.1	1824	**5**	**0.0**	**294**	**5**	**0.0**	389
	200	1	2.4	10000	3	0.7	2539	0	100.0	10000	2	2.3	10000	1	54.0	10000	**5**	**0.0**	1785	**5**	**0.0**	**1756**
	250	3	1.3	3423	**5**	**0.0**	**141**	0	100.0	10000	1	5.0	10000	0	59.3	10000	4	0.2	2007	4	0.1	4139
E100	100	0	8.3	10000	0	7.3	10000	0	100.0	10000	**5**	**0.0**	886	**5**	**0.0**	1211	**5**	**0.0**	**717**	**5**	**0.0**	1150
	150	0	12.1	10000	0	9.4	10000	0	100.0	10000	1	6.7	10000	0	51.8	10000	**2**	1.6	10000	**2**	**0.5**	10000
	200	**0**	9.5	10000	**0**	7.5	10000	**0**	100.0	10000	**0**	11.8	10000	**0**	72.5	10000	**0**	4.5	10000	**0**	**2.0**	10000
	250	**0**	7.3	10000	**0**	5.5	10000	**0**	100.0	10000	**0**	12.3	10000	**0**	73.6	10000	**0**	5.9	10000	**0**	**3.2**	10000
R1000	1000	3	4.3	725	3	7.9	2294	0	100.0	10000	3	6.0	8035	3	34.5	7217	**5**	**0.0**	**24**	**5**	**0.0**	46
	1500	1	4.0	10000	1	9.0	10000	0	100.0	10000	1	20.2	10000	1	74.3	10000	**5**	**0.0**	**97**	**5**	**0.0**	174
	2000	4	1.3	310	3	3.6	577	0	100.0	10000	1	47.3	10000	0	92.9	10000	**5**	**0.0**	44	**5**	**0.0**	**32**
	2500	**5**	**0.0**	24	4	0.6	33	0	100.0	10000	0	77.3	10000	1	75.1	10000	**5**	**0.0**	17	**5**	**0.0**	**13**
C1000	1000	4	2.0	3208	4	1.9	2220	0	100.0	10000	1	7.8	10000	1	41.1	10000	**5**	**0.0**	30	**5**	**0.0**	**16**
	1500	0	5.6	10000	0	4.0	10000	0	100.0	10000	0	15.2	10000	0	75.1	10000	**5**	**0.0**	511	**5**	**0.0**	**280**
	2000	0	6.1	10000	0	3.9	10000	0	100.0	10000	0	77.7	10000	0	78.0	10000	**3**	0.4	9556	**3**	**0.1**	**5152**
	2500	1	2.6	10000	4	0.6	**896**	0	100.0	10000	0	76.8	10000	0	87.7	10000	**5**	**0.0**	1389	**5**	**0.0**	3545
E1000	1000	0	8.3	10000	1	8.4	10000	0	100.0	10000	0	14.6	10000	0	71.0	10000	**5**	**0.0**	**840**	4	0.3	4919
	1500	0	10.2	10000	0	8.8	10000	0	100.0	10000	0	57.4	10000	0	79.9	10000	**3**	1.8	**8383**	2	**1.3**	10000
	2000	0	9.5	10000	0	7.4	10000	0	100.0	10000	0	75.1	10000	0	82.5	10000	0	3.9	10000	**1**	**1.2**	10000
	2500	**0**	7.6	10000	**0**	5.3	10000	**0**	100.0	10000	**0**	93.3	10000	**0**	90.0	10000	**0**	5.0	10000	**0**	**1.6**	10000

Table 2 provides computational results and a comparison between all models applied on the described test sets. We present the numbers of found optimal solutions, the average gaps between lower and upper bounds and the median runtimes in seconds if the method finished before reaching the time limit of 10000 seconds. As already mentioned in [6] the E instances are much harder to solve than the other instances which can easily be seen in the results especially when considering instances with large sets of different delay values and high bounds. Model SAL^{dcut} provides extremely tight LP relaxation values, see [3], where branching is hardly necessary to compute optimal integer solutions. Nevertheless, one has to consider the additional runtime for searching violated cuts which in most cases turned out to be more efficient than branching. The only advantage of the *MTZ* formulations is the independence of actual delay values and bounds. But even so it is not competitive to our approaches in almost all cases. Model *MCFL* obviously suffers from the huge amount of flow variables and therefore it is rarely applicable to the used instances. Most of the time solving model $SAL^{(dcut)}$ outperforms all other methods, but when huge sets of achievable delay values or high bounds arise, *ALF* is clearly superior. The overhead of iteratively computing small ILP models becomes worth when even LP relaxations of model $SAL^{(dcut)}$ on the full layered graph are hard to solve. Even though *ALF* is mostly slower than solving model $SAL^{(dcut)}$ it provides tight gaps and robust performance throughout all test sets. Furthermore, *ALF* consumes substantially less memory since the graphs it works on are significantly smaller than the full layered graphs, see Table 1.

7 Conclusions and Future Work

We presented two approaches to solve delay-constrained minimum tree problems based on using an appropriate layered graph. The first ILP model utilizes the special structure of this graph, mainly its acyclicity, to reduce the number of necessary variables and constraints. It provides excellent results in many cases except on instances with huge sets of possible discrete edge delay values and high bounds since the size of the layered graph heavily depends on these properties. The second approach – an algorithmic framework – tries to tackle exactly these issues. By computing lower and upper bounds for the optimal solution value on reduced layered graphs it obtains small gaps and shows robust performance throughout all test sets. Besides consuming significantly less memory it even yields tight bounds in cases where it is not possible to compute LP relaxations of the model on the full layered graph in reasonable time.

In future we try to combine heuristic methods with our adaptive layers framework to further speed up the convergence and reduce the number of necessary iterations. Also, we want to embed more sophisticated state-of-the-art solvers for the STP to additionally improve our framework. For the sake of a more comprehensive comparison we intend to re-implement the column generation and Lagrangian relaxation approach from [6]. Last but not least, we plan to adapt our framework for other optimization problems.

References

1. de Aragão, M., Uchoa, E., Werneck, R.: Dual heuristics on the exact solution of large Steiner problems. Electronic Notes in Discrete Mathematics 7, 150–153 (2001)
2. Dreyfus, S.E., Wagner, R.A.: The Steiner problem in graphs. Networks 1, 195–207 (1971)
3. Gouveia, L., Simonetti, L., Uchoa, E.: Modeling hop-constrained and diameter-constrained minimum spanning tree problems as Steiner tree problems over layered graphs. Mathematical Programming, pp. 1–26 (2010)
4. Gouveia, L.: Using Variable Redefinition for Computing Lower Bounds for Minimum Spanning and Steiner Trees with Hop Constraints. Informs Journal on Computing 10(2), 180–188 (1998)
5. Gouveia, L.: Using hop-indexed models for constrained spanning and Steiner tree models, pp. 21–32. Kluwer Academic Publishers, Dordrecht (1999)
6. Gouveia, L., Paias, A., Sharma, D.: Modeling and Solving the Rooted Distance-Constrained Minimum Spanning Tree Problem. Computers and Operations Research 35(2), 600–613 (2008)
7. Koch, T., Martin, A.: Solving Steiner tree problems in graphs to optimality. Networks 32(3), 207–232 (1998)
8. Kompella, V.P., Pasquale, J.C., Polyzos, G.C.: Multicast routing for multimedia communication. IEEE / ACM Transactions on Networking 1(3), 286–292 (1993)
9. Leggieri, V., Haouari, M., Triki, C.: An Exact Algorithm for the Steiner Tree Problem with Delays. Electronic Notes in Discrete Mathematics 36, 223–230 (2010)
10. Ljubic, I., Gollowitzer, S.: Modelling the hop constrained connected facility location problem on layered graphs. In: Electronic Notes in Discrete Mathematics, vol. 36, pp. 207–214. Elsevier, Amsterdam (2010)
11. Ljubic, I., Weiskircher, R., Pferschy, U., Klau, G., Mutzel, P., Fischetti, M.: An algorithmic framework for the exact solution of the prize-collecting Steiner tree problem. Mathematical Programming 105(2), 427–449 (2006)
12. Manyem, P., Stallmann, M.: Some approximation results in multicasting. Tech. Rep. TR-96-03, North Carolina State University (1996)
13. Nastansky, L., Selkow, S., Stewart, N.: Cost-minimal trees in directed acyclic graphs. Mathematical Methods of Operations Research 18(1), 59–67 (1974)
14. Robins, G., Zelikovsky, A.: Improved Steiner tree approximation in graphs. In: SODA 2000: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms, pp. 770–779. Society for Industrial and Applied Mathematics (2000)
15. Ruthmair, M., Raidl, G.R.: A Kruskal-Based Heuristic for the Rooted Delay-Constrained Minimum Spanning Tree Problem. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) EUROCAST 2009. LNCS, vol. 5717, pp. 713–720. Springer, Heidelberg (2009)
16. Ruthmair, M., Raidl, G.R.: Variable Neighborhood Search and Ant Colony Optimization for the Rooted Delay-Constrained Minimum Spanning Tree Problem. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 391–400. Springer, Heidelberg (2010)
17. Xu, Y., Qu, R.: A GRASP approach for the Delay-constrained Multicast routing problem. In: Proceedings of the 4th Multidisplinary International Scheduling Conference (MISTA4), Dublin, Ireland, pp. 93–104 (2009)
18. Xu, Y., Qu, R.: A hybrid scatter search meta-heuristic for delay-constrained multicast routing problems. Applied Intelligence, 1–13 (2010)
19. Zelikovsky, A.: A series of approximation algorithms for the acyclic directed Steiner tree problem. Algorithmica 18(1), 99–110 (1997)