The 7th International Symposium on Operations Research and Its Applications (ISORA'08)
Lijiang, China, October 31–Novemver 3, 2008

Computing Lower Bounds for Steiner Trees in Road Network Design

Justus Schwartz[1] Jürg Stückelberger[2]

[1]ETH Zürich, Institute of Theoretical Computer Science, ETH Zentrum, CH 8092 Zürich, Switzerland, E-mail : justuss@inf.ethz.ch
[2]ETH Zürich, Chair of Land Use Engineering, ETH Zentrum, CH 8092 Zürich, Switzerland, E-mail : stueckelberger@env.ethz.ch

Abstract In the automated design of road networks an important sub-task is the solution of an instance of the *Steiner tree problem*, a well known $\mathcal{NP}$-hard problem. In this paper we present the experimental evaluation of a heuristic for the *Steiner tree problem* used in the design of low volume forest road networks. To evaluate a heuristic it is necessary to compare to the optimum value or a good lower bound on the optimum. Thus, the focus of this paper is on a new approach to compute such lower bounds for the Steiner tree problem in graphs emanating from the automated road network design problem. We evaluated the heuristic on data from a mountainous region of the Swiss Alps. The algorithms and the model were developed in a collaborative project between the Institute of Theoretical Computer Science and the Chair of Land Use Engineering at ETH Zurich.

1 Introduction

The problem of computing Steiner minimum trees is a well studied topic. In this work we investigate the performance of heuristics applied to graphs originating from road design for low volume forest road networks. As these graphs are huge the focus is on efficiently computing good lower bounds to experimentally evaluate the applied heuristics. A main point is that the number of terminals is relatively small but the graphs are very large.

Previously, one of us showed in [9] how to overcome limitations of previous approaches by modeling the spatial variability in road construction costs, and improving the model representation of forest roads by graphs ([10], [11]). To design a forest road network we build on this previous work and obtain a weighted graph which represents the road construction costs; this is described in detail in Section 2. Then the problem of designing a road network of minimal cost, which connects a given set of points, is equivalent to finding a *Steiner Minimum Tree* (SMT) in the modeled graph. Here the two main difficulties, which we will address in this paper, arise:

1. Since the *Steiner tree problem* is $\mathcal{NP}$-hard and $\mathcal{APX}$-complete, we have to use heuristics or approximation algorithms. Furthermore our graphs do not even satisfy the triangle inequality and we, therefore, have to use heuristics which can only guarantee a constant factor approximation. For the heuristics of practical importance this is a factor of two.

2. In order to evaluate the used heuristics a good lower bound has to be computed. This is especially important as the road construction costs are high (≈400$ per meter) and the theoretical approximation guarantees are not good enough, as they would only guarantee that we pay at most twice as much as we have to.

To our knowledge all known algorithms to compute good lower bounds are either based on linear programming or do use the approximation guarantees of approximation algorithms. As the number of terminals is small in our case we were able to use a different approach by exploiting the geometric structure and use dynamic programming to obtain a lower bound.

This paper is organized as follows. In the next section we will describe the application and the resulting model. In Section 3 we present the used heuristic, which is an improvement on the one used in [10]. Then in Section 4 we introduce our new approach to compute the lower bound, before in Section 5 we present our experimental results. To ease presentation we will not present the directed graph version of our result (computing a *Steiner arborescence*) but state the algorithms for the undirected Steiner tree problem and note that the discussion for the directed version remains basically the same.

2 Application and Model

The methods described in [10] for automatically locating road networks need as a first step the discretiziation of road segments in a high resolution to map road-geometry constraints on a graph $G = (V, E)$. Further from the definition of the objectives – mainly the construction costs – the edge weights are derived. The second step is to solve the graph optimization problem, that is to find a good approximation of the *Steiner tree problem* for the given terminals.

Graph Model

Road engineers control the geometry of a road layout by following a sequence of vectors, known as a 'traverse'. The road is then designed as a series of curves inside the angles between consecutive vectors [4]. In order to map the problem to a discrete graph the geographical area is discretized. The prevailing approach utilizes a regular grid. The land use engineering research group at ETH Zurich has been using a $10\ m \times 10\ m$ resolution ([6], [10]) since in mountainous and heterogeneous areas, that grid size has proved to be appropriate for low-volume roads because it equals the lower limits of the design elements. A vertex is then connected to neighbouring vertices according to a link pattern introduced in [10]. Roads must fulfill technical requirements for safe truck traffic, especially by including a gradient that is lower than the maximum allowed and by having a minimal curve radius along the road centerline. The gradient is easily computed from the digital elevation model and the curve of a road is limited by the minimum turning clearance circle of a vehicle. For traffic comfort, the designed minimal curve radius is about two or three times the technical minimum turning clearance. Thus, we have to check if the radius of each curve is above that minimum allowable in order to evaluate the feasibility of the horizontal road centerline. This implies that incoming edges and outgoing edges in the graph model have to obey some direction constraints. This is achieved by introducing 16 vertices for each square of the project area. Each of these vertices represents the same point in the project area but a different orientation of the incoming road. Now we can

check for each edge if it has a feasible curvature and gradient and depending on this add it to the graph or not, see [11].

Thus, our graph has 16 directional layers and for each 10 $m \times 10$ m square of the project area there is a vertex in each layer. For a typical project area of approximately $35km^2$ this leads to a graph with 5,600,000 vertices and approximately 80,000,000 edges, using the link pattern introduced in [10].

Road construction costs model

The most important achievement of [9] opposed to previous work was to model the road construction costs and the maintenance cost using all digitally available data, instead of assigning a cost just based on the road length. These estimates are based on five input factors: 1. a digital elevation model, 2. classification of geotechnical properties of the subsoil, 3. specification of road design parameters, 4. unit costs for structural components, 5. a rock-excavation share model ([9]). Further the cost estimation of roads include structures for embankment, retaining and support, pavement, and drainage and stream-crossing. This leads to a cost $w(u,v)$ assigned to each edge $(u,v) \in E$.

The Steiner Tree Problem

In the remainder of this section we will formalize the Steiner tree problem and introduce some notation that we will need to describe our approach.

In the following sections we will use the following notation for ease of presentation. Let p_v be the point in the project area corresponding to the vertex $v \in V$. We will not explicitly say "the point p_v corresponding to vertex $v \in V$" but instead say the point v. Further we will write $\forall v \in A$ for some area A instead of, "for all vertices corresponding to points in A".

In the remainder of this paper we will for a graphs $G = (V,E)$ always set $n := |V|$ and $m := |E|$. Further we define the weight of a subgraph as the sum of the weights of its edges. The distance $d(u,v)$ between vertex u and v is the length of the shortest path between u and v. Furthermore the points which have to be connected by the road network are called *terminals* and we will denote the set of terminals by $Z \subset V$ and define $k := |Z|$.

The *Steiner tree problem* is the problem of finding a *Steiner minimum tree*: Given a Graph $G = (V,E)$, a set $Z \in V$ and a weight function $w : E \mapsto \mathbb{R}^+$, the *Steiner minimum tree* (SMT) is a connected subgraph T of minimum weight containing all the vertices in Z.

The vertices of T with degree larger than two which are not in Z are called *Steiner points*. Thus, our problem is exactly the Steiner tree problem and the Steiner points are additional junctions in the road network. In the next section we will describe the heuristic used to computed the approximate solutions before we show in Section 4 how we obtained good lower bounds on the optimum value to evaluate our approach.

3 Heuristic

As graphs emanating from the automated road network design problem have several millions of vertices and edges it was not possible to compute the optimum solution. Despite the fact that the Steiner tree problem is polynomially solvable for a constant number of terminals, our instances are too large, such that even the tables needed in a dynamic

programming approach are too large to fit into the memory of current standard computers (even for 10 terminals). Further, all the known reduction techniques did not decrease the problem sizes significantly, which is the basis of many of the best currently known techniques, see [1],[8]. As the Steiner tree problem is $\mathcal{NP}$-hard there is a long history of heuristics and approximation algorithms. In our case not even the triangle inequality is satisfied, and therefore the beautiful approximation algorithms for the geometric variant are not applicable. For a survey of different heuristics see [7] and [8]. The main difficulty in computing the SMT is to determine the Steiner points. Once given the Steiner points the SMT can be found by computing the MST in the distance graph ([7]). On the other hand we will show that given a set C of candidates for Steiner points, the SMT restricted to use only Steiner points from the set C is easily computed using a dynamic programming approach. As the running time grows exponentially in $|Z|$ we stress that this method is only applicable if the number of terminals is small, which, in the case of our road networks, is satisfied. The number of terminals in a typical instance for the forest road network problem is supposed to be at most eleven. There have been different heuristics proposed in the literature to find good Steiner points. As we have to avoid the computation of the all pairs shortest paths, we use a variant of the average distance heuristic similar to the one proposed by Chen [2]. Our heuristic works in the following way:

1. In time $\mathcal{O}(k(m+n\log n))$ the distance from each terminal to all vertices in V can be computed using Dijkstra's Algorithm. And for each triple of terminals (z_1,z_2,z_3) the optimal Steiner point is the vertex v for which $d(z_1,v)+d(z_2,v)+d(z_3,v)$ is minimal. For each triple of terminals the optimal Steiner point is added to the set C of candidates. Therefore the overall running time for this step is $\mathcal{O}\left(k(m+n\log n)+k^3n\right)$ This procedure may be iterated to improve the solution as long as $|C|$ remains small.

2. Then for each $u,v \in C$ the distance $d(u,v)$ is computed. This can be done in $\mathcal{O}(|C|(m+n\log n))$. Finally a tree of minimal weight which can use only Steiner points from the set C is computed using a dynamic programming (DP) approach, which is a variant of the Algorithm by Dreyfuss and Wagner [3]. The running time of this algorithm, which we will describe in more detail in the remainder of the section, is

$$\mathcal{O}\left(3^k|C|+2^k|C|^2+|C|n\log n+|C|m\right).$$

In the following we will describe the idea of the DP and give a proof of its correctness. The proof is basically the same as for the original DP but we present it here as we will need some notation for the presentation of the lower bound.

For each subset $X \subset Z$ and $v \in C \cup Z$ we denote by $D(X,v) := D(X \cup \{v\})$ the weight of the minimum weight tree for the terminal set $X \cup \{v\}$ which uses only Steiner points from the set C. And we write $D_2(X,v)$ for the weight of the minimum weight tree of the terminal set $X \cup \{v\}$ under the extra condition that the degree of v in the tree is at least 2. As v has degree at least 2 in the tree for $D_2(X,v)$, it can be obtained by merging two optimal trees $D(Y,v)$ and $D(X \setminus Y,v)$

for $\emptyset \subsetneq Y \subsetneq X$.Therefore we have the following relation.

$$D_2(X,v) = \min_{\emptyset \subsetneq Y \subsetneq X} (D(Y,v)+D(X \setminus Y,v)) \tag{1}$$

On the other hand there are three possible configurations for $D(X,v)$:

1. If v has degree at least 2 then $D(X,v) = D_2(X,v)$.

2. If v has degree 1 than v is either joined by a shortest path to a vertex $u \in C$ (as the Steiner points are only from the set C) of degree at least 2, that is $D(X,v) = d(v,u) + D_2(X,u)$

3. or v is joined to a terminal $u \in X$ and we have $D(X,v) = d(v,u) + D(X)$. Thus, we have

$$D(X,v) = \min\left(D_2(X,v), \min_{u \in C \setminus X}(d(v,u) + D_2(X,u)), \min_{u \in X}(d(v,u) + D(X))\right). \quad (2)$$

For the complete algorithm see Algorithm DWSET. This reduces to the original Dreyfuss Wagner Algorithm by setting $C = V$. In earlier work on this road network model ([10]) the same candidate points were computed, but then, instead of using the dynamic program to compute the optimum tree with respect to this set of Steiner points either all possible subsets of candidates were enumerated or a "good" solution was found using simulated annealing. Besides computing the optimal solution for C our solution is also faster.

4 Lower Bound Computation

As our input graphs are very large and reduction techniques did not reduce the instances significantly we could not use linear programming formulations to compute lower bounds. Instead we used the clustering of vertices given by the geometry. The general idea was that the distance between two vertices u and v does not differ two much from the distance of u' and v' if u,u' and v,v' respectively are close to each other. The reduction tests we tried were all *alternative based* reductions (see Polzin [8]).

Our method is based on a quadtree partition of the project area. A quadtree is a rooted tree, where every internal vertex has four children. A vertex in the quadtree corresponds to a rectangle and if a vertex has children they correspond to the four quadrants of the rectangle.

The rectangles corresponding to the leaf vertices of the tree form a subdivision of the rectangle of the root vertex. In the following we will call the rectangles of the quadtree *quads*, and assume that the rectangle of the root vertex covers the project area.
If Q_1, Q_2 are two quads then we define the distance between the quads as $d(Q_1,Q_2) := \min_{u \in Q_1, v \in Q_2}(d(u,v))$. Further the distance between a vertex u and a quad Q is $d(u,Q) := \min_{v \in Q}(d(u,v))$.

We denote the set of quads defining the subdivision, i.e. the leafs of the quad tree, by QT. For a vertex v let $B(v)$ be the quad in QT containing v. Our first important observation is that if T is a tree with edge weights $d(.)$ on the vertices $v_1,\ldots,v_s$, if T_B is the corresponding tree on the quads $B(v_1),\ldots,B(v_s)$ then we have

$$w(T) = \sum_{(u,v)\in T} d(u,v) \geq \sum_{(u,v)\in T} d(B(u),B(v)) = \sum_{(u_B,v_B)\in T_B} d(u_B,v_B).$$

Therefore the weight of a tree on vertices from V is always greater then the weight of the corresponding tree on the quads. As further the number of Steiner points in an SMT is at most $k-2$ and each Steiner point has degree at least 3 we have

$$w(T) = \sum_{(u,v)\in T} d(u,v) \geq \sum_{(u,v)\in T} d(B(u),B(v)) = \sum_{(u_B,v_B)\in T_B} d(u_B,v_B).$$

Therefore the weight of a tree on vertices from V is always greater then the weight of the corresponding tree on the quads. As further the number of Steiner points in an SMT is at most $k-2$ and each Steiner point has degree at least 3 we have

$$w(SMT) \geq \min \left\{ w(T) \;\middle|\; \begin{array}{l} T \text{ is a tree on } Z \text{ and at most } k-2 \text{ quads} \\ \text{and each quad has degree at least 3} \end{array} \right\} := LOW.$$

In Section 4.2 we describe a way to compute a lower bound for on t quads in time $\mathcal{O}\left(3^k t + 2^k t^2\right)$ if the distance function d is given. But first we will show how we obtained a "good" subdivision into quads.

4.1 Obtaining good Subdivisions

Our goal is to find a quadtree subdivision of the project area, such that the number of quads is "small" and the distances between the quads reflect the real distances as good as possible. As the final running time will be quadratic in the number of quads, the subdivision has to be chosen in a way such that the number of quads is small and still will give a good lower bound. First we note that it is not feasible to split the project area in a regular fashion, as either the number of quads gets to large or the lower bound obtained is bad.

Starting from a single quad covering the project area a heuristic is applied to decide which quads should be split into four new quads.

Algorithm 1 LOWTEST

```
1: function LOWTEST (Z, S, d(.))
2:     T ← the MST of the complete weighted graph (Z ∪ S, d)
3:     while not all S in T have degree ≥ 3 do          ▷ at most 30 times
4:         T ← the next largest spanning tree           ▷ Gabow's algorithm[5]
5:     return w(T)
```

Our main observation is that if there is for a multi-set S of quads with $|S| \leq k-2$ a tree T on $Z \cup S$ where each vertex in S has degree at least 3 then $LOW \leq w(T)$. The reason why S is a multi-set is because in one quad there could be more than one Steiner point. For a multi-set S of quads and the terminals Z the Algorithm 1 gives a lower bound on the minimum weight of such a tree by enumerating spanning trees. This algorithm is then used to cut branches in the branch and bound algorithm we will describe now.

Our general approach is to split the quads of all tuples for which Algorithm 1 returns a "small" value, where "small" depends on the desired lower bound given to Algorithm 2 as parameter b (e.g. 95% of our solution obtained by the heuristic).

We enumerate and test the multi-sets using the quad tree structure and a branch and bound algorithm, see the listing of Algorithm 2. The way in which in Algorithm 2 line 5 the new multi-sets are generated, ensures that no set is tested twice. If we call Algorithm 2 with a multi-set containing the root of the quad tree s times then all multi-sets of size s are tested. In line 3 and 4 we have to make sure that if the quads have been split before, while testing another multi-set, we do not recompute the same information. The computation in line 4 can be done using Dijkstra's algorithm as follows. To compute $d(Q,X)$ for a quad

Algorithm 2 TESTMSET

1: **function** TESTMSET $((S = ((Q_1, m_1), \ldots, (Q_s, m_s))), Z, d(.), b)$
$\triangleright$ bound b, multiset $S = \{\underbrace{Q_1, \ldots, Q_1}_{m_1}, \ldots, \underbrace{Q_s, \ldots, Q_s}_{m_s}\}$
2: **if** LOWTEST $(S, Z, d(.)) > b$ **then return**
3: split all quads $Q_1, \ldots Q_s$ if necessary and from sets $S_1 = \{Q_{11}, Q_{12}, Q_{13}, Q_{14}\}$,$S_2 = \{Q_{21}, \ldots\}$, etc.
4: update $d(.)$ for the new quads
5: **for all** $S_{next} = \underbrace{Q'_1 \in S_1, Q'_2 \in S_1, \ldots, Q'_{m_1} \in S_1}_{m_1}, \ldots$ **do**
6: TESTMSET $(S_{next}, Z, d(.), b)$

Q and all quads $X \in QT$ we add a new vertex v_0 and connect it by edges of weight 0 to all vertices in Q. Then we compute using Dijkstra's algorithm the distance $d_0(.)$ from v_0 to all other vertices in V. By definition we have then $d(Q,X) = \min(d_0(x)|x \in X)$. Thus for one new quad we can update $d(.)$ with one call to Dijkstra's algorithm. These Dijkstra computations are the bottleneck and give the main contribution to the overall running time. While the number of multi-sets increases drastically with the size s, we do not have to enumerate all of them as many branches are cut in Algorithm 2 line 2 at a small depth in the quad tree.

4.2 DP on quads

After computing a quadtree defining a subdivision of the project area, we use the subdivision and the distances $d(.)$ to compute a lower bound. By running Algorithm DWSET with the set of quads as parameter C we obtain our desired lower bound. Instead of showing that DWSET computes *LOW* as defined above, we will show directly that we obtain a valid lower bound. DWSET run with parameter $C = V$ returns the optimum value as this is the original Dreyfuss Wagner algorithm. We denote by $D_V(.)$ the value $D(.)$ in DWSET called with $C = V$ and for DWSET called with the set of all quads in QT as parameter C we write $D_q(.)$ for $D(.)$. Further we define $D_{2V}(.)$ and $D_{2q}(.)$ in the same way for $D_2(.)$. To show that a valid lower bound is obtained we will prove that for all $X \subset Z$ and $v \in V$ we have

$$D_V(X,v) \geq D_q(X,B(v)) \quad \text{and} \quad D_{2V}(X,v) \geq D_{2q}(X,B(v)).$$

In the remainder of this section we prove the above claim. The prove is by induction over the size of X. The basis of the induction is $D_V(\emptyset,v) = D_q(\emptyset,B(v)) = 0$ by definition of the algorithm. By Equation (1) there are two sets $X_1, X_2 \subset V$ with $D_{2V}(X,v) = D_V(X_1,v) + D_V(X_2,v)$ satisfying $X_1 \cup X_2 = X$ and $X_1 \cap X_2 = \emptyset$. Then we have in the same way by induction

$$\begin{aligned} D_{2q}(X,B(v)) &= \min_{\emptyset \subsetneq Y \subsetneq X} D_q(Y,B(v)) + D_q(X \setminus Y,B(v)) \leq D_q(X_1,B(v)) + D_q(X_2,B(v)) \\ &\leq D_V(X_1,v) + D_V(X_2,v) = D_V(X,v). \end{aligned}$$

Since for all $u,v \in V$, $d(u,v) \geq d(B(u),B(v))$ and $d(v,u) \geq d(v,B(u))$ we have by induction from Equation (2) that $D_q(X,B(v)) \leq D_V(X,v)$. Therefore we have $D_q(Z) \leq D_V(Z) = OPT$ and, thus, have obtained a valid lower bound.

5 Experimental Results

The project area on which we tested our algorithms is located in a mountainous region of the Swiss Alps (around Wägitaler See). It is a typical area of $7km \times 5km$ for which all necessary data was available. For all test inputs the underlying geological and terrain data was the same but different sets of terminals were chosen. To generate the inputs a set of 30 terminals was hand picked to guarantee reasonable inputs. From this set several sets of size 10 were chosen at random as test inputs. In the following we will present the results on ten such sets named *term1* up to *term10* in more detail. The experiments were carried out on a 64-bit intel-machine equipped with 8GB of memory and a 2.4GHz CPU. All implementations were done in C++. The graph resulting from the model had $\sim 5,600,000$ vertices and $\sim 80,000,000$ edges and 10 terminals. Applicable reduction tests (see e.g. [8]) reduced the number of edges by less than 0.1% and therefore are not mentioned explicitly. As the chosen project area is very typical for areas where such road networks are build we are confident that we can derive an indication about the performance of our approach in general.

In Set	MST-heuristic	Our Heuristic
term0	31.35s	207.480s
term1	31.14s	162.856s
term2	31.15s	202.887s
term3	30.09s	159.356s
term4	30.93s	175.169s
term5	30.56s	133.810s
term6	30.12s	167.515s
term7	30.65s	182.212s
term8	31.07s	196.931s
term9	37.11s	190.626s

Table 1: Running Times of our Heuristic and standard MST-heuristic

In Table 1 the running time of our algorithm and the running time of the standard MST-heuristic are given. The main contribution to the running time of both heuristics are shortest path computations. In Table 2 the performance of the two heuristics are compared to the respective lower bound given by the algorithm described in Section 4. Our heuristic is always at most 4% worse than the optimum which is for the application in road network design acceptable as the accuracy of the model is estimated to be only 5%. Further the improvement of our heuristic upon the standard MST-heuristic is clearly visible.

6 Conclusion

In this paper we described a new approach to compute lower bounds for the *Steiner tree problem* on a certain class of graphs. Further we reported on a heuristic used in the

Table 2: Objective Values computed by our Heuristic and the MST Heuristic.The lower bound on the optimum. Values in CHF (1CHF=0.91USD). The performance compared to the lower bound is given in brackets.

In Set	MST Heuristic	Our Heuristic	Lower Bound
term0	6,534,950 (109.83%)	6,177,913 (103.83%)	5,949,772
term1	4,635,660 (108.61%)	4,351,570 (101.95%)	4,268,105
term2	6,269,646 (111.37%)	5,775,189 (102.59%)	5,629,325
term3	6,867,598 (107.75%)	6,495,180 (101.91%)	6,373,292
term4	6,304,231 (108.00%)	5,953,613 (101.99%)	5,837,146
term5	6,525,948 (103.53%)	6,435,952 (102.10%)	6,303,114
term6	7,353,889 (109.26%)	6,997,140 (103.96%)	6,730,360
term7	7,848,047 (109.25%)	7,325,523 (101.98%)	7,182,998
term8	6,889,819 (109.53%)	6,406,460 (101.84%)	6,290,167
term9	6,721,477 (107.93%)	6,343,452 (101.86%)	6,227,226

design of road networks. We showed experimentally using the new lower bound that this heuristic works well on inputs emanating from a project area in a mountainous region of the Swiss Alps. We are confident that the heuristic will perform similarly on inputs from similar areas. In future experiments it will be interesting to work on data from other regions. Further research could work on improving the underlying model by e.g. not using a regular grid but a mapping of the vertices on the project area which takes the elevation model into account.

References

[1] E. Althaus, T. Polzin, and S. V. Daneshmand. Improving linear programming approaches for the steiner tree problem. In *Experimental and efficient algorithms*, volume 2647 of *Lecture Notes in Comput. Sci.*, pages 1–14. Springer, Berlin, 2003.

[2] N. P. Chen. New algorithms for steiner tree on graphs. In *IEEE International Symposium on Circuits and Systems*, pages 1217–1219, 1983.

[3] S. E. Dreyfus and R. A. Wagner. The steiner problem in graphs. *Networks*, 1:195–207, 1971/72.

[4] S. M. Ervin and M. D. Gross. Roadlab – a constraint based laboratory for road design. *Artificial Intelligence in Engineering*, 2(4):224–234, 1987.

[5] H. N. Gabow. Two algorithms for generating weighted spanning trees in order. *SIAM J. Comput.*, 6(1):139–150, 1977.

[6] H. R. Heinimann, J. A. Stückelberger, and W. Chung. Improving automatic grid cell based road route location procedures. In Karl Stampfer, editor, *Proceedings Austro 2003*, October 5–9, 2003. Schlaegl, Austria, 2003. CD-ROM.

[7] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner tree problem*, volume 53 of *Annals of Discrete Mathematics*. North-Holland Publishing Co., Amsterdam, 1992.

[8] T. Polzin. *Algorithms for the Steiner problem in networks.* PhD thesis, Universität des Saarlandes, 2003.

[9] J. A. Stückelberger, H. R. Heinimann, and E. C. Burlet. Modelling spatial variability in the life-cycle costs of low-volume forest roads. *European Journal of Forest Research*, 125(5):377–390, 2006.

[10] J. A. Stückelberger, H. R. Heinimann, and W. Chung. Improved road network design models with the consideration of various link patterns and road design elements. *Canadian Journal of Forest Research*, 2007. (Accepted, in press).

[11] J. A. Stückelberger, H. R. Heinimann, W. Chung, and M. Ulber. Automatic road-network planning for multiple objectives. In Woodam Chung and Han-Sup Han, editors, *The 29th Council on Forest Engineering*, pages 233–248, July 30 – August 2, 2006, Coeur d'Alene, ID, U.S.A., 2006.