A Kruskal-Based Heuristic for the Rooted Delay-Constrained Minimum Spanning Tree Problem

Mario Ruthmair and Günther R. Raidl

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

1 Introduction

The rooted delay-constrained minimum spanning tree (RDCMST) problem is an NP-hard [1] combinatorial optimization problem arising for example in the design of centralized broadcasting networks, where quality of service constraints are of concern. The objective is to find a minimum cost spanning tree of a given graph with the additional constraint that the sum of delays along the paths from a specified root node to any other node must not exceed a given delay-bound.

More formally, we are given a graph $G = (V, E)$ with a set of nodes V, a set of edges E, a cost function $c : E \rightarrow \mathbb{R}^+$, a delay function $d : E \rightarrow \mathbb{R}^+$, a fixed root node s and a delay-bound B. An optimal solution to the RDCMST problem is a spanning tree $T = (V, E'),\ E' \subset E$, with minimum cost $c(T) = \sum_{e \in E'} c(e)$, satisfying the constraints: $\sum_{e \in P(s,v)} d(e) \leq B,\ \forall v \in V$. $P(s, v)$ denotes the unique path from the specified root node s to a node $v \in V$.

Exact approaches to the RDCMST problem have been examined by Gouveia et al. in [2], but these methods can only solve small graphs with less than 100 nodes to proven optimality in reasonable time if considering complete instances. A heuristic approach was presented by Salama et al. in [1], where a construction method based on Prim's algorithm to find a minimum spanning tree [3] is described. This Prim-based heuristic starts from the root node and iteratively connects the node which can be reached in the cheapest way without violating the delay-constraint. If at some point no node can be connected anymore, the delays in the existing tree are reduced by replacing edges. These steps are repeated until a feasible RDCMST is obtained. A second phase improves the solution by local search using the edge-exchange neighborhood structure.

A general problem of the Prim-based heuristic is the fact that the nodes in the close surrounding of the root node are connected rather cheaply, but at the same time delay is wasted, and so the distant nodes can later only be linked by many, often expensive edges, see Fig. 1. The stricter the delay-bound the more this drawback will affect the costs negatively. This fact led us to a more de-centralized approach by applying the idea of Kruskal's minimum spanning tree algorithm [4] to the RDCMST problem.

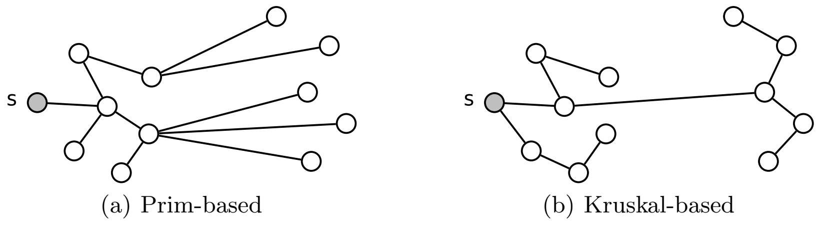

Fig. 1. Problem of Prim-based heuristic (a) compared to Kruskal-based heuristic (b).

2 Kruskal-Based Construction Heuristic

Our Kruskal-based heuristic constructs a solution in two stages. Stage one sorts all edges by ascending costs and then iterates over this list. An edge is added to the solution if no cycle is created and if it is possible to connect the newly created component to the root node in any way without violating the delay-constraint. To speed up this feasibility-check additional information has to be stored in each node. This information consists of the shortest-path-delay to the root node and the maximum delay to any other node in the same component.

At the end of stage one the graph need not to be connected, so in stage two the left components are attached to each other by adding the shortest-delay-path from the root node to each component. Resulting cycles have to be dissolved to form a feasible tree.

The runtime of stage one is in $\mathcal{O}(|E|\log|E|+|V|^2)$, stage two runs in $\mathcal{O}(|V|^3)$.

3 Variable Neighborhood Descent

Additionally we use a variable neighborhood descent (VND) [5] for improving a constructed solution. The VND uses two neighborhoods: Expensive-Edge-Removing (EER) and Component-Renewing (CR).

A move in the EER-neighborhood removes the most expensive edge and connects the resulting two components in the cheapest possible way.

A CR-neighborhood-move also deletes the most expensive edge, but completely dissolves the component which does not include the root node; it then re-adds the individual nodes by applying Prim's algorithm. As before in some cases not all single nodes can be added due to the delay-bound. These left nodes are again added by shortest-delay-paths, dissolving created cycles.

4 Preliminary Results

Our Kruskal-based heuristic for the RDCMST problem – applied to large complete instances of 500 nodes – produces usually significantly better solutions than the Prim-inspired algorithm, especially if the delay-constraint is strict, see the exemplary result in Table 1.

Table 1. Comparison of Kruskal- and Prim-based heuristic, applied on a set of 30 random complete instances with 500 nodes (B denotes the delay-bound). The results of the tests using the GRASP-extension are average values of 30 runs stopping after five iterations without gain.

	Prim			Prim +VND			Kruskal			Kruskal +VND			Kruskal +GRASP+VND		
B	$\overline{c}$	σ	$t[s]$	$\overline{c}$	σ	$t[s]$	$\overline{c}$	σ	$t[s]$	$\overline{c}$	σ	$t[s]$	$\overline{c}$	σ	$t[s]$
6	19790	1734	3.4	9611	630	25.7	**10791**	654	0.2	**9498**	660	15.3	8901	711	197.7
10	9483	1628	0.3	5781	662	23.8	**7075**	338	0.2	**4797**	221	18.7	4462	227	242.9
15	5831	1072	0.2	3862	422	24.8	**5551**	431	0.2	**3068**	173	20.8	2872	118	260.5
20	**4229**	841	0.1	2940	389	23.8	4737	388	0.2	**2309**	102	21.3	2182	88	292.5

Concerning the runtime our construction heuristic can compete with the Prim-based approach, although the administration effort is higher when updating the node information in each step of stage one. We can observe that the runtime is nearly independent of the specified delay-bound B in contrast to the Prim-based heuristic, where tight bounds lead to longer runtimes due to the repeated delay-relaxation process, see Table 1.

Results when also applying the VND indicate that the Kruskal-based solution is a more promising starting point for improvement with the EER- and CR-neighborhoods than the Prim-based solution.

To improve the quality of the results we used a greedy randomized adaptive search procedure (GRASP) [6] in stage one of the Kruskal-based construction heuristic to provide more different starting solutions for the VND.

References

1. Salama, H.F., Reeves, D.S., Viniotis, Y.: The delay-constrained minimum spanning tree problem. In Blum, C., Roli, A., Sampels, M., eds.: Proceedings of the 2nd IEEE Symposium on Computers and Communications – ISCC '97. (1997) 699–703
2. Gouveia, L., Paias, A., Sharma, D.: Modeling and solving the rooted distance-constrained minimum spanning tree problem. Computers and Operations Research **35**(2) (2008) 600–613
3. Prim, R.C.: Shortest connection networks and some generalizations. Bell System Technical Journal **36** (1957) 1389–1401
4. Kruskal, J.B.: On the shortest spanning subtree of a graph and the traveling salesman problem. Proceedings of the American Mathematics Society **7**(1) (1956) 48–50
5. Hansen, P., Mladenović, N.: Variable neighborhood search: Principles and applications. European Journal of Operational Research **130**(3) (2001) 449–467
6. Feo, T., Resende, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization **6**(2) (1995) 109–133