Noname manuscript No.
(will be inserted by the editor)

Inverse 1-center location problems with edge length augmentation on trees

Behrooz Alizadeh · Rainer E. Burkard · Ulrich Pferschy

Received: date / Accepted: date

Abstract This paper considers the inverse 1-center location problem with edge length augmentation on a tree network T with $n+1$ vertices. The goal is to increase the edge lengths at minimum total cost subject to given modification bounds such that a prespecified vertex s becomes an absolute 1-center under the new edge lengths. Using a set of suitably extended AVL-search trees we develop a combinatorial algorithm which solves the inverse 1-center location problem with edge length augmentation in $O(n \log n)$ time. Moreover, it is shown that the problem can be solved in $O(n)$ time if all the cost coefficients are equal.

Keywords network center location · inverse optimization · combinatorial optimization · AVL-search tree

1 Introduction

Facility location problems belong to basic optimization models in the field of operations research and have received strong theoretical interest due to their relevance in practice. These problems involve determining the optimal locations of one or more new facilities in network systems or in space in order to fulfill the demands of customers. For detailed

The first author acknowledges financial support by the NAWI-Project in joint cooperation between Graz University of Technology and Karl-Franzens University of Graz under the grant F-NW-MATH-05. This research has also been supported by the Austrian Science Fund (FWF) Project P18918-N18.

B. Alizadeh
Graz University of Technology, Institute of Optimization and Discrete Mathematics, Steyrergasse 30, 8010 Graz, Austria. E-mail: alizadeh@opt.math.tugraz.at
Sahand University of Technology, Faculty of Basic Sciences for Engineering, Department of Applied Mathematics, Tabriz, Iran. E-mail: alizadeh@sut.ac.ir

R.E. Burkard
Graz University of Technology, Institute of Optimization and Discrete Mathematics, Steyrergasse 30, 8010 Graz, Austria. E-mail: burkard@opt.math.tugraz.at

U. Pferschy
University of Graz, Department of Statistics and Operations Research, Universitaetsstr. 15, 8010 Graz, Austria. E-mail: pferschy@uni-graz.at

surveys on location problems the reader is referred to the books of Daskin [8], Drezner et al. [9], Francis et al. [10], Love et al. [19] and Mirchandani et al. [20].

Two well-known location problems are the 1-median and 1-center location model. While the goal of the 1-median problem is to minimize the sum of (weighted) distances between a facility and its customers, the 1-center location problem seeks to minimize the maximum among the (weighted) distances to the customers. Such minimax location 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 variants of the 1-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 $V(G)$ and edge set $E(G)$ 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 we want 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 with respect to the edge lengths ℓ. 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, then we say that the optimal solution p^* is a *vertex 1-center location* of G.

The inverse version of optimization problems, particularly *inverse location problems* have found significant interest in recent years. 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 optimal with respect to the new parameter values.

For a detailed survey on inverse optimization problems see the article by Heuberger [16]. In the context of location problems Cai et al. [6] 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 et al. [4] 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 modification on tree networks. Hatzl [17] as well as Galavii [11] showed later that this problem can actually be solved in $O(n)$ time. Moreover, Burkard et al. [4] 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 [5] 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 [12] suggested an $O(n \log n)$ time solution method for the inverse 1-maxian (or negative weight 1-median) problem with edge length modifications on tree networks. The inverse Fermat-Weber problem was studied by Burkard et al. [3]. 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 [11] 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 [13] 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 Alizadeh and Burkard [1] investigated the inverse 1-center location problem on tree networks in which one is allowed to increase or reduce the edge lengths. They showed that the problem can be solved in $O(n^2)$ time by an exact algorithm provided that no essential topology change occurs on the tree. Dropping this condition, they proposed an $O(n^2r)$ time exact algorithm for the general case where r is the compressed depth of the underlying tree.

In this paper we consider the inverse 1-center location problem with edge length augmentation (a special case of the problem investigated in [1]) on a tree in which the aim is only to increase the edge lengths of the given tree at minimum total cost with respect to modification bounds so that a prespecified vertex s becomes an absolute 1-center. We propose new solution algorithms with improved time complexities.

This article is organized as follows: In Section 2 we recall the fundamental properties of the classical network 1-center problem and formulate the inverse 1-center location problem with edge length augmentation. Then we derive a basic solution idea for the problem under investigation. In Subsection 3.1 we introduce a data structure based on AVL-search trees which can be used to develop an $O(n \log n)$ time exact algorithm in Subsection 3.2. Finally, in Section 4 we propose a linear time solution method for the case that all the cost coefficients are equal.

2 Problem formulation and basic solution idea

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 $\ell(e)$. Let s be a prespecified vertex on T. We want to increase the edge lengths at minimum total cost such that s becomes the absolute 1-center. Suppose that we incur the nonnegative cost $c^+(e)$ if $\ell(e)$ is increased by one unit. Moreover, we assume that it is not possible to increase the edge lengths arbitrarily but every increased edge length must satisfy an upper bound $\ell_{upp}(e)$. For convenience, let

$$\ell^+(e) = \ell_{upp}(e) - \ell(e) \quad \text{for all} \quad e \in E(T).$$

Thus we can state the inverse 1-center location problem with edge length augmentation on the given tree network T as follows:

Increase the edge lengths $\ell(e),\ e \in E(T)$, by an amount $x(e)$ with $\tilde{\ell}(e) = \ell(e) + x(e)$ such that the following statements (i), (ii) and (iii) are fulfilled:

(i) Vertex s becomes an absolute 1-center of T with respect to $\tilde{\ell}$, i.e.,

$$\max_{v \in V(T)} d_{\tilde{\ell}}(v, s) \leq \max_{v \in V(T)} d_{\tilde{\ell}}(v, p) \quad \text{for all } p \in T.$$

(ii) The linear cost function $\sum_{e\in E(T)} c^+(e)x(e)$ becomes minimum.

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

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

Hence, the inverse 1-center location problem on the tree network T can be formulated as the following *nonlinear semi-infinite optimization model* with an infinite number of nonlinear constraints:

$$\begin{array}{lll} \text{minimize} & \displaystyle\sum_{e\in E(T)} c^+(e)x(e) & \\ \text{subject to} & \max d_{\tilde{\ell}}(v,s) \le \max d_{\tilde{\ell}}(v,p) & \text{for all } p \in T, \\ & \tilde{\ell}(e) = \ell(e) + x(e) & \text{for all } e \in E(T), \\ & x(e) \le \ell^+(e) & \text{for all } e \in E(T), \\ & x(e) \ge 0 & \text{for all } e \in E(T). \end{array} \tag{2}$$

The above problem is a special case of the more general inverse 1-center location problem where also reduction of edge lengths is allowed. For the latter Alizadeh and Burkard in [1] proposed an $O(n^2r)$, $r < n$, time solution algorithm. However, we are now going to use the specific structure of (2) in order to develop faster algorithms. In case of arbitrary cost coefficients, we develop an $O(n \log n)$ time algorithm. The case of equal costs can be solved in $O(n)$ time. Both algorithms are based on the following optimality criterion, the so-called *midpoint-property*, introduced by Handler [15]:

Theorem 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.*

Furthermore, Handler [15] proved the following Lemma:

Lemma 2 *The absolute* 1*-center of an unweighted tree network is unique.*

Considering Theorem 1 and Lemma 2, we conclude that for solving the inverse 1-center location problem on a given tree T, it is sufficient to increase the edge lengths of T at minimum total cost within the given bounds until the vertex s becomes the midpoint of a longest path on T with respect to the new edge lengths.

Let $\deg(s)$ denote the degree of the prespecified vertex s. If $\deg(s) = 0$, then s is the absolute 1-center of T. If $\deg(s) = 1$, then the problem has no solution and s can not become the absolute 1-center of T. Therefore, we assume that $\deg(s) \ge 2$. We partition T into nontrivial subtrees $T_1, ..., T_{\deg(s)}$ such that

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

Now let P_0 be a longest path from s to one of the leaves of T. Assume w.l.o.g. that $P_0 \subseteq T_1$. Define

$$T' = \bigcup_{i=2}^{\deg(s)} T_i$$

Let $P = \{P_1, ..., P_{\tilde{n}}\}$, $\tilde{n} \le n-1$, be the set of all paths from s to the leaves of the subtree T'. Moreover, we denote the length of a path P_i by $\ell(P_i)$. As an immediate consequence of the Lemmas 4.1 and 4.2 provided by Alizadeh and Burkard [1] we get the following lemma describing which edges of T must be considered for modification:

Lemma 3 *Edge lengths are only increased along a path P_k, $k \in \{1, \dots, \tilde{n}\}$, whose length $\ell(P_k)$ can be increased at minimum total cost to the target value $\ell(P_0)$.*

Based on the above considerations we are now going to develop a combinatorial algorithm with improved running time for the inverse 1-center location problem with arbitrary costs.

3 Inverse 1-center location problem with arbitrary costs

First let us to define

$$\xi_i = \ell(P_0) - \ell(P_i) \qquad \text{for } i = 1, ..., \tilde{n},$$

where ξ_i is the amount by which the length of P_i must be increased to be equal to $\ell(P_0)$. For every $P_i \in P$, $i = 1, ..., \tilde{n}$, let

$$E(P_i) = \{e_1^i, ..., e_{r_i}^i\}, \qquad 1 \le r_i \le n.$$

Corresponding to any path $P_i \in P$, $i = 1, ..., \tilde{n}$, we define the following linear programming model which is able to increase the length of P_i at minimum total cost such that s becomes the midpoint of a longest path in T.

$$\begin{aligned} (\mathrm{LP_i}) \qquad \text{minimize} \quad & \sum_{j=1}^{r_i} c^+(e_j^i) x(e_j^i) \\ \text{subject to} \quad & \sum_{j=1}^{r_i} x(e_j^i) = \xi_i \\ & x(e_j^i) \le \ell^+(e_j^i) \qquad \text{for } j = 1, ..., r_i, \\ & x(e_j^i) \ge 0 \qquad \text{for } j = 1, ..., r_i. \end{aligned}$$

If we introduce the new notations

$$\ell_{ij} = \ell^+(e_j^i), \quad x_j = \frac{x(e_j^i)}{\ell^+(e_j^i)} \quad \text{and} \quad c_{ij} = c^+(e_j^i)\ell^+(e_j^i) \qquad \text{for } j = 1, ..., r_i,$$

then problem $(\mathrm{LP_i})$ can be written as

$$\begin{aligned} (\mathrm{CKP_i}) \qquad \text{minimize} \quad & Z_i = \sum_{j=1}^{r_i} c_{ij} x_j \\ \text{subject to} \quad & \sum_{j=1}^{r_i} \ell_{ij} x_j = \xi_i, \\ & 0 \le x_j \le 1 \qquad \text{for } j = 1, ..., r_i, \end{aligned}$$

where ℓ_{ij}, ξ_i and c_{ij} are positive real numbers. Problem $(\mathrm{CKP_i})$ is a continuous knapsack problem in minimization form (see e.g. [18, Sec. 13.3.3]). For solving the inverse 1-center location problem on the tree network T, it is sufficient to find all optimal objective values Z_i^*, $i = 1, \dots, \tilde{n}$, and then determine

$$k \in \operatorname{argmin}\{Z_i^* : i = 1, \dots, \tilde{n}\}.$$

The path P_k corresponding to problem $(\mathrm{CKP_k})$ will be the best candidate for edge length modification and thus the optimal solution of the inverse 1-center location problem on tree T can be obtained from the optimal solution of the problem $(\mathrm{CKP_k})$.

It is well known that the solution of a continuous minimization knapsack problem with capacity c and n items, each of them defined by a profit p_j and a weight w_j, $j = 1, \dots, n$, can be derived by simply sorting the items in increasing order of their profit to weight ratios (cf. [18]).

Lemma 4 *After renumbering the items of a continuous minimization knapsack problem such that* $\frac{p_1}{w_1} \le \frac{p_2}{w_2} \le \dots \le \frac{p_n}{w_n}$*, the optimal solution vector* x^* *is given by*

$$\begin{aligned}
x_j^* &= 1, && \text{for } j = 1, \dots, b-1,\\
x_b^* &= \frac{1}{w_b}\left(c - \sum_{j=1}^{b-1} w_j\right), &&\\
x_j^* &= 0, && \text{for } j = b+1, \dots, n,
\end{aligned}$$

where the break item b *is given by the smallest index such that* $\sum_{j=1}^{b} w_j > c$.

Moreover, Balas and Zemel [2] gave an algorithm to compute the break item and thus the optimal solution x^* in linear time based on the linear time median algorithm.

Lemma 5 *The break item* b *and the optimal solution* x^* *of a continuous minimization knapsack problem can be computed in* $O(n)$ *time.*

A simple way of finding the optimal objective values $Z_i^*, i = 1, ..., \tilde{n}$, is to solve all the problems $(\mathrm{CKP_i})$, $i = 1, ..., \tilde{n}$, separately. Based on Lemma 5 the overall time complexity of this approach is $O(n^2)$.

But we are interested in finding an exact solution method with lower complexity. Our construction is based on the fact that neighboring paths (as they are determined by a depth-first search ordering of leaves) are likely to differ only in a small number of edges. Therefore, we keep the edges corresponding to every knapsack problem $(\mathrm{CKP_i})$, i.e. every path P_i, $i = 1, ..., \tilde{n}$, organized in an AVL-search tree T_i^{avl}. Moving from one AVL-tree to the next, i.e. from one path to the next, requires only a limited number of update operations. Thereby, we will develop a solution algorithm which solves the inverse 1-center location problem with edge length augmentation in $O(n \log n)$ time.

3.1 Construction of AVL-search trees T_i^{avl}

We will refrain from going into the technical details of AVL-trees, which can be found e.g. in [14], and concentrate on their application to our problem. This will require some extension of their original concept. An AVL-search tree T_i^{avl} is a binary tree such that each node is an object (key, info) where "key" is called the key field of the node, and "info" is called the information element containing pointers to all adjacent nodes and additional data about the structure of the tree. This tree satisfies the following two properties:

- Binary search-tree property:
  For each node α of T_i^{avl}, if α' is a node in the left subtree of α, then the key field of α' is not less than the key field of α. If α' is a node in the right subtree of α, then the key field of α is not less than the key field of α'.
- Height balance property:
  For each node α of T_i^{avl}, the height of the left and right subtrees of α differ by at most 1.

Pointers to missing child or parent nodes are set to NULL. In our application, each node of T_i^{avl} corresponds to exactly one item of the problem ($\mathrm{CKP_i}$), i.e. one edge of P_i. In every node α of T_i^{avl} we also store the following information thus augmenting the standard model:

j_α : corresponding item of α in ($\mathrm{CKP_i}$), i.e. an index $j \in \{1, \ldots, r_i\}$
c_α: cost coefficient of the corresponding item of α in ($\mathrm{CKP_i}$) given by c_{ij_α}
w_α: weight of the corresponding item of α in ($\mathrm{CKP_i}$) given by ℓ_{ij_α}
key_α: (key field of α) efficiency of the corresponding item of α in ($\mathrm{CKP_i}$) which is equal to the value $\frac{c_\alpha}{w_\alpha}$
$left_\alpha$: pointer to the left child node of α in T_i^{avl}
$right_\alpha$: pointer to the right child node of α in T_i^{avl}
C_α: sum of costs over all nodes of the subtree rooted at α (including c_α)
W_α: sum of weights over all nodes of the subtree rooted at α (including w_α)

It will turn out that we can search for the break item of ($\mathrm{CKP_i}$) in the AVL-search tree T_i^{avl} in $O(\log n)$ time. We will generate all trees T_i^{avl}, $i = 1, ..., \tilde{n}$, in the order of discovering the leaves corresponding to the paths P_i, $i = 1, ..., \tilde{n}$, in the subtree T' in a depth-first search manner by performing the following algorithm:

Algorithm Cons(AVL): constructs all AVL-search trees T_i^{avl}

(i) Perform a depth-first search procedure on T' and explore all the edges of a path P_1 until its endpoint is discovered. Construct the AVL-search tree T_1^{avl} by $|E(P_1)|$ insertion operations for these edges. Let $i = 1$.
(ii) Construct T_{i+1}^{avl} from T_i^{avl} for $i = 1, ..., \tilde{n} - 1$:
Perform $|E(P_i)\backslash E(P_{i+1})|$ removal operations on T_i^{avl} for edges in $E(P_i)\backslash E(P_{i+1})$. Insert new nodes into T_i^{avl} corresponding to the edges in $E(P_{i+1})\backslash E(P_i)$ by performing $|E(P_{i+1})\backslash E(P_i)|$ insertion operations resulting in the tree T_{i+1}^{avl}. After every insertion and removal operation rebalancing by rotations is executed to maintain the height balance property. This effects at most $O(\log n)$ nodes.
Update the information of the nodes of T_{i+1}^{avl}.

The main extension of our data structure from the classical AVL-tree model are the fields C_α and W_α concerning the subtrees rooted in every node. Since the rebalancing operation consists of rotations, which change the structure of at most logarithmically many nodes, these values can be reconstructed in $O(\log n)$ time during every rebalancing step.

The following lemma describes the required total time for the construction of all AVL-search trees T_i^{avl} (see e.g. [14]):

Lemma 6 *In an AVL-search tree T_i^{avl} every insertion, removal and rebalancing operation is performed in $O(\log |E(T_i^{avl})|)$ time.*

Lemma 7 *All AVL-search trees $T_1^{avl}, ..., T_{\tilde{n}}^{avl}$ are constructed in $O(n \log n)$ total time.*

Proof The performance of the depth-first search procedure on T' takes $O(n)$ time. During the construction of $T_1^{avl}, ..., T_{\tilde{n}}^{avl}$, every edge $e \in E(T')$ is inserted and removed at most once which according to Lemma 6 takes $O(n \log n)$ time in total. At most $2n$ rebalancing operations take $O(\log n)$ time each, and they also require the updating of information in $O(\log n)$ nodes. This yields a total running time of $O(n \log n)$.

3.2 An $O(n \log n)$ time solution algorithm

In the continuous knapsack problem ($\mathrm{CKP_i}$) we say that an item $j \in \{1, ..., r_i\}$ is packed in the knapsack if $x_j > 0$. Based on a given AVL-search tree T_i^{avl} representing problem ($\mathrm{CKP_i}$) with capacity $cap = \xi_i$ we can compute the break item and the optimal objective value Z_i as follows:

Let $\alpha =$ root of T_i^{avl}. If the inequality

$$W_{right_\alpha} \geq cap \tag{3}$$

holds, then item j_α can not be packed into the knapsack. Hence all the packed items including the break item will be items corresponding to some nodes of the subtree rooted in $right_\alpha$ of T_i^{avl}. In this case we set $\alpha := right_\alpha$ and repeat the above procedure. Otherwise, if

$$W_{right_\alpha} + w_\alpha \geq cap, \tag{4}$$

then we conclude according to Lemma 4 that j_α is the break item of ($\mathrm{CKP_i}$) and all the items in the subtree rooted in $right_\alpha$ must be packed in the knapsack. The packing of these items in the knapsack increases the objective value of problem ($\mathrm{CKP_i}$) by the amount

$$C_{right_\alpha} + \frac{cap - W_{right_\alpha}}{w_\alpha} c_\alpha.$$

On the other hand, if both (3) and (4) do not hold, then it means that an optimal solution of the problem ($\mathrm{CKP_i}$) includes item j_α completely and also all the items on the subtree rooted in $right_\alpha$, describing an increase in the objective value of ($\mathrm{CKP_i}$) by the amount

$$C_{right_\alpha} + c_\alpha.$$

Moreover, the optimal solution contains the packing of the items on some nodes of the subtree rooted in $\alpha := left_\alpha$. In this case we update the capacity cap by

$$cap := cap - W_{right_\alpha} - w_\alpha,$$

and perform the above procedure with respect to the new α and cap.

Note that the algorithm terminates if either (3) does not hold and (4) is satisfied or $\alpha =$ NULL. In the first case the algorithm finds the optimal objective value Z_i^* and a break item of the knapsack problem ($\mathrm{CKP_i}$). In the second case ($\mathrm{CKP_i}$) is infeasible and we assign $Z_i^* := M$, where M is a very big value.

The preceding considerations are summarized in Algorithm 1.

Lemma 8 *Given an AVL-search tree T_i^{avl}, $1 \leq i \leq \tilde{n}$, Algorithm 1 runs in $O(\log n)$ time.*

Algorithm 1 Finds the break item b_i and the optimal objective value Z_i^* of problem (CKP$_\mathrm{i}$) using the AVL-search tree T_i^{avl}.

```
1: $cap := \xi_i$; $Z_i^* := 0$
2: $\alpha :=$ root of $T_i^{avl}$
3: repeat
4:    if $W_{right_\alpha} \geq cap$ then
5:       $\alpha := right_\alpha$
6:    else if $W_{right_\alpha} + w_\alpha \geq cap$ then
7:       $Z_i^* := Z_i^* + C_{right_\alpha} + \frac{cap - W_{right_\alpha}}{w_\alpha} c_\alpha$
8:       $b_i := j_\alpha$; stop
9:    else
10:      $Z_i^* := Z_i^* + C_{right_\alpha} + c_\alpha$
11:      $cap := cap - W_{right_\alpha} - w_\alpha$
12:      $\alpha := left_\alpha$
13:      if $\alpha =$ NULL then
14:         $Z_i^* := M$; stop
15:      end if
16:   end if
17: until forever
```

Proof Every execution of the **repeat**-loop runs in $O(1)$ time. Since this loop is iterated at most $O(\log n)$ times on the AVL-search tree T_i^{avl} the total running time of the algorithm is bounded by $O(\log n)$.

The proof of correctness of Algorithm 1 is given in the following theorem:

Theorem 9 *Algorithm 1 correctly finds the optimal objective value of the knapsack problem* (CKP$_\mathrm{i}$) *for any* $i = 1, ..., \tilde{n}$.

Proof For a given $i \in \{1, ..., \tilde{n}\}$, let $\alpha_1, ..., \alpha_{r_i}$, be the set of nodes of the AVL-search tree T_i^{avl} in the ordering they are visited by an *inorder tree walk* (see Cormen et al. [7, p. 254]). It follows from the search tree property of T_i^{avl} that

$$key_{\alpha_1} \leq key_{\alpha_2} \leq ... \leq key_{\alpha_{r_i}},$$

where $key_{\alpha_j} = \frac{c_{\alpha_j}}{w_{\alpha_j}}$.

By the conditions in lines 4 and 6 Algorithm 1 finds a node α_{b_i} such that

$$\sum_{j=1}^{b_i-1} w_{\alpha_j} < cap \quad \text{and} \quad \sum_{j=1}^{b_i} w_{\alpha_j} \geq cap,$$

where $cap = \xi_i$. But this is exactly the definition of the break item of (CKP$_\mathrm{i}$) given in Lemma 4. Applying Lemma 4 further we can state the optimal solution of the underlying continuous minimization knapsack problem as

$$\begin{aligned} x_j &= 1, && \text{for } j = 1, \ldots, b_i - 1, \\ x_{b_i} &= \frac{1}{w_{\alpha_{b_i}}} \left(cap - \textstyle\sum_{j=1}^{b_i-1} w_{\alpha_j} \right), && \\ x_j &= 0, && \text{for } j = b_i + 1, \ldots, r_i. \end{aligned} \tag{5}$$

Plugging the values of (5) into the objective function of (CKP$_\mathrm{i}$) yields

$$Z_i^* = \sum_{j=1}^{b_i-1} c_{\alpha_j} + \frac{cap - \sum_{j=1}^{b_i-1} w_{\alpha_j}}{w_{\alpha_{b_i}}} c_{\alpha_{b_i}}, \tag{6}$$

which is exactly the expression computed in line 7 of Algorithm 1. Recalling that $c_\alpha = c_{ij_\alpha}$ and $w_\alpha = \ell_{ij_\alpha}$ the statement of the theorem follows.

Recall that for solving the inverse 1-center location problem under investigation, we need to choose the best candidate path P_k by

$$k \in \operatorname{argmin}\{Z_i^* : i = 1, ..., \tilde{n}\}, \tag{7}$$

and then derive the optimal solution of the continuous knapsack problem ($\mathrm{CKP_k}$). To do so we put together the pieces developed so far and run **Algorithm Cons(AVL)**. At the end of each of its iterations when we have fully constructed an AVL-tree T_i^{avl} we call **Algorithm 1** to compute Z_i^*. Taking the minimum value over all Z_i^* we identify Z_k^*. If $Z_k^* = M$ the problem is infeasible, otherwise we have obtained the break item b_k of ($\mathrm{CKP_k}$) and compute the optimal solution vector x^* of ($\mathrm{CKP_k}$) according to Lemma 4 in linear time. Finally, we set the optimal solution of the original problem (2) as

$$x^*(e) = \begin{cases} \ell^+(e)x'(e) & \text{if } e \in \{e_1^k, ..., e_{r_k}^k\}, \\ 0 & \text{otherwise,} \end{cases}$$

where

$$x'(e_j^k) = x_j^* \quad \text{for } j = 1, \dots, r_k.$$

Altogether we get the following theorem:

Theorem 10 *The inverse 1-center location problem with edge length augmentation can be solved in $O(n \log n)$ time on a tree with n edges.*

Proof The above observations imply that the optimal solution of the inverse 1-center location problem is given by Z_k^* as defined in (7). Obviously, this value is derived by taking the minimum over all $\tilde{n}$ executions of Algorithm 1.

The time complexity follows from Lemma 7 and the at most n executions of Algorithm 1 each of them requiring $O(\log n)$ time as stated in Lemma 8.

4 Inverse 1-center location problem with equal costs

In this section we assume that all the cost coefficients assigned to the edge lengths of the tree network T are equal. We will show that this special case of the inverse 1-center location problem on trees can be solved efficiently in $O(n)$ time. Observe that the maximum permissible amount by which the length $\ell(P_i)$ can be increased is given by

$$\ell^+(P_i) = \sum_{e \in E(P_i)} \ell^+(e).$$

Now we let

$$I = \{i : \ell^+(P_i) \geq \xi_i, \ i = 1, ..., \tilde{n}\}.$$

Clearly, by increasing the length of every path P_i, $i \in I$, we can fulfill the midpoint-property on the tree network T. But the best candidate path P_k for edge length modification is determined by

$$k \in \operatorname{argmax}\{\ell(P_i) : i \in I\}.$$

Hence, the prespecified vertex s becomes the absolute 1-center of T at minimum total cost if we increase the length of P_k by the amount ξ_k. Since all the cost coefficients assigned to the edge lengths are equal, it is not necessary to consider an ordering for increasing the length of edges of the best candidate path P_k. On the other hand, note that the computation of all amounts $\ell^+(P_i)$, $i = 1, ..., \tilde{n}$, takes $O(n)$ total time if we traverse the given tree network T in a depth-first search manner. Moreover, the set I and the index k are determined in $O(n)$ time. Therefore, we conclude:

Theorem 11 *The inverse 1-center location problem with edge length augmentation can be solved in $O(n)$ time on a tree with n edges provided that all cost coefficients are equal.*

References

1. B. Alizadeh and R.E. Burkard, The inverse 1-center location problem on a tree, *Technical Report 2009-03*, Graz University of Technology, 2009.
2. E. Balas, E. Zemel, An algorithm for large zero-one knapsack problems, *Operations Research*, **28** (1980) 1130–1154.
3. R.E. Burkard, M. Galavii and E. Gassner, The inverse Fermat-Weber problem, *Technical Report 2008-14*, Graz University of Technology, 2008.
4. R.E. Burkard, C. Pleschiutschnig and J. Zhang, Inverse median problems, *Discrete Optimization*, **1** (2004) 23–39.
5. R.E. Burkard, C. Pleschiutschnig and J. Zhang, The inverse 1-median problem on a cycle, *Discrete Optimization*, **5** (2007) 242–253.
6. 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.
7. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, 2001.
8. M.S. Daskin, *Network and Discrete Location: Modeles, Algorithms and Applications*, John Wiley, New York, 1995.
9. Z. Drezner and H.W. Hamacher, *Facility Location, Applications and Theory*, Springer Verlag, Berlin, 2004.
10. R.L. Francis, L.F. McGinnis and J.A. White, *Facility Layout and Location, An Analytical Approach*, Prentice Hall, Englewood Cliffs, 1992.
11. M. Galavii, *Inverse* 1*-Median Problems*, Ph.D. Thesis, Institute of Optimization and Discrete Mathematics, Graz University of Technology, Graz, Austria, 2008.
12. E. Gassner, The inverse 1-maxian problem with edge length modification, *J. Combinatorial Optimization*, **16** (2007) 50–67.
13. E. Gassner, An inverse approach to convex ordered median problems in trees, Technical Report 2008-16, Graz University of Technology, 2008.
14. M.T. Goodrich, R. Tamassia and D. Mount, *Data Structures and Algorithms in C++*, John Wiley and Sons, New York, 2003.
15. G.Y. Handler, Minimax location of a facility in an undirected tree graph, *Transportation Science*, **7** (1973) 287–293.
16. C. Heuberger, Inverse combinatorial optimization: A survey on problems, methods, and results, *Journal of Combinatorial Optimization*, **8** (2004) 329–361.
17. J. Hatzl, personal communication, 2006.
18. H. Kellerer, U. Pferschy and D. Pisinger, *Knapsack Problems*, Springer, 2004.
19. R.F. Love, J.G. Morris and G.O. Wesolowsky, *Facilities Location: Models and Methods*, North-Holland, New York, 1988.
20. B.P. Mirchandani, R.L. Francis, *Discrete Location Theory*, John Wiley, New York, 1990.