Algorithmen & mathematische Modellierung

Forschungsschwerpunkt

Algorithmen und mathematische Modellierung

TU Graz
Graz University of Technology

Combinatorial algorithms for inverse absolute and vertex 1-center location problems on trees

Behrooz Alizadeh and Rainer Ernst Burkard

Project Area(s):
Effizient lösbare kombinatorische Optimierungsprobleme

Institut für Optimierung und Diskrete Mathematik (Math B)

Report 2009-9, April 2009

Combinatorial algorithms for inverse absolute and vertex 1-center location problems on trees

Behrooz Alizadeh*† Rainer E. Burkard *

April 28, 2009

Abstract. In an inverse network absolute (or vertex) 1-center location problem the parameters of a given network, like edge lengths or vertex weights, have to be modified at minimum total cost such that a prespecified vertex s becomes an absolute (or a vertex) 1-center of the network. In this article, the inverse absolute and vertex 1-center location problem on trees with $n+1$ vertices is considered where the edge lengths can be changed within certain bounds. For solving these problems a fast method is developed for reducing the height of one tree and increasing the height of a second tree under minimum cost until the height of both trees is equal. Using this result a combinatorial $O(n^2)$ time algorithm is stated for the inverse *absolute* 1-center location problem in which no topology change occurs. If topology changes are allowed, an $O(n^2 r)$ time algorithm solves the problem where r, $r < n$, is the compressed depth of the tree network T rooted in s. Finally, the inverse *vertex* 1-center problem with edge length modifications is solved on T. If all edge lengths remain positive, a new approach yields the improved $O(n^2)$ time complexity. In the general case one gets the improved $O(n^2 r_v)$ time complexity where the parameter r_v is bounded by $\lceil n/2 \rceil$.

Keywords: network center location, inverse optimization, combinatorial optimization,

1 Introduction

Location problems are fundamental optimization models in operations research and play a considerable role in practice and theory. These problems are concerned with determining the optimal locations of one or more new facilities in network systems or in

*Institute of Optimization and Discrete Mathematics, Graz University of Technology, Steyrergasse 30, A-8010 Graz, Austria. {alizadeh,burkard}@opt.math.tugraz.at

†Department of Applied Mathematics, Faculty of Basic Sciences for Engineering, Sahand University of Technology, Tabriz, Iran. alizadeh@sut.ac.ir

Corresponding author: Rainer E. Burkard, Tel. +43(316) 8735350, Fax. +43(316) 8735369

The first author acknowledges financial support by the NAWI-Project under the grant F-NW-MATH-05. This research has also been supported by the Austrian Science Fund (FWF) Project P18918-N18.

space in order to fulfill the demands of customers. See the books of Daskin [6], Drezner and Hamacher [7], Francis, McGinnis and White [8], Love, Morris and Wesolowsky [18], and Mirchandani and Francis [21] for detailed surveys on location problems.

The 1-center location problem, or simply center location problem, is the special case where one center is to be located such that the maximum weighted distance to the given points becomes minimum. Such problems occur when the best location of an emergency service, a hospital, a fire station, a police office, a bank branch or another facility center has to be found. One of the most important components of a center location model is the *classical network* 1*-center location problem* which is stated in the following way. Let a connected graph $G = (V(G), E(G))$ with vertex set $V(G)$, $|V(G)| = n$, and edge set $E(G)$, $|E(G)| = m$, be given. Every edge $e \in E(G)$ has a positive length $\ell(e)$. Moreover, for any vertex $v \in V(G)$, let $w(v)$ be a nonnegative vertex weight. We say that point p lies in G, $p \in G$, if p coincides with a vertex or lies on an edge of G. In the classical network 1-center location problem, the task is to find a point $p \in G$ such that the maximum weighted distance from any vertex $v \in V(G)$ to point p becomes minimum, or,

$$\begin{aligned} &\text{minimize} \quad \max_{v \in V(G)} w(v) d_\ell(v, p) \\ &\text{subject to} \quad p \in G, \end{aligned} \tag{1}$$

where $d_\ell(v, p)$ denotes the shortest path distance from v to p. A point p^* which solves problem (1), is said to be an *absolute* 1*-center location*. If in problem (1) point p is restricted to be a vertex (i.e., $p \in V(G)$), then we say that the optimal solution p^* is a *vertex* 1*-center location* of G.

Kariv and Hakimi [16] designed an $O(mn \log n)$ algorithm for finding an absolute 1-center of a weighted network and an $O(mn + n^2 \log n)$ algorithm for finding an absolute 1-center of an unweighted network, provided that the distance matrix of the network is available. Moreover, they proposed an $O(n \log n)$ time algorithm for the absolute (or vertex) 1-center location problem on weighted trees. Later, in 1983, Megiddo [20] showed that the weighted 1-center of a tree can be obtained in $O(n)$ time, since the objective function of the problem is convex on every simple path of the tree. For the unweighted case an efficient $O(n)$ time algorithm was developed by Handler [12].

Inverse optimization problems, particularly *inverse location problems* have found a significant interest in recent years. Since we usually are confronted with the situation that facilities already exist and cannot be relocated, inverse location problems play an important role in practice. Given a feasible solution for a location problem, the inverse location problem is concerned with modifying parameters of the original problem at minimum total cost within certain modification bounds such that the given feasible solution becomes an optimal solution with respect to the new parameter values.

A detailed survey on inverse optimization problems has been compiled by Heuberger [14]. In the context of location problems Cai, Yang and Zhang [5] proved that the inverse 1-center location problem with edge length modification on general unweighted directed graphs is $\mathcal{NP}$-hard, while the underlying center location problem is solvable in polynomial time. In 2004, Burkard, Pleschiutschnig and Zhang [3] considered inverse p-median problems and showed that discrete inverse p-median location problems can be

solved in polynomial time, when p is fixed and not an input parameter. They proposed a greedy-like $O(n \log n)$ time algorithm for the inverse 1-median problem with vertex weight modifications on tree networks. Hatzl [13] as well as Galavii [9] showed later that this problem can actually be solved in $O(n)$ time. Moreover, Burkard et al. [3] proved that the inverse 1-median problem on the plane under Manhattan (or Chebyshev) norm can be solved in $O(n \log n)$ time. Later the same authors [4] investigated the inverse 1-median problem with vertex weight modification and unit cost on a cycle. They showed that this problem can be solved in $O(n^2)$ time by using methods from computational geometry. In 2007, Gassner [10] suggested an efficient $O(n \log n)$ time solution method for the inverse 1-maxian problem with edge length modifications on tree networks. The inverse Fermat-Weber problem was studied by Burkard, Galavii and Gassner [2]. The authors derived a combinatorial approach which solves the problem in $O(n \log n)$ time for unit cost and under the assumption that the prespecified point that should become a 1-median does not coincide with a given point in the plane. Galavii [9] showed in his Ph.D. thesis that the 1-median on a path with pos/neg weights lies in one of the vertices with positive weights or lies in one of the end points of the path. This property allows to solve the inverse 1-median problem on a path with negative weights in $O(n)$ time. Gassner [11] considered an inverse version of the convex ordered median problem and showed that this problem is $\mathcal{NP}$-hard on general graphs, even on trees. Further, it was shown that the problem remains $\mathcal{NP}$-hard for unit weights or if the underlying problem is a k-centrum problem (but not, if both of these conditions hold). The inverse unit-weight k-centrum problem with unit cost coefficients on a tree can be solved in $O(n^3k^2)$ time. Recently, Yang and Zhang [23] proposed an $O(n^2 \log n)$ time solution method for the inverse vertex center problem on a tree provided that the modified edge lengths always remain positive. Dropping this condition, they mention that the general problem can be solved in $O(n^3 \log n)$ time.

This article develops novel combinatorial solution methods for the inverse absolute (or vertex) 1-center location problem where the edge lengths of a given tree network are modified at minimum total cost with respect to modification bounds such that a prespecified vertex s becomes an absolute (or a vertex) 1-center. The article is organized as follows: In the next section, we state the inverse absolute (or vertex) 1-center location problem on tree networks and show that the problem can be formulated as a nonlinear semi-infinite (or nonlinear) optimization model. By recalling basic properties from the classical absolute (or vertex) 1-center location problem we discuss the main ideas of purely combinatorial solution algorithms for solving the inverse absolute (or vertex) 1-center location problem on unweighted tree networks. In Section 3 the tree height reduction problem is considered where the height of a given rooted tree is to be reduced at minimum cost by a prespecified amount. The solution method for this problem is applied in Section 4 to develop an exact algorithm for balancing the heights of two rooted trees by increasing the height of one tree and decreasing the height of the second tree under minimum cost. The latter algorithm is used in Subsection 5.1 to solve the inverse absolute 1-center location problem in $O(n^2)$ time where we assume that no topology change is admitted. Dropping this condition, we develop an exact algorithm for the general case with $O(n^2r)$ time complexity in Subsection 5.2 where r, $r < n$, is the compressed depth of the given tree rooted in the vertex s. Finally, in Section 6,

the height balancing algorithm is again used in a novel solution method for the inverse vertex 1-center location problem on a tree. It yields improved time complexities for both cases, when the modified edge lengths remain positive and the general case.

2 Problem statement and main solution ideas

2.1 Problem statement

Let an undirected tree network $T = (V(T), E(T))$ with vertex set $V(T)$, $|V(T)| = n+1$, and edge set $E(T)$ be given such that every edge $e \in E(T)$ has a positive length $\ell(e)$. We assign a nonnegative weight $w(v)$ to every vertex $v \in V(T)$. Let s is a prespecified vertex of T. We want to modify the edge lengths at minimum total cost such that s becomes the absolute (or vertex) 1-center. Suppose that we incur nonnegative cost $c^+(e)$, if $\ell(e)$ is increased by one unit and we incur nonnegative cost $c^-(e)$, if $\ell(e)$ is reduced by one unit. Moreover, we assume that it is not possible to increase and reduce the edge lengths arbitrarily. Namely, every edge length $\ell(e)$ can only be changed between a lower bound $\ell_{low}(e) \geq 0$ and an upper bound $\ell_{upp}(e)$. Therefore, we can state the *inverse absolute (or vertex) 1-center location problem* on the given tree network T as follows:

Modify the edge lengths $\ell(e)$, $e \in E(T)$, to $\tilde{\ell}(e)$ such that the following three statements hold:

(i) The vertex s becomes an absolute (or a vertex) 1-center of T with respect to $\tilde{\ell}$, i.e., for all $p \in T$ (or $p \in V(T)$),

$$\max_{v \in V(T)} w(v) d_{\tilde{\ell}}(v, s) \leq \max_{v \in V(T)} w(v) d_{\tilde{\ell}}(v, p).$$

(ii) The linear cost function

$$\sum_{e \in E(T)} (c^+(e)x(e) + c^-(e)y(e))$$

becomes minimum, where $x(e)$ and $y(e)$ are the amounts by which the edge length $\ell(e)$ is increased and reduced, respectively.

(iii) The new edge lengths lie within given modification bounds

$$\ell_{low}(e) \leq \tilde{\ell}(e) \leq \ell_{upp}(e) \qquad \text{for all } e \in E(T).$$

Hence, based on the problem statement mentioned above, the inverse absolute (or vertex) 1-center location problem on the tree network T can be formulated as the following *nonlinear semi-infinite (or nonlinear) optimization model*:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{e\in E(T)} (c^+(e)x(e) + c^-(e)y(e)) \\
\text{subject to} \quad & \max_{v\in V(T)} w(v)d_{\tilde{\ell}}(v,s) \le \max_{v\in V(T)} w(v)d_{\tilde{\ell}}(v,p) \quad \text{for all } p \in T \text{ (or } p \in V(T)), \\
& \tilde{\ell}(e) = \ell(e) + x(e) - y(e) \quad \text{for all } e \in E(T), \qquad (2) \\
& x(e) \le \ell^+(e) \quad \text{for all } e \in E(T), \\
& y(e) \le \ell^-(e) \quad \text{for all } e \in E(T), \\
& x(e), y(e) \ge 0 \quad \text{for all } e \in E(T),
\end{aligned}
$$

where $\ell^+(e) = \ell_{upp}(e) - \ell(e)$ and $\ell^-(e) = \ell(e) - \ell_{low}(e)$ are the maximum feasible amounts by which $\ell(e)$ can be increased and reduced, respectively. Every feasible solution (x, y) with $x = \{x(e) : \ e \in E(T)\}$ and $y = \{y(e) : \ e \in E(T)\}$ is also called a *feasible modification* of the inverse absolute (or vertex) 1-center location problem.

2.2 Main solution ideas based on optimality criteria for the classical center location problems

The solution methods for the inverse 1-center problems are based on optimality criteria for 1-center problems. Handler [12] showed:

Theorem 2.1 *(midpoint-property)*
In an unweighted tree network the midpoint of a longest path is an absolute 1*-center. The closest vertex to the absolute* 1*-center is a vertex* 1*-center of the given network.*

Moreover, Handler [12] proved the following Lemma:

Lemma 2.2 *The absolute* 1*-center of an unweighted tree network is unique.*

Lemma 2.2 means that the unique absolute 1-center of an unweighted tree is the midpoint of all longest paths. Therefore, the *main idea* for solving the inverse absolute 1-center location problem on a tree network T is the following: the given tree T is split into two subtrees L and R where L is given by a longest path from s to a leaf together with all its adjacent vertices, whereas the subtree R rooted in s contains all other vertices (see Fig.1). If the heights of L and R are not equal, then the length of some edges in L have to be reduced and the length of some edges in R has to be increased. The edge lengths are modified at minimum total cost with respect to the given modification bounds until the prespecified vertex s fulfills the midpoint-property on T.

The splitting of T into two subtrees L and R is done in the following way. Let $\deg(s)$ be the degree of the prespecified vertex s. If $\deg(s) = 1$, then we set

$$L = T \ , \ \ R = s. \qquad (3)$$

Otherwise we partition T into nontrivial subtrees T_1, T_2, ... , $T_{\deg(s)}$ such that

$$\bigcup_{i=1}^{\deg(s)} T_i = T \quad \text{and} \quad T_i \cap T_j = s \quad \text{for } 1 \leq i, j \leq \deg(s),\ i \neq j.$$

Let P_{uv} denote the unique path between two arbitrary vertices $u, v \in V(T)$ and let $\ell(P_{uv})$ be the length of P_{uv} with respect to ℓ. Without loss of generality a longest path P_{sz^*} from s to the leaves of T is contained in T_1. Then we define

$$L = T_1 \quad , \quad R = \bigcup_{i=2}^{\deg(s)} T_i. \tag{4}$$

Figure 1 illustrates the partitioning of the tree T into subtrees L and R for the case $\deg(s) \geq 2$.

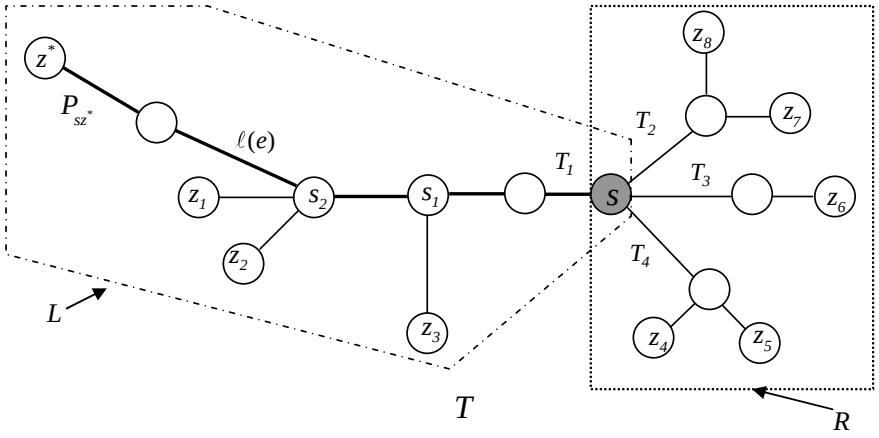

Figure 1: Illustration for partitioning the given tree network T into subtrees L and R where $\deg(s) \geq 2$. The path P_{sz^*} is a longest path from s to the leaves of T

If subtree L is not a path let s_1 denote the nearest vertex to s in the subtree L with $\deg(s_1) \geq 3$. We say that an *essential topology change* occurs if the length $\ell(P_{ss_1})$ is reduced to zero, i.e., if vertex s_1 coincides with vertex s in the reduced tree. In this case some vertices of L move to the right tree R. Thus we get new left and right trees. For example, consider the tree T given in Figure 1. After reducing the edge length $\ell(P_{ss_1})$ to 0, the leaf z_3 moves in the reduced tree T^1 from the left tree L to the right tree R^1 , see Figure 2.

Note that an essential topology change can only occur if $\ell^-(e) = \ell(e)$ for all $e \in E(P_{ss_1})$. Moreover, observe that an essential topology change never occurs if L is a path.

Now let us turn to the inverse *vertex* 1-center location problem. Our solution approach for this problem relies on the following ideas:

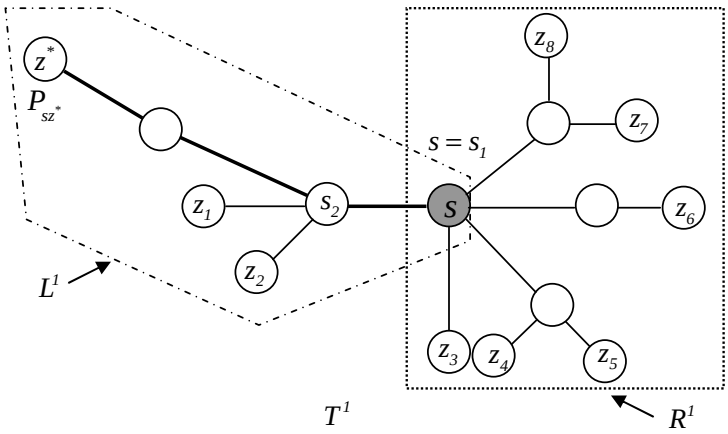

Figure 2: Tree T^1 obtained from tree T in Figure 1 by reducing the length of path P_{ss_1} to 0. The leaf z_3 which lies in T in the left tree L moves after the reduction to the right tree R^1 of T^1.

Let $Z(T)$ denote the set of leaves of T. Consider a longest path P_{sz^*}, $z^* \in Z(T)$, from the prespecified vertex s to the leaves. Let $a(s)$ be the unique adjacent vertex to s on this path P_{sz^*} and let e_s be the unique edge with end points s and $a(s)$. If we delete edge e_s, then the tree network T is split into two disjoint subtrees $\hat{L}$ and $\hat{R}$ with $a(s) \in V(\hat{L})$ and $s \in V(\hat{R})$, i.e.,

$$\hat{L} \cup \hat{R} = T - e_s \quad , \quad \hat{L} \cap \hat{R} = \emptyset, \tag{5}$$

and

$$a(s) \in V(\hat{L}) \quad , \quad s \in V(\hat{R}). \tag{6}$$

We root subtree $\hat{L}$ in vertex $a(s)$ and subtree $\hat{R}$ in vertex s. The proposed solution algorithms for solving the inverse vertex 1-center location problem are based on the following property, the so-called *optimality-inequality*, see Yang and Zhang [23]:

Theorem 2.3 *(optimality-inequality)*
Given a tree network T, a vertex s is a vertex 1-center location if and only if the inequality

$$\ell(P_{sz^*}) \leq \max\{\ell(P_{a(s)z}) : z \in Z(\hat{R})\}$$

holds where $Z(\hat{R})$ is the set of leaves of the rooted subtree $\hat{R}$.

Theorem 2.3 leads to the following *main idea* for solving the inverse vertex 1-center problem: split T into two rooted subtrees $\hat{L}$ and $\hat{R}$. If the optimality-inequality is not satisfied, then reduce the edge lengths on $\hat{L}$ and increase the edge lengths on $\hat{R}$ at minimum total cost with respect to the given modification bounds such that the maximum distance from $a(s)$ to the leaves $z \in Z(\hat{L})$ becomes equal to the maximum distance from s to the leaves $z \in Z(\hat{R})$.

In the next section, we discuss the problem of reducing the height of a tree. In Sections 5 and 6 we apply this to the subtrees L and $\hat{L}$, respectively.

3 Tree height reduction problem

Let $T = (V(T), E(T))$ be a tree with vertex set $V(T)$ and edge set $E(T)$ which is rooted in a vertex s. Every edge $e \in E(T)$ has a positive length $\ell(e)$. Reducing this length incurs nonnegative cost $c(e)$ per unit. Furthermore, let $h_\ell(T)$ denote the height of T which is equal to the length of a longest path from the root to the leaves of T with respect to the edge lengths ℓ. The *tree height reduction problem* is concerned with reducing $h_\ell(T)$ by a given amount δh, the so-called *reduction argument*, at minimum total cost. Suppose that $\ell^-(e)$ is the maximum feasible amount by which the length $\ell(e)$ can be reduced. In the tree height reduction problem on T, the goal is to reduce the edge lengths $\ell(e)$ by amounts $y(e)$ with respect to the given bounds $0 \leq y(e) \leq \ell^-(e)$ for all $e \in E(T)$ such that the total cost

$$\sum_{e \in E(T)} c(e)y(e)$$

becomes minimum while the height of T is reduced by the amount δh.

In the following we are going to present a solution method for the tree height reduction problem which is an extension of the solution approach of Zhang, Liu and Ma [24] developed for reverse center location problems. The method relies on a sequence of minimum $s - t$ cuts in a corresponding tree-like network.

Given the rooted tree T, we introduce a new vertex t and connect it to every leaf $z \in Z(T)$. Then we define

$$V(N) = V(T) \cup \{t\} \quad \text{and} \quad E(N) = E(T) \cup \{e = zt : z \in Z(T)\}.$$

Moreover, let

$$\ell_N(e) = \begin{cases} \ell^-(e) & \text{if } e \in E(T), \\ h_\ell(T) - d_\ell(s, z) & \text{if } e = zt,\ z \in Z(T), \end{cases} \tag{7}$$

$$c_N(e) = \begin{cases} c(e) & \text{if } e \in E(T), \ell_N(e) > 0, \\ 0 & \text{if } e = zt,\ z \in Z(T),\ \ell_N(e) > 0, \\ M & \text{otherwise,} \end{cases} \tag{8}$$

where M is a very big value. The network $N = (V(N), E(N), \ell_N, c_N)$ is called the *corresponding tree-like network* of T with vertex set $V(N)$, edge set $E(N)$ and edge lengths ℓ_N. All edges are directed from s to t and have capacities $c_N(e)$, $e \in E(N)$ (see Figure 3). Furthermore, define the paths

$$P_z = P_{sz} \cup \{zt\} \quad \text{for all} \quad z \in Z(T),$$

on network N where P_{sz} is the unique path from s to the leaf z on tree T. Thus we get the following lemma:

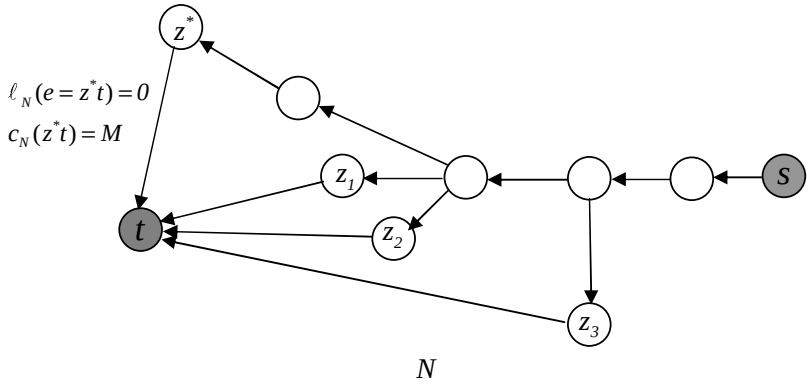

Figure 3: N is the corresponding tree-like network of the subtree L shown in Figure 1.

Lemma 3.1 *There exists a one-to-one correspondence between feasible solutions of the following two problems (a) and (b) with the same objective function values:*
Problem (a): reduce the height $h_\ell(T)$ of tree T by the amount δh
Problem (b): reduce all path lengths $\ell_N(P_z)$, $z \in Z(T)$, by the amount δh

Now we show that by finding a minimum $s-t$ cut on network N one can reduce all the path lengths $\ell_N(P_z)$, $z \in Z(T)$, and consequently according to Lemma 3.1, the height of the given tree T at minimum cost by changing the lengths of those edges which are contained in the minimum $s-t$ cut of N. Note that in network N we are allowed to reduce the length of every edge to zero.

Let $K = (X, Y)$ with $s \in X,\ t \in Y$ be a minimum $s-t$ cut in N and $E(K) = \{e = uv \in E(N) :\ u \in X, v \in Y\}$. The capacity of K is computed by

$$c(K) = \sum_{e \in E(K)} c_N(e).$$

If $c(K) \geq M$, then this means that there exists at least one path P_z, $z \in Z(T)$, with $\ell_N(e) = 0$ for all $e \in E(P_z)$. In this case the length of P_z cannot be reduced any more and thus the tree height reduction problem is infeasible. Otherwise, let

$$\delta(K) = \min\{\ell_N(e) :\ e \in E(K)\}.$$

The parameter $\delta(K)$ is the maximum amount by which we can reduce the length of all edges in $E(K)$. The following lemma is crucial for the correctness of our solution method for the tree height reduction problem.

Proposition 3.2 *Let K be a minimum $s-t$ cut in N with $c(K) < M$ and let $0 < \delta \leq \delta(K)$. If we reduce the edge lengths of N by amounts*

$$y^*(e) = \begin{cases} \delta & \text{if } e \in E(K), \\ 0 & \text{otherwise}, \end{cases}$$

then the height of tree T is reduced by the amount δ at minimum total cost.

Proof. Let K be a minimum $s-t$ cut with $c(K) < M$. This implies that all edges $e \in E(K)$ have a positive length. Therefore $\delta(K) > 0$. The cut contains in particular an edge of every longest path P_{sz} in T, since $c_N(zt) = M$. Therefore all longest paths in T are reduced by δ. If P_{sz} is not a longest path in T, the cut contains the edge zt, since $c_N(zt) = 0$ and the cut has minimum cost. The definition of $\delta(K)$ implies that after the reduction all reduced longest paths have a length at least as long as a second longest path P_{sz} in T before the reduction. Thus the height of T has been reduced by δ at minimum cost. ■

Let us now assume that we have to reduce the height of tree T by the amount δh. We determine a first minimum $s-t$ cut K_1 in network N with corresponding value $\delta(K_1)$. If $\delta h \leq \delta(K_1)$, the height can be reduced by using the cut K_1. Otherwise, choose in the first step $\delta = \delta(K_1)$, reduce the edge lengths in N by

$$\ell_N(e) = \begin{cases} \ell_N(e) - \delta(K_1) & \text{if } e \in E(K_1), \\ \ell_N(e) & \text{otherwise.} \end{cases}$$

and then update

$$\delta h = \delta h - \delta(K_1),$$

$$c_N(e) = \begin{cases} c_N(e) & \text{if } \ell_N(e) > 0, \\ M & \text{otherwise.} \end{cases}$$

The resulting network is again called $N = (V(N), E(N), \ell_N, c_N)$. Observe that after such a transformation at least one edge $e \in E(N)$ will have the new length $\ell_N(e) = 0$. Hence, by iterating this process we find a sequence of minimum $s-t$ cuts K_i, $i = 1, ...q$, with capacities $c(K_i)$ which lead successively to the reduction of the underlying tree T by the amounts $\delta(K_i), i = 1, ...q$. The process terminates if either $\delta(K_q) \geq \delta h$ or if $\delta(K_q) \geq M$. In the latter case the problem is infeasible.

In order to solve the tree height reduction problem we generate at most n minimum $s-t$ cuts in N. Every minimum $s-t$ cut in network N can be found in $O(n)$ time, since $N - \{t\}$ is an arborescence (see e.g., Hochbaum [15] or Vygen [22]). Also the updating of network N takes at most $O(n)$ time. Hence we get

Theorem 3.3 *The tree height reduction problem can be solved in $O(n^2)$ time where n is the number of edges on the given tree.*

In the next section the tree height reduction problem will be used as a subproblem for balancing the heights of two rooted trees.

4 Height balancing of two rooted trees

Let T' and T'' be two trees rooted in vertices s' and s'', respectively. The problem to balance the height of T' and T'' consists in modifying (increasing or reducing) the edge lengths $\ell(e)$ for all $e \in E(T'), E(T'')$ at minimum toal cost so that the height of T' and the height of T'' are equal with respect to the new edge lengths. Let $c^+(e)$ and $c^-(e)$ be

the cost coefficients for increasing and reducing $\ell(e)$, respectively. Moreover, suppose that $\ell^+(e)$ and $\ell^-(e)$ are the maximum amounts by which the length $\ell(e)$ is allowed to be increased and reduced, respectively. Therefore, in the height balancing of T' and T'' we either increase $\ell(e)$ by an amount $x(e)$ or reduce $\ell(e)$ by a amount $y(e)$ within the given bounds $0 \leq x(e) \leq \ell^+(e)$ and $0 \leq y(e) \leq \ell^-(e)$ for all $e \in E(T'), E(T'')$ such that the total cost

$$\sum_{e \in E(T' \cup T'')} (c^+(e)x(e) + c^-(e)y(e))$$

becomes minimum and $h_{\tilde{\ell}}(T') = h_{\tilde{\ell}}(T'')$ under the new edge lengths $\tilde{\ell}$.

Note that if the equality $h_\ell(T') = h_\ell(T'')$ holds for the original edge lengths ℓ, then the two rooted trees T' and T'' are balanced. Otherwise we have to change the edge lengths on T' and T''. Without loss of generality assume that $h_\ell(T') \geq h_\ell(T'')$. Basic for the solution method are the following lemmas.

Lemma 4.1 *In order to balance the heights of T' and T'', it is sufficient to reduce the height of T' and to increase the height of T'' at minimum total cost subject to the modification bounds, until T' and T'' have equal heights.*

Proof. Since the cost for increasing or decreasing an edge are positive, every increase of an edge in T' or decrease of an edge in T'' would cause avoidable cost. ■

Thus it is sufficient to reduce edge lengths in T' and to increase edge lengths in T''. In particular we get the following lemma:

Lemma 4.2 *If (x^*, y^*) is an optimal edge length modification needed for the height balancing of T' and T'', then there exists exactly one path $P_{s''z}$ from s'' to a leaf $z \in Z(T'')$ that contains all the edges $e \in E(T'')$ with $x^*(e) > 0$.*

Proof. Since for every $e \in E(T'')$ the cost coefficients $c^+(e)$ are positive, increasing the edge lengths on more than one path $P_{s''z},\ z \in Z(T'')$, implies additional cost. ■

From Lemma 4.1 and Lemma 4.2 we conclude that for balancing the heights of T' and T'' it is required to reduce the height of the tree T' and to find one path $P_{s''z^0}$ from s'' to a leaf z^0 in T'', the so-called *best candidate path*, whose length is increased such that the heights of T' and T'' become equal.

The minimum height of tree T' can be computed in the following way. Let $Z(T')$ be the set of leaves of T'. Every path $P_{s'z}$ from s' to a leaf $z \in Z(T')$ can be reduced at most by the amount

$$\ell^-(P_{s'z}) = \sum_{e \in E(P_{s'z})} \ell^-(e).$$

Therefore, the shortest possible height $h_{\min}$ of T' is given by

$$h_{\min} = \max_{z \in Z(T')} \{\ell(P_{s'z}) - \ell^-(P_{s'z})\}$$

This means that the tree T' can be reduced by an amount $\delta \leq \delta_{\max}$ where $\delta_{\max}$ is given by

$$\delta_{\max} = h_\ell(T') - h_{\min}. \tag{9}$$

On the other hand, Lemma 4.1 implies that it is not necessary to reduce the height of the tree T' below the height of the tree T''. Thus we define

$$\Delta(T', T'') = \min(\delta_{\max}, h_\ell(T') - h_\ell(T'')). \tag{10}$$

We reduce the height of tree T' by the amount $\Delta(T', T'')$ by generating q minimum $s-t$ cuts $K_1, ..., K_q$. Obviously, we have

$$\Delta(T', T'') \leq \sum_{j=1}^{q} \delta(K_j).$$

Recall that for every $j = 1, ..., q$, the corresponding capacity $c(K_j)$ of the minimum $s-t$ cut K_j is the minimum cost for reducing the height of T' by the amount $\delta(K_j)$. The problem to equalize the height of T' with the length $\ell(P_{s''z})$ of a path from s'' to a leaf z in the tree T'' under minimum cost C_z can be formulated as the following linear optimization problem:

$$\begin{aligned}
(\mathrm{LP_z}) \qquad \text{minimize} \quad & \sum_{j=1}^{q} c(K_j)\tau(K_j) + \sum_{e \in E(P_{s''z})} c^+(e)x(e) \\
\text{subject to} \quad & \sum_{j=1}^{q} \tau(K_j) + \sum_{e \in E(P_{s''z})} x(e) = h_\ell(T') - \ell(P_{s''z}), \\
& 0 \leq \tau(K_j) \leq \delta(K_j) \quad \text{for } j = 1, \cdots, q, \\
& 0 \leq x(e) \leq \ell^+(e) \qquad \text{for all } e \in E(P_{s''z}),
\end{aligned}$$

where $\tau(K_j)$ is the amount by which all the edges of the cut set $E(K_j)$ are reduced.

Without loss of generality we may assume that $\ell^+(e) > 0$ for every $e \in E(P_{s''z})$. If there exists an edge e with $\ell^+(e) = 0$, then its corresponding variable is removed from problem $(\mathrm{LP_z})$. On the other hand we know that the cut values $\delta(K_j),\ j = 1, \cdots, q$, are positive. This allows us to reduce $(\mathrm{LP_z})$ to a continuous knapsack problem by introducing the following new variables

$$\xi^-(K_j) = \frac{\tau(K_j)}{\delta(K_j)} \qquad \text{for } j = 1, ..., q, \tag{11}$$

$$\xi^+(e) = \frac{x(e)}{\ell^+(e)} \qquad \text{for all } e \in E(P_{s''z}), \tag{12}$$

$$\begin{aligned}
\tilde{c}^-(K_j) &= \delta(K_j)c(K_j) \qquad \text{for } j = 1, ..., q, \\
\tilde{c}^+(e) &= \ell^+(e)c^+(e) \qquad \text{for all } e \in E(P_{s''z}),
\end{aligned}$$

Thus we get

$$
\begin{aligned}
(\mathrm{KP_z}) \qquad \text{minimize} \quad & \sum_{j=1}^{q} \tilde{c}^{-}(K_j)\xi^{-}(K_j) + \sum_{e\in E(P_{s''z})} \tilde{c}^{+}(e)\xi^{+}(e) \\
\text{subject to} \quad & \sum_{j=1}^{q} \delta(K_j)\xi^{-}(K_j) + \sum_{e\in E(P_{s''z})} \ell^{+}(e)\xi^{+}(e) = h_\ell(T') - \ell(P_{s''z}), \\
& 0 \le \xi^{-}(K_j) \le 1 \qquad \text{for } j = 1, \cdots, q, \\
& 0 \le \xi^{+}(e) \le 1 \qquad \text{for all } e \in E(P_{s''z}).
\end{aligned}
$$

If the inequality

$$
\sum_{j=1}^{q} \delta(K_j) + \sum_{e\in E(P_{s''z})} \ell^{+}(e) \ge h_\ell(T') - \ell(P_{s''z}) \tag{13}
$$

is satisfied, then problem ($\mathrm{KP_z}$) is feasible. Its optimal objective function value C_z can be obtained in linear time time by using the solution algorithm of Balas and Zemel, see e.g., [1], [17], [19]. Through the transformations (11) and (12) an optimal solution (x, τ) of the problem ($\mathrm{LP_z}$) with the same objective value C_z is obtained. Otherwise, if inequality (13) does not hold, then problem ($\mathrm{KP_z}$) has no solution. In this case we set $C_z = +\infty$. The best candidate path $P_{s''z^0}$ is obtained by

$$
z^0 \in \operatorname{argmin}\{C_z : z \in Z(T'')\}.
$$

If $C_{z^0} = +\infty$, then the given problem is not feasible. Otherwise, we derive an optimal solution of the height balancing problem on trees T' and T'' from the optimal solution $(\tilde{\xi}^{+}, \tilde{\xi}^{-})$ of the continuous knapsack problem ($\mathrm{KP_{z^0}}$) as follows: For $j = 1, ..., q$ let

$$
y_j(e) = \begin{cases} \tilde{\xi}^{-}(K_j)\delta(K_j) & \text{if } e \in E(K_j) \cap E(T' \cup T''), \\ 0 & \text{otherwise.} \end{cases}
$$

Now define

$$
y^*(e) = \sum_{j=1}^{q} y_j(e) \qquad \text{for all } e \in E(T' \cup T''), \tag{14}
$$

and

$$
x^*(e) = \begin{cases} \tilde{\xi}^{+}(e)\ell^{+}(e) & \text{if } e \in E(P_{s''z^0}), \\ 0 & \text{otherwise.} \end{cases} \tag{15}
$$

Then (x^*, y^*) is an optimal solution of the height balancing problem on the trees T' and T''. The preceding considerations are summarized in Algorithm 1.

The solution algorithm has the following complexity: Let $n = |E(T')| + |E(T'')|$. The reduction argument $\delta_{\max}$ is computed in at most $O(|E(T')|)$ time, if we traverse the tree network T' in a depth-first manner. Furthermore, according to Theorem 3.3, the reduction of the height $h_\ell(T')$ is performed in $O(|E(T')|^2)$ time. On the other hand,

Algorithm 1 balances the heights of two rooted trees T' and T''.

begin
compute the maximum reduction value $\Delta(T', T'')$ according to (10);
reduce the height of tree T' by $\Delta(T', T'')$; this yields the reduction values $\delta(K_1), ..., \delta(K_q)$, the edge sets $E(K_1), ..., E(K_q)$, and the capacities $c(K_1), ..., c(K_q)$;
for every $z \in Z(T'')$ **do**
  solve the knapsack problem ($\mathrm{KP_z}$) and obtain the objective value C_z; if ($\mathrm{KP_z}$) is infeasible, set $C_z = +\infty$;
end for
choose $z^0 \in \operatorname{argmin}\{C_z : \ z \in Z(T'')\}$;
if $C_{z^0} = +\infty$ **then**
  T' and T'' can not be balanced, stop;
else
  an optimal solution (x^*, y^*) of the height balancing problem can be derived from the optimal solution $(\tilde{\xi}^+, \tilde{\xi}^-)$ of the problem ($\mathrm{KP_{z^0}}$) by (14) and (15);
end if
end

during the execution of Algorithm 1 we solve at most $|E(T'')|$ continuous knapsack problems ($\mathrm{KP_z}$), $z \in Z(T'')$. A continuous knapsack problem can be solved in $O(|E(T'')|)$ time. Moreover, an optimal solution vector (x^*, y^*) of the height balancing problem is obtained from the optimal solution of ($\mathrm{KP_{z^0}}$) in $O(n^2)$ time. Thus we conclude that the time complexity of the algorithm is bounded by $O(n^2)$. Altogether we get

Theorem 4.3 *The height balancing of two rooted trees T' and T'' can be performed in $O(n^2)$ time by Algorithm 1 where n is the number of edges in T' and T''.*

In the next sections we use the balancing of two rooted trees as a subproblem for solving the inverse absolute (and vertex) 1-center location problems.

5 Inverse absolute 1-center location problem

5.1 Inverse absolute 1-center location problem without essential topology change

In the following we propose a combinatorial algorithm for the inverse absolute 1-center location problem on the tree network T in which we assume that the given bounds for reducing the edge lengths obey the condition

$$\exists e \in E(P_{ss_1}) \quad \text{such that} \quad \ell^-(e) < \ell(e), \tag{16}$$

where P_{ss_1} is the unique path from the prespecified vertex s to the nearest vertex $s_1 \in E(L)$ with $\deg(s_1) \geq 3$ as introduced in Subsection 2.2. This condition implies that an essential topology change never occurs on the tree T. We want to modify the edge lengths of T at minimum total cost within given bounds such that s becomes the absolute 1-center under the new edge lengths.

Let $\deg(s) \geq 2$ and consider the subtrees L and R of T as explained in Subsection 2.2. We root L and R in vertex s. If

$$h_\ell(L) = h_\ell(R),$$

then s is the midpoint of all longest paths on T and therefore the wanted absolute 1-center. Otherwise we have to change the edge lengths in T. The following lemma is the basic for the solution method.

Lemma 5.1 *In order to solve the inverse absolute* 1*-center location problem without essential topology change on the given tree network T, it is sufficient to balance the heights of the subtrees L and R both rooted in the prespecified vertex s.*

Proof. The lemma is an immediate consequence of Theorem 2.1 and the fact that the topology of the tree T does not change by virtue of condition (16). ■

Based on all considerations above we can state the following algorithm (Algorithm 2).

Algorithm 2 finds an optimal solution of the inverse absolute 1-center location problem on a nontrivial tree network T without essential topology change, vertex s being the new absolute 1-center.

---

begin
if $\deg(s) = 1$ **then**
  the problem is infeasible, stop;
else
  partition T into subtrees L and R according to (4);
  root L and R in vertex s;
  execute Algorithm 1 in order to balance the heights of L and R;
end if
end

Algorithm 2 has the following time complexity: The longest paths from s to the leaves $z \in Z(T)$ are found in $O(n)$ time. Thus the partition of T into subtrees L and R takes $O(n)$ time. Based on Theorem 4.3 the heights of subtrees L and R can be balanced in $O(n^2)$ time. Therefore, Algorithm 2 runs in $O(n^2)$ overall time. Summarizing, we get

Theorem 5.2 *The inverse absolute* 1*-center location problem with edge length modification can be solved in $O(n^2)$ time on a tree network (with n edges) by Algorithm 2, provided that no essential topology change occurs in the given tree.*

5.2 The general inverse absolute 1-center location problem

In this subsection, we drop condition (16), i.e., an essential topology change may occur in the tree T during the height reduction of the left tree L. This means, after a length

reduction in the left tree L some vertices of L move to the right tree R, see Figure 1 and Figure 2.

We start our investigation with the definition of the compressed depth of T. Let $\mathcal{Z} \subseteq Z(T)$ be the set of all leaves of T with maximum distance $d_\ell(s,z)$. Consider the value $\Delta(L,R)$ which can be obtained according to (10). On every path P_{sz} let $s_1, s_2, ..., s_{i(z)}$ with $s \neq s_1$ be the sequence of vertices from s to z with the following three properties

$$\deg(s_k) \geq 3, \tag{17}$$

$$\ell(P_{ss_k}) = \ell^-(P_{ss_k}), \tag{18}$$

and

$$\ell(P_{ss_k}) \leq \Delta(L,R) \tag{19}$$

for all $k = 1, 2, ..., i(z)$.

Definition 5.3 *The* compressed depth *of T is given by*

$$r = \min\{i(z) : z \in \mathcal{Z}\} + 1.$$

Moreover, a path P_{sz}, $z \in \mathcal{Z}$, with $i(z) = r-1$ is called compressed longest path *in T.*

Observe that if $r = 1$, then the left tree L is a path or at least one of the properties $\ell(P_{ss_1}) = \ell^-(P_{ss_1})$ and $\ell(P_{ss_1}) \leq \Delta(L,R)$ does not hold, where s_1 is the nearest vertex to s with $\deg(s_1) \geq 3$ on the left tree L as introduced in Section 2. In this case an optimal solution of the problem can be found without an essential topology change on the underlying tree T. Hence, Algorithm 2 can be applied. Otherwise we consider a compressed longest path P_{sz^*} of T. Let $s_1, ..., s_{r-1}$ be the sequence of its vertices from s to s_{r-1} with properties (17), (18) and (19). Moreover, let $s_0 = s$.

For any k, $0 \leq k < r$, let T^k be the tree obtained from T by reducing the length $\ell(P_{ss_k})$ to zero. Moreover, let L^k and R^k be the two subtrees of T^k which are rooted in $s = s_i$, $i = 0, ..., k$, and are determined by (3) or (4). For example, Figure 4 illustrates the resulting tree T^2 with the two corresponding subtrees L^2 and R^2 which is obtained from the tree T in Figure 1 by reducing the length of the path P_{ss_2} to zero.

For every tree T^k, $k = 0, 1, ..., r-1$ we determine the cost $C_{s,k}$ incurred by reducing the length of path P_{ss_k} to 0 and the minimum cost $C(T^k)$ incurred by changing the edge lengths of tree T^k such that s becomes an absolute 1-center of T^k. Thus the total cost of subproblem (SP_k) is given by

$$C_k = C_{s,k} + C(T^k), \tag{20}$$

where

$$C_{s,k} = \sum_{e \in E(P_{ss_k})} c^-(e)\ell(e), \tag{21}$$

and $C(T^k)$ is the optimal value obtained by Algorithm 2 applied to tree T^k. The minimum cost C_{k_0} with

$$k_0 \in \operatorname{argmin}\{C_k : k = 0, ..., r-1\} \tag{22}$$

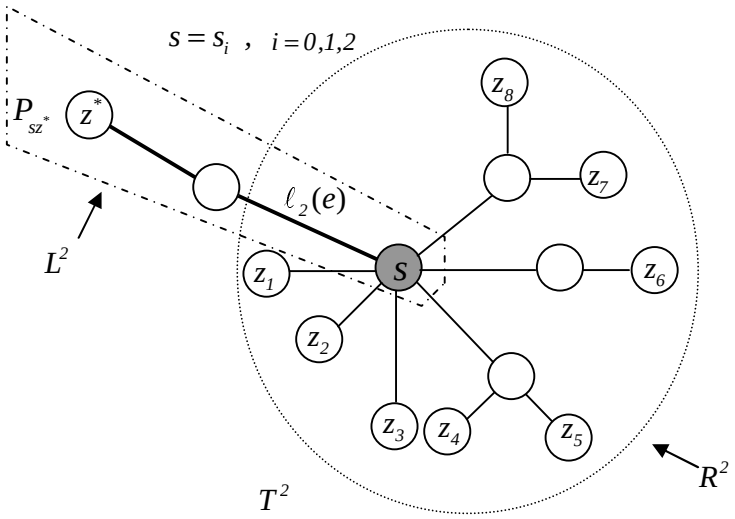

Figure 4: T^2 is obtained from the tree T of Figure 1 by reducing the length of P_{ss_2} to zero. P_{sz^*} is a compressed longest path of T.

yield the optimum objective function value of the general inverse absolute 1-center location problem on tree T. In the case that C_{k_0} is a finite value, an optimal solution of the problem is obtained by

$$y^*(e) = \begin{cases} \ell_0(e) & \text{if } e \in E(P_{ss_{k_0}}), \\ y_{k_0}(e) & \text{otherwise,} \end{cases} \tag{23}$$

$$x^*(e) = \begin{cases} 0 & \text{if } e \in E(P_{ss_{k_0}}), \\ x_{k_0}(e) & \text{otherwise.} \end{cases} \tag{24}$$

where (x_{k_0}, y_{k_0}) is the optimal solution obtained by Algorithm 2 applied to tree T^{k_0}, and $\ell_0(e)$ maintains the original length $\ell(e)$ on the underlying tree T.

The cost $C_{s,k}$ can be computed in a recursive way by

$$C_{s,k} := C_{s,k-1} + \sum_{e \in E(P_{s_{k-1}s_k})} c^-(e)\ell(e). \tag{25}$$

Moreover, if

$$\sum_{e \in E(P_{s_{k-1}s_k})} c^-(e)\ell(e) \geq \min_{0 \leq i \leq k-1} C(T^i), \tag{26}$$

we can skip the computation of C_i, $i \geq k$, since none of these problems can improve the best objective function value already found. Note that the tree T^k is obtained from T^{k-1} by reducing the length $\ell(P_{s_{k-1}s_k})$ to 0.

Further we can observe that after performance of any optimal modification the path P_{sz^*} remains as a longest path from s to the leaves. Therefore, if for a given

$k \in \{1, ..., r-1\}$ the inequality

$$\ell(P_{s_{k-1}s_k}) > h_\ell(L^{k-1}) - h_\ell(R^{k-1}) \tag{27}$$

holds with respect to the edge lengths of T^{k-1}, then an optimal solution never contains the reduction of $\ell(P_{s_{k-1}s_k})$ to 0. In this case we also skip the computation of C_i, $i \geq k$.

The previous considerations lead to the following Algorithm 3 which solves the general inverse absolute 1-center location problem on a tree T.

Algorithm 3 finds an optimal solution of the general inverse absolute 1-center location problem on a tree network T, vertex s being the new absolute 1-center.

---

begin
find the compressed depth r and a compressed longest path P_{sz^*}, $z^* \in Z(T)$;
let $T^0 = T$; obtain the optimal objective value $C_0 = C(T^0)$ of the inverse absolute 1-center problem on T^0 by executing Algorithm 2;
if $r = 1$ **or** $C_0 = 0$ **then**
  stop: the original problem has been solved;
else
  let s_k, $k = 1, ..., r-1$, be the sequence of vertices on P_{sz^*} with three properties (17), (18) and (19); set $s_0 = s$, $k = 1$ and $C_{s,0} = 0$;
  while $k < r$ **do**
    if (26) or (27) do not hold **then**
      construct the tree T^k from T^{k-1}; execute Algorithm 2 on T^k due to obtain the optimal objective value $C(T^k)$;
      compute the cost $C_{s,k}$ and then C_k according to (25) and (20), respectively;
      else set $C_i = +\infty$ for all $i = k, ..., r-1$; break while loop;
    end if
    set $k = k + 1$;
  end while
  choose the index k_0 by (22);
  if C_{k_0} is finite **then**
    an optimal solution (x^*, y^*) of the original problem can be found according to (23) and (24); **else** stop: the original problem is infeasible;
  end if
end if
end

The discussion above shows that Algorithm 3 finds an optimal solution of the general inverse absolute 1-center location problem on the tree network T if the problem is feasible. Now we are going to compute the running time of this algorithm. A compressed longest path of T is found in $O(n)$ time if T is traversed in a depth-first search manner. The total time needed to construct the trees T^k, $k = 1, ..., r-1$, is $O(n)$. Moreover, due to Theorem 5.2 we know that Algorithm 2 runs in $O(n^2)$ time. On the other hand, during execution of Algorithm 3 we recall Algorithm 2 at most r times where $r < \frac{n}{2} + 1$.

Therefore the overall time complexity of Algorithm 3 is bounded by $O(n^2 r)$. Observe that $r \leq \log(n)$ when the underlying tree T is a complete binary tree with root s. Thus we have shown

Theorem 5.4 *The general inverse absolute* 1*-center location problem with edge length modification can be solved in* $O(n^2 r)$ *time on a tree network by performing Algorithm 3 where* r *is the compressed depth of the given tree and* n *is the number of the edges.*

6 Inverse vertex 1-center location problem

This section considers the inverse vertex 1-center location problem on the given tree network T. Recently Yang and Zhang [23] investigated a solution method which solves the problem in $O(n^2 \log n)$ time provided that the modified edge lengths always remain positive, i.e., $\tilde{\ell} > 0$. Moreover, they pointed out that the general problem, i.e., in case of $\tilde{\ell} \geq 0$, can be solved in $O(n^3 \log n)$. However, a solution procedure was not discussed in this case. In the following we derive new solution methods with improved time complexities for the inverse vertex 1-center problem using the tree height balancing algorithm.

6.1 Inverse vertex 1-center location problem with positive modified edge lengths

In this subsection we investigate the inverse vertex 1-center location problem in which we assume that

$$\ell^-(e) < \ell(e) \quad \text{for all } e \in E(T). \tag{28}$$

This condition describes that during solving the problem the modified edge lengths always remain positive, i.e., $\tilde{\ell} > 0$. Let $\hat{L}$ and $\hat{R}$ be two subtrees of T which are determined according to (5) and (6) and are rooted in the vertices $a(s)$ and s, respectively. From Theorem 2.3 and according to Assertion 4 in [23], we immediately get the following lemma .

Lemma 6.1 *For solving the inverse vertex* 1*-center location problem in a given tree* T*, it is sufficient to balance the heights of the subtrees* $\hat{L}$ *and* $\hat{R}$ *provided that the condition* (28) *is satisfied.*

Therefore, an optimal solution of the problem can be obtained in the following way: We execute Algorithm 1 in order to balance the heights of $\hat{L}$ and $\hat{R}$. If these two subtrees could not be balanced, then the original problem is infeasible. Otherwise, an optimal solution of the inverse vertex 1-center location problem can be obtained from the optimal solution (x^*, y^*) of the height balancing problem together with fixed $x^*(e_s) = y^*(e_s) = 0$.

Note that the subtrees $\hat{L}$ and $\hat{R}$ can be determined in $O(n)$ time if we traverse the tree T in a depth-first search manner. On the other hand, based on Theorem 4.3 the balancing the heights of $\hat{L}$ and $\hat{R}$ is performed in $O(n^2)$ time. Thus we get

Theorem 6.2 *The inverse vertex* 1*-center location problem with positive modified edge lengths can be solved in* $O(n^2)$ *time on a tree with* n *edges.*

6.2 The general inverse vertex 1-center location problem

In the following we describe a solution method with improved time complexity for the general inverse vertex 1-center location problem on the given tree network T dropping condition (28).

We start with defining the *minimal repeat number* r_v on tree T. On every path P_{sz}, $z \in Z(T)$, let $s_1, s_2, \ldots, s_{j(z)}$ with $s \neq s_1$ denote the sequence of all vertices from s to the leaf z which satisfy the properties (18) and (19) for all $k = 1, 2, \ldots, j(z)$.

Definition 6.3 *Given a tree T, the* minimal repeat number r_v *is defined as*

$$r_v = \min\{j(z) : z \in \mathcal{Z}\} + 1.$$

Observe that if $r = 1$, then an optimal solution of the problem can be found if we apply the solution method developed in Subsection 6.1 to the tree T. Otherwise we consider a path P_{sz^*} of T with $j(z^*) = r_v - 1$. Let $s_1, s_2, \ldots, s_{r_v-1}$ be the sequence of its vertices from s to z^* with properties (18) and (19). Further, we set $s_0 = s$. Recall that T^k is the tree obtained from T by reducing the length $\ell(P_{ss_k})$, $0 \leq k < r_v$, to 0. Moreover, let $\hat{L}^k$ and $\hat{R}^k$ be two subtrees of T^k which are determined according to (5)–(6) and are rooted in $a(s_k)$ and $s = s_i$, $i = 0, ..., k$, respectively.

For every tree T^k we determine a corresponding total cost C_k according to

$$C_k = C_{s,k} + \hat{C}(T^k) \tag{29}$$

where the cost $C_{s,k}$ is incurred by reducing the length $\ell(P_{ss_k})$ to 0 and $\hat{C}(T^k)$ is the optimal value obtained by applying the solution method developed in Subsection 6.1 to T^k. If we choose the index k_0 by (22), then C_{k_0} is the minimum objective function value of the general inverse vertex 1-center location problem on the given tree network T. In the case that C_{k_0} is a finite value (i.e., the problem is feasible), then an optimal solution of the problem under investigation can be obtained by (23) and (24) where (x_{k_0}, y_{k_0}) is the optimal solution obtained by the solution method developed in Subsection 6.1 to T^{k_0} and $\ell_0(e)$ is the original length of the edge e on the underlying tree T. Note that if

$$\sum_{e \in E(P_{s_{k-1}s_k})} c^-(e)\ell(e) \geq \min_{0 \leq i \leq k-1} \hat{C}(T^i), \tag{30}$$

or

$$h_\ell(\hat{L}^{k-1}) \leq h_\ell(\hat{R}^{k-1}), \tag{31}$$

hold, then we can skip the computation of C_i, $i \geq k$, and assign immediately $C_i = +\infty$ for all $i \geq k$.

Since for solving the general inverse vertex 1-center location problem one needs to execute the solution method developed in Subsection 6.1 at most r_v times (i.e., solve the inverse vertex 1-center problem under the condition $\tilde{\ell} > 0$ on trees $T^0, ..., T^{r_v-1}$), then based on Theorem 6.2 we conclude that the overall time complexity of the solution method is bounded by $O(n^2 r_v)$ where $r_v < \frac{n}{2} + 1$. Altogether we get

Theorem 6.4 *The general inverse vertex 1-center location problem with edge length modification can be solved in $O(n^2 r_v)$ time on a tree network where r_v is the minimal repeat number of the given tree and n is the number of the edges.*

7 Concluding remarks

In this article, we investigated the inverse absolute (or vertex) 1-center location problem with variable edge lengths on tree networks and showed that the problem is formulated as a nonlinear semi-infinite (or nonlinear) programming model. We developed exact combinatorial solution algorithms for the problem in the special and general cases with fast running times.

In case that the underlying tree network is a star graph, then the inverse absolute (or vertex) 1-center location problem can be solved efficiently in $O(n)$ time in straightforward approaches.

In the extensive absolute (or extensive vertex) 1-center location problem on a network G, the demand points lie on edges of the network as well as in vertices. Therefore, a point $p^* \in G$ (or $p^* \in V(G)$) is called an extensive absolute (or extensive vertex) 1-center of G if and only if p^* minimizes the maximum shortest distance to all points $x \in G$, i.e., the point p^* is a solution of the problem

$$\min_{\substack{p\in G \\ (\text{or } p\in V(G))}} \max_{x\in G} d_\ell(p,x).$$

Note that in a tree network the absolute (or vertex) 1- center location problem is equivalent with the extensive absolute (or extensive vertex) 1-center location problem. Hence the newly developed algorithms in this article also solve the inverse extensive absolute (or extensive vertex) 1-center location problem on tree networks correctly.

Acknowledgments The authors thank Elisabeth Gassner and Ulrich Pferschy for their constructive suggestions.

References

[1] E. Balas, E. Zemel, An algorithm for large zero-one knapsack problems, *Operations Research* 28(1980), 1130–1154.

[2] R.E. Burkard, M. Galavii and E. Gassner, The inverse Fermat-Weber problem, *Technical Report 2008-14*, Graz University of Technology, Graz, 2008.

[3] R.E. Burkard, C. Pleschiutschnig and J. Zhang, Inverse median problems, *Discrete Optimization* 1(2004), 23–39.

[4] R.E. Burkard, C. Pleschiutschnig and J. Zhang, The inverse 1-median problem on a cycle, *Discrete Optimization* 5(2007), 242–253.

[5] M.C. Cai, X.G. Yang and J.Z. Zhang, The complexity analysis of the inverese center location problem, *Journal of Global Optimization* 15(1999), 213–218.

[6] M.S. Daskin, *Network and Discrete Location: Modeles, Algorithms and Applications*, John Wiley, New York, 1995.

[7] Z. Drezner and H.W. Hamacher, *Facility Location, Applications and Theory*, Springer Verlag, Berlin, 2004.

[8] R.L. Francis, L.F. McGinnis and J.A. White, *Facility Layout and Location, An Analytical Approach*, Prentice Hall, Englewood Cliffs, 1992.

[9] M. Galavii, *Inverse* 1-*Median Problems*, Ph.D. Thesis, Institute of Optimization and Discrete Mathematics, Graz University of Technology, Graz, 2008.

[10] E. Gassner, The inverse 1-maxian problem with edge length modification, *J. Combinatorial Optimization* 16(2007), 50–67.

[11] E. Gassner, An inverse approach to convex ordered median problems in trees, Technical Report 2008-16, Graz University of Technology, Graz, 2008.

[12] G.Y. Handler, Minimax location of a facility in an undirected tree graph, *Transportation Science* 7(1973), 287–293.

[13] J. Hatzl, personal communication, 2006.

[14] C. Heuberger, Inverse combinatorial optimization: A survey on problems, methods, and results, *Journal of Combinatorial Optimization* 8(2004), 329–361.

[15] D.S. Hochbaum, The pseudoflow algorithm: A new algorithm for the maximum flow problem, *Operations Research*, to appear.

[16] O. Kariv and S. L. Hakimi, An algorithmic approach to network location problems. I: The p-centers, *SIAM J. Appl. Math.* 37(1979), 513–538.

[17] H. Kellerer, U. Pferschy and D. Pisinger, *Knapsack Problems*, Springer Verlag, Berlin, 2004.

[18] R.F. Love, J.G. Morris and G.O. Wesolowsky, *Facilities Location: Models and Methods*, North-Holland, New York, 1988.

[19] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*, John Wiley, Chichester, 1990.

[20] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Comput.* 6(2007), 5–16.

[21] B.P. Mirchandani, R.L. Francis, *Discrete Location Theory*, John Wiley, New York, 1990.

[22] J. Vygen, On dual minimum cost flow algorithms, *Mathematical Methods of Operations Research* 56(2002), 101–126.

[23] X. Yang and J. Zhang, Inverse center location problem on a tree, *Journal of Systems Science and Complexity* 21(2008), 651–664.

[24] J. Zhang, Z. Liu and Z. Ma, Some reverse location problems, *European Journal of Operational Research* 124(2000), 77–88.