INVERSE SINGLE FACILITY LOCATION PROBLEM ON A TREE WITH BALANCING ON THE DISTANCE OF SERVER TO CLIENTS

Shahede Omidi and Jafar Fathali*
Faculty of Mathematical Sciences
Shahrood University of Technology
University Blvd., Shahrood, Iran
(Communicated by Leen Stougie)

Abstract. We introduce a case of inverse single facility location problem on a tree where by minimum modifying in the length of edges, the difference of distances between the farthest and nearest clients to a given facility is minimized. Two cases are considered: bounded and unbounded nonnegative edge lengths. In the unbounded case, we show the problem can be reduced to solve the problem on a star graph. Then an $O(n \log n)$ algorithm is developed to find the optimal solution. For the bounded edge lengths case an algorithm with time complexity $O(n^2)$ is presented.

1. Introduction. The classical location models deal with finding the location of facilities with respect to different criteria such as minimizing time, cost and distances between the clients and facilities. Among them the $p$-median and $p$-center problems are two basic facility location models. In these problems, the goal are finding the location of $p$ facilities such that respectively the sum and maximum weighted distances from clients to the nearest facility is minimized. For more details about classical location models, see e.g. [23]. In some real applications, the facilities may already exist and one or more parameters of the problem should be changed with minimum cost, such that the given facility locations are optimal. This kind of location models are called inverse facility location problems. The inverse $p$-median and $p$-center problems have been considered by many authors. The NP-hardness of the inverse center problem has been shown by Cai et al. [10]. Alizadeh et al. [2] presented an $O(n \log n)$ algorithm for the inverse 1-center problem with edge length augmentation on trees, where $n$ is the number of vertices of tree. Later, Alizadeh and Burkard [3] showed that the inverse 1-center problem can be solved in $O(n^2)$ time. Recently, Alizadeh and Etemad [4] developed an algorithm for inverse obnoxious center problem on graphs. Burkard et al. [8] investigated the inverse $p$-median problem and developed an $O(n \log n)$ algorithm for the inverse 1-median problem on a tree. Galavii [14] improved the time complexity of this problem to linear time. When the underlying network is a cycle, Burkard et al. [9] proposed

2020 Mathematics Subject Classification. Primary: 90B80; Secondary: 90B06.
Key words and phrases. Inverse facility location, balanced allocation, equality measure, variable edge length.

* Corresponding author: Jafar Fathali.
an $O(n^2)$ algorithm for the inverse 1-median problem. The inverse 1-median and 1-center problems on trees with Chebyshev and Hamming norms have been considered by Guan and Zhang [16] and Nguyen and Sepasian [26], respectively. Sepasian and Rahbarnia [30] investigated the inverse 1-median problem with varying vertex and edge length on trees. Nguyen [25] developed an algorithm for the inverse 1-median problem with variable vertex weight on block graphs. Recently, Nazari et al. [24] considered the inverse of backup 2-median problem on trees. Alizadeh et al. [1] proposed a combinatorial algorithm for the inverse obnoxious median problem on trees.

The balanced facility location problems deal with finding the location of the facilities such that the equality in serving to the demand points is maximized. This kind of facility location models have been considered by many authors. Among them, the model of maximizing the difference of distances between a demand point to its farthest and nearest facility has been considered by Gavalec and Hudec [15]. Berman et al. [6] studied the problem of finding the location of $p$ facilities such that the maximum weight assigned to each facility is minimized. Marin [21] deals with the case that the difference between the maximum and minimum weights allocated to different facilities is minimized. Fathali and Zaferanieh [13] developed polynomial time algorithms for the balanced 2-median and obnoxious 2-median models on trees. Davoodi [11] presented a multi-objective model for balanced center location problem. Omidi et al. [27] considered the inverse 2-facility location problem on a tree which aim to modify the edge lengths such that the problem is balanced. Lejeune and Prasad [20] proposed a bi-criteria model for the problem in which has a trade off between effectiveness and equity. Landete and Marin [19] studied the problem of minimizing the differences of the weights that allocated to the facilities. Barbati and Piccolo [5] developed some properties of the equality measures in facility location models to describe the behavior of this problems. Sayar et al. [29] considered the problem of balancing allocation with regard to the efficiency of the servers. Pham et al. [28] discussed the balance vertices on trees and developed a linear time algorithm to find the set of balance vertices on the underlying tree. For more details and literature reviews on balancing in location theory see e.g. [22, 12].

The 1-center problem consider the finding the location of a new facility such that the distance of farthest client to the new facility is minimized. Therefore, just the clients on the farthest points are investigated. To make a balance in serving the clients, other kind of objective functions can be considered. One of them, is minimizing the difference of distances between the farthest and closest clients to the new facility. Applications of this kind of location model arise when some clients may move to the location of other clients. In this paper, we consider the inverse of this single facility location problem on a tree, i.e. by modifying the lengths of edges with minimum total cost the difference of distances between the farthest and nearest clients to a given facility is minimized. Two cases are considered: 1- the length of edges are unbounded and nonnegative, 2- the length of edges are bounded and positive.

In what follows the considered inverse single facility model is defined in Section 2. In Section 3 we show that the problem with nonnegative unbounded edge lengths on tree graphs can be reduced to solve the problem on a star graph. Then an $O(n\log n)$ algorithm is presented to solve this problem in Section 3.1. In Section 4 we consider the problem with bounded edge lengths on a tree and develop an $O(n^2)$ algorithm.
2. Problem definition. Let the tree $T$ with vertices set $V$, $|V| = n$, and edges set $E$, be given. For each edge $e_i \in E$, let $l_i \geq 0$ be the length of edge $e_i$. Let $d(v_i, v_j)$ be the length of the shortest path between vertices $v_i$ and $v_j$ in $T$. Let $x \in V$ be the location of the given facility in $T$. In the inverse balanced facility location problem we want to modify the lengths of edges with minimum cost such that the difference distances between the farthest and closest clients to the given facility is minimized. For any edges $e_i$, suppose that the cost of increasing per unit of $l_i$ is $c_i^+ \geq 0$ and the cost of decreasing per unit of $l_i$ is $c_i^- \geq 0$. Let $q_i^+$ and $q_i^-$ be the amounts by which the length $l_i$ is increased and decreased, respectively. For each $e_i \in E$, let $\hat{l}_i = l_i + q_i^+ - q_i^-$. Note that the new length of edges should not be negative, i.e. $\hat{l}_i \geq 0$ or $q_i^- \leq l_i$, for every $e_i \in E$.

Let

$$f_1 = \sum_{e_i \in E} (c_i^+ q_i^+ + c_i^- q_i^-),$$

$$f_2 = \max_{v_i \in V} \hat{d}(v_i, x) - \min_{v_i \in V} \hat{d}(v_i, x),$$

Then in our inverse model, among modified edge lengths minimizing $f_2$, we want to find the modification that minimized $f_1$.

As mentioned in Heuberger [18], possible application of the inverse location models arises when we are faced with the situation that the facility already exist and can not be relocated in reasonable costs. In this case, we may change the weight of vertices or edge lengths of the network as little as possible such that a given facility becomes optimum. Specially, our considered inverse model can be applied in the situation that a road network and the location of a facility server is given and we want to modify the traffic network such that the servicing of the facility to the clients is balanced.

3. The problem with unbounded edge lengths. In this section we consider the unbounded edge lengths case, i.e. $\hat{l}_i \geq 0$, for every $e_i \in E$. We focus on minimizing $f_2$ with minimum cost. Then obviously in this case, the optimal value of $f_2$ is zero, where the lengths of all paths from $x$ to other vertices are equal. Therefore, we can consider the following model.

$$\min f_1 = \sum_{e_i \in E} (c_i^+ q_i^+ + c_i^- q_i^-)$$

$$\text{s.t.}$$

$$\max_{v_i \in V} \hat{d}(v_i, x) - \min_{v_i \in V} \hat{d}(v_i, x) = 0.$$  

$$\hat{l}_i \geq 0$$

Therefore, in this section we try to find a solution that satisfied in (4), i.e. feasible solution, with minimum cost.

Let $P_1$ and $P_2$ be the set of paths from $x$ to the farthest and closest vertices in $T \setminus \{x\}$, respectively. Note that $P_1$ and $P_2$ are not fixed and will be updated during the algorithm. Also the following observations can be considered.
Observation 1. A feasible solution can be obtained by either reduce the length of some edges on paths in $P_1$ or augment the length of some edges on paths in $P_2$.

Observation 2. If there exists two paths $p_i \in P_1$ and $p_j \in P_2$, such that $p_j$ contains $p_i$ then any change on edges of $p_j$ does not improve the feasibility of the solution.

Let $\text{Adj}$ be the set of all adjacent edges of $x$. In the following lemma, we show that if any path in $P_2$ is a part of a path in $P_1$, then in the optimal solution, the length of all nonadjacent edges to $x$ is zero.

**Lemma 3.1.** If there exists a path in $P_1$ which contains a path in $P_2$, then in the optimal solution the length of each edge $e \notin \text{Adj}$ is zero.

**Proof.** Let $p_i \in P_1$ be contains $p_j \in P_2$. Then by observations 1 and 2, we should reduce the edges of $p_i \setminus p_j$. Therefore, to obtain a feasible solution, we set zero the length of all nonadjacent vertices of $x$.

Also we can conclude the following lemmas.

**Lemma 3.2.** If just reduction length of edges is permitted, then to obtain an optimal solution it suffices to set

$$
\hat{l}_i = \begin{cases} 
0 & \text{if } e_i \notin \text{Adj} \\
Lm & \text{if } e_i \in \text{Adj},
\end{cases}
$$

where $Lm$ is the length of any path in $P_2$. In this case the cost of finding the optimal solution can be computed as

$$
f_1 = \sum_{e_i \in \text{Adj}} c^- (l_i - Lm) + \sum_{e_i \notin \text{Adj}} c^- l_i.
$$

**Proof.** Since just reduction is permitted, then the same as Lemma 3.1, in the optimal solution the length of all nonadjacent edges to $x$ are zero. Also the length of all adjacent edges of $x$, that they are not a part of any path in $P_2$ should be reduced to $Lm$. The reduction cost of edges is concluded immediately.

**Lemma 3.3.** If just augmenting length of edges is permitted, then to obtain a feasible solution it suffices to augment the length of edges adjacent to $x$.

In the following theorem we show that to solve the problem on a tree, it suffices to solve it on a more smaller sub-tree.

**Lemma 3.4.** Solving the problem on a tree, is equivalent to find the optimal solution on a tree which induced in $x$ and its adjacent vertices.

**Proof.** The proof is concluded with this fact that to obtain a feasible solution we should reduce and augment the length of some edges on the paths in $P_1$ and $P_2$, respectively. Then we will reach a path in $P_2$ which is a part of a path in $P_1$. Therefore, by Lemma 3.1, the length of all nonadjacent edges to $x$ should be zero in the optimal solution.

Let $T_x$ be the sub-tree of $T$ which is induced by $x$ and its adjacent vertices. Indeed, $T_x$ is a star graph. Therefore, in the next section, we concentrate on the finding the optimal solution on an star graph and show it can be found in $O(n_x \log n_x)$ time, where $n_x$ is the number of edges of $T_x$. Therefore, the following theorem can be concluded for the problem on a tree.
Theorem 3.5. Let $d$ be the degree of $x$ on the tree $T$. Then the inverse single facility location problem with balancing on distances from clients to the server can be solved in $O(d \log d)$ time.

Note that since $d \leq n$, therefore the upper bound of complexity on a tree is $O(n \log n)$.

3.1. The problem on a star graph. Let $T$ be a star graph and $x$ be the vertex of $T$ with degree larger than 1. Note that we consider this case, because by Lemma 3.4 the problem on tree reduce to solve the problem on the mentioned star graph. Let $P_3$ and $P_4$ be the set of longest and shortest edges in $T$, respectively. Then to find the optimal solution on $T$, let

$$d_1 < d_2 < \ldots < d_r$$

be the ascending order of distances between vertices of $T$ and $x$. The following actions can be used to achieve a feasible solution.

1. Reduce the length of each edge $e_j \in P_3$, $j = 1, \ldots, n_1$, where $n_1 = |P_3|$. In this case to guarantee all edges in $P_3$ remain longest edges in $T$, we should decrease the length of all of them, simultaneously. Note that the length of each edges in $P_3$ is $d_r$ and should be reduced to $d_r - 1$. This reduction improves the feasibility with $K_1 = d_r - d_{r-1}$, which needs the following cost

$$C_1 = \sum_{e_j \in P_3} c_j K_1.$$  

(7)

2. Augment the length of all edges in $P_4$. In this case the length of any edge $e$ in $P_4$ is equal to $d_1$ and can be increased to $d_2$. Therefore, the feasibility is improved by

$$K_2 = d_2 - d_1.$$  

(8)

The cost of this length augmentation is

$$C_2 = \sum_{e_i \in P_4} c_i^+ K_2.$$  

(9)

In each iteration of the algorithm we select one of these actions. The action with most feasibility improvement and minimum cost should be chosen. However,

$$\frac{K_1}{C_1} = \frac{1}{\sum_{e_j \in P_3} c_j}$$

and

$$\frac{K_2}{C_2} = \frac{1}{\sum_{e_i \in P_4} c_i^+}.$$  

Therefore, the action corresponding to the following is selected.

$$\min\{\sum_{e_i \in P_4} c_i^+, \sum_{e_j \in P_3} c_j^-\}$$

These ideas lead us to the following algorithm for finding the optimal solution on a star graph.

Algorithm [ISFBDS].

Input: The star graph $T$, where $x$ as the location of the facility, is the vertex of $T$ with degree larger than 1. Also, the cost of increasing and decreasing of edges lengths are given.
Output: The new lengths of edges $\hat{t}_i$ such that $f_2 = 0$.

Initialization:
Let $D = \{d_1, d_2, \ldots, d_r\}$ be the increasing order of edge lengths of $T$, (therefore $d_1 < d_2 < \ldots < d_r$).
Let $P_3$ and $P_4$ be the sets of edges that their lengths are $d_r$ and $d_1$, respectively.
Set $f_1 := 0$, $f_2 := d_r - d_1$, $t := r$ and $s := 1$.
Set
\[
C_1 := \sum_{e_j \in P_3} c_j^-, \quad C_2 := \sum_{e_j \in P_4} c_j^+.
\]

Iteration step
While $f_2 > 0$ do
1. If $C_1 < C_2$ then
   (a) For each edge $e_j \in P_3$, set $l_j := d_{t-1}$.
   (b) Set $f_1 := f_1 + C_1(d_t - d_{t-1})$.
   (c) Set $f_2 := f_2 - (d_t - d_{t-1})$.
   (d) Set $P := \{e_i \in T| l_i = d_{t-1}\}$.
   (e) Set $P_3 := P_3 \cup P$ and
        \[
        C_1 := C_1 + \sum_{e_j \in P} c_j^-.
        \]
   (f) Set $t := t - 1$.
   else
   (a) For every edge $e_j$ in $P_4$, set $l_j := d_{s+1}$.
   (b) Set $f_1 := f_1 + C_2(d_{s+1} - d_s)$.
   (c) Set $f_2 := f_2 - (d_{s+1} - d_s)$.
   (d) Set $P := \{e_i \in T| l_i = d_{s+1}\}$.
   (e) Set $P_4 := P_4 \cup P$ and
        \[
        C_2 := C_2 + \sum_{e_j \in P} c_j^+.
        \]
   (f) Set $s := s + 1$.
End while

End of algorithm

To calculate the time complexity of the algorithm, let $n_T$ be the number of vertices of star $T$. Then the length edges of $T$ can be sorted in $O(n_T \log n_T)$ time. In the worst case we may need to jump the length of an edge from $d_1$ to $d_2$ and then $d_3$ until reach $d_{n_T}$ or vice versa. Therefore, in this case Iteration step will iterated $O(n_T)$ time. Therefore the time complexity of the algorithm is $O(n_T \log n_T)$.

Theorem 3.6. The inverse single facility location problem with balancing on distances from clients to server on stars can be solved in $O(n_T \log n_T)$ time.

To illustrate the presented algorithm consider the following example.

Example 1. Consider the tree $T$ depicted in Fig. 1. The length of edges and the costs of increasing and decreasing the lengths of edges are given in Table 1.

Using Lemma 3.4 it suffices to reduce the length of edges $l_1, l_4, l_5, l_7, l_9$ and $l_{10}$ to zero and find the solution on the remaining star graph. The cost of reduction
INVERSE LOCATION WITH BALANCING DISTANCE

Figure 1. The tree $T$ with 10 vertices.

Table 1. The lengths and costs of changing lengths of edges in tree $T$.

| $e_i$    | $l_i$ | $c_i^+$ | $c_i^-$ |
|----------|-------|---------|---------|
| $e_1 = (v_1, v_2)$ | 3     | 1       | 2       |
| $e_2 = (v_2, x)$    | 3     | 3       | 1       |
| $e_3 = (x, v_3)$    | 5     | 2       | 2       |
| $e_4 = (v_3, v_4)$  | 2     | 1       | 3       |
| $e_5 = (v_4, v_5)$  | 8     | 1       | 4       |
| $e_6 = (x, v_6)$    | 1     | 5       | 3       |
| $e_7 = (v_6, v_7)$  | 2     | 3       | 1       |
| $e_8 = (x, v_8)$    | 3     | 5       | 2       |
| $e_9 = (v_8, v_9)$  | 4     | 3       | 4       |
| $e_{10} = (v_9, v_{10})$ | 3 | 4       | 4       |

The problem with bounded edge lengths. In this section we consider the case that the length of edges are bounded, i.e., for each $e_i \in E$, $0 < l_i \leq \bar{l}_i \leq \bar{l}_i$, where $\bar{l}_i$ and $l_i$ are the upper and lower bounds of the length of edge $e_i$, respectively.

4. The problem with bounded edge lengths. In this section we consider the case that the length of edges are bounded, i.e., for each $e_i \in E$, $0 < l_i \leq \bar{l}_i \leq \bar{l}_i$, where $\bar{l}_i$ and $l_i$ are the upper and lower bounds of the length of edge $e_i$, respectively. In this case the optimal value of $f_2$ may not be zero.

Let $T = (V, E)$ be the underlying network. Let the notations $P_1$, $P_2$, $Lm$ and $Adj$ are defined the same as Section 3. Then the observations 1 and 2 and Lemma 3.3 also hold for bounded edge lengths case. Since we suppose $l_2 > 0$ then the following observation also holds.

**Observation 3.** Each path in $P_2$ contains only one edge which are adjacent to $x$.

**Lemma 4.1.** In the bounded edge lengths case if there exists a path in $P_1$ which contains a path in $P_2$, then in the optimal solution exist a path $p_i \in P_1$ such that for each edge $e_r \in p_i \setminus Adj$

$$\bar{l}_r = \underline{l}$$
Proof. Since there exists a path in \( P_1 \) which contains a path in \( P_2 \), by observation 2, we should just reduce the length of edges on the paths in \( P_1 \). Therefore, we reduce the length of all edges in all paths in \( P_1 \) that are not adjacent to \( x \), until the length of all edges of a path in \( P_1 \) that are not in \( Adj \), reaches their lower bounds.

Now to present the algorithm, we want to choose an edge and reduce or augment its length, in each iteration. Before explaining the algorithm, we need the following definition.

**Definition 4.2.** A path \( p \) in \( P_1 \) is called saturated if the length of all edges in \( p \) reaches their lower bounds, i.e. for each \( e_t \in p \), \( l_t = l_t^- \), and a path \( q \) in \( P_2 \) is called saturated if for each \( e_r \in q \), \( l_r = \bar{l}_r \).

Firstly, let there is no path in \( P_1 \) that contains a path in \( P_2 \). The same as Section 3, let \( d_1 < d_2 < \ldots < d_r \) be the ascending order of distances between vertices of \( T \) and \( x \). The following two actions may be used to achieve a feasible solution.

**Reduction.** Reduce the length of an edge on each path \( p_j \in P_1 \), \( j = 1, \ldots, n_1 \), where \( n_1 = |P_1| \). In this case to guarantee all path in \( P_1 \) remain longest path from \( x \), we should decrease the length of all of them, simultaneously. If there exists a saturated path in \( P_1 \) then we can not reduce the length of longest path, therefore this action is incapable. Otherwise, in each path \( p_j \in P_1 \), \( j = 1, \ldots, n_1 \), we select the edge with minimum reduction cost and reduce its length. Note that if this edge is in \( Adj \) then its length can not be less than \( Lm \). Therefore, for \( j = 1, \ldots, n_1 \), set \( h_j = \text{Argmin}\{c_h^-|e_h \in p_j, l_h > l_h^-\} \) and

\[
RL_j = \begin{cases} 
\min\{l_h_j - Lm, l_h_j - l_h^-\} & \text{if } e_h_j \in Adj \\
l_h^- - l_h_j & \text{if } e_h_j \notin Adj.
\end{cases}
\]

Then we reduce the length of each \( e_h_j \) to \( l_h_j - K_1 \), where

\[ K_1 = \min\{d_r - d_{r-1}, RL_1, \ldots, RL_{n_1}\}, \]

which is the value of feasibility improvement. This reduction needs the following cost

\[
CP_1 = \sum_{p_j \in P_1} c_h^- K_1.
\]

**Augmentation.** Augment the length of all edges on paths in \( P_2 \). In this case the length of each edge \( e_t \) on path \( p_j \in P_2 \) is equal to \( d_t \). Note that \( p_j \) contains singleton edge \( e_t \). Therefore, the feasibility can be improved by

\[ K_2 = \min\{d_2 - d_1, \bar{l}_1 - l_1, \ldots, \bar{l}_{n_2} - l_{n_2}\}, \]

where \( n_2 = |P_2| \).

The cost of this length augmentation is

\[
CP_2 = \sum_{p_t \in P_2, e_t \in p_t} c_h^+ K_2.
\]

In each iteration the action with most feasibility improvement and minimum cost should be chosen. In the same manner of Section 3.1, the action corresponds to the following is selected.

\[
\min\{\sum_{p_j \in P_1, e_t \in p_t} c_h^+, \sum_{p_j \in P_1} c_h^-\}.
\]
In this case, i.e. in the case that there is no path in $P_1$ contains a path in $P_2$, the algorithm terminates, if the following conditions hold.

C1. An adjacent edge to $x$ in $P_2$ reaches its upper bound, on the other word there exists a saturated path in $P_2$.

C2. There exists a path $p_r \in P_1$, that all nonadjacent edges to $x$ of $p_r$ reach their lower bounds and the length of edge in $p_r \cap \text{Adj}$ reaches its lower bound or $Lm$. On the other word, let $e_a \in p_r$ be the adjacent edges to $x$, then for each edge $e_t \in p_r$, $e_t \neq e_a$, $\hat{l}_t = \underline{l}_t$ and $\underline{l}_a = \max\{\underline{l}_2, Lm\}$.

Second, if we reach the case that a path in $P_1$ contains a path in $P_2$ then in the current iteration and all the next iterations just reduction is considered. In this case the algorithm terminates if condition (C2) holds.

These ideas lead us to the following algorithm.

**Algorithm [ISFBDB].**

**Input:** The tree $T = (E, V)$ with the edge length bounds, the vertex $x$ as the location of the facility. Also, the cost of increasing and decreasing of edges lengths are given.

**Output:** The new lengths of edges $\hat{l}_i$.

**Initialization:**

- **Find** the sets of all longest paths and shortest paths from $x$ and call them $P_1$ and $P_2$, respectively.
- **Find** the first and second farthest vertices to $x$ and call their distances to $x$, $d_{max1}$ and $d_{max2}$, respectively.
- **Find** the first and second closest vertices to $x$ and call their distances to $x$, $d_{min1}$ and $d_{min2}$, respectively.

Let

$$\text{Adj} = \{ e_i \in E | e_i \text{ is an adjacent of } x \}.$$ 

Set $f_1 := 0$, $f_2 := d_{max1} - d_{min1}$, $Lm := d_{min1}$, $n_1 := |P_1|$ and $n_2 := |P_2|$.

**Iteration step**

1. **If** there exists a saturated path in $P_1$ and a saturated path in $P_2$ then stop, the current length of edges are optimal.

2. **If** there exists a saturated path in $P_2$ and condition (C2) holds then stop, the current length of edges are optimal.

3. **If** there exists a saturated path in $P_1$ then

   (a) Set $C_{P_2} := \sum_{p_i \in P_2, e_t \in p_i} c_t^+$.

   (b) Compute $K_2$ using relation (12).

   (c) For any edge $e_t$ on each path $p_i \in P_2$, set $l_t := l_t + K_2$.

   (d) Set $f_1 := f_1 + C_{P_2} K_2$.

   (e) Set $f_2 := f_2 - K_2$.

   (f) Set $d_{min1} = d_{min1} + K_2$.

else

   (a) **If** there exists a path in $P_1$ which contains a path in $P_2$ then

      (i) If condition (C2) holds then stop, the current length of edges are optimal.

      (ii) For each $p_j \in P_1$, $j = 1, ..., n_1$, set

      $$h_j := \text{Argmin}\{c_h^{-} | e_h \in p_j, l_h > \underline{l}_h\},$$
Consider again the example. In the first iteration there isn’t any saturated path and the path \( p_1 \) does not contain \( p_2 \). Therefore, in this iteration we should calculate both \( C_{P_1} \) and

\[
C_{P_1} := \sum_{p_j \in P_1} c_{h_j} 
\]

(iii) Set \( Lm := d_{min1} \) and \( K_1 := \min\{d_r - d_{r-1}, RL_1, ..., RL_{n_1}\} \), where for \( j = 1, ..., n_1 \), \( RL_j \) are defined in relation (10).

(iv) For \( j = 1, ..., n_1 \), set \( l_{h_j} := l_{h_j} - K_1 \).

(v) Set \( f_1 := f_1 + C_{P_1} K_1 \).

(vi) Set \( f_2 := f_2 - K_1 \).

(vii) Set \( d_{max1} = d_{max1} - K_1 \).

else

(i) For each \( p_j \in P_1 \), \( j = 1, ..., n_1 \), set \( h_j := \text{Argmin}\{c_{h_j} e_j \in p_j, l_h > l_{h_j}\} \).

(ii) Set \( C_{P_1} := \sum_{p_j \in P_1} c_{h_j} \) and \( C_{P_2} := \sum_{p_j \in P_2} c_{h_j} \).

(iii) If \( C_{P_1} < C_{P_2} \) then

(A) Set \( K_1 := \min\{d_r - d_{r-1}, RL_1, ..., RL_{n_1}\} \), where for \( j = 1, ..., n_1 \), \( RL_j \) are defined in relation (10).

(B) For \( j = 1, ..., n_1 \), set \( l_{h_j} := l_{h_j} - K_1 \).

(C) Set \( f_1 := f_1 + C_{P_1} K_1 \).

(D) Set \( f_2 := f_2 - K_1 \).

(E) Set \( d_{max1} = d_{max1} - K_1 \).

else

(A) Compute \( K_2 \) using relation (12).

(B) For any edge \( e_t \) on each path \( p_t \in P_2 \), set \( l_t := l_t + K_2 \).

(C) Set \( f_1 := f_1 + C_{P_1} K_2 \).

(D) Set \( f_2 := f_2 - K_2 \).

(E) Set \( d_{min1} = d_{min1} + K_2 \).

(b) If \( f_2 = 0 \) stop, the optimal solution is achieved.

else

(i) Find \( P_1 \) and \( P_2 \) which are the sets of longest and shortest paths from \( x \), respectively.

(ii) Find \( d_{min2} \) and \( d_{max2} \) which are the second closest and farthest vertices to \( x \), respectively.

(iii) Go to Step 1.

End of algorithm

To calculate the time complexity of ISFBDB algorithm, note that the longest and shortest paths can be computed in \( O(n) \) time (see e.g. [17, 7]) and each step of the algorithm needs at most \( O(n) \) time. In each iteration either \( f_2 \) reduces from \( d_r \) to \( d_{r-1} \) or an edge reaches to its lower or upper bound. Therefore, the iteration step repeats at most \( O(n) \) times. Thus the time complexity of algorithm is \( O(n^2) \).

**Theorem 4.3.** The inverse single facility location problem with balancing on distances in the edge length bounded case on a tree can be solved in \( O(n^2) \) time.

**Example 2.** Consider again the tree \( T \) in Example 1, where the bounds of its edge lengths are given in Table 2.

The distances of farthest and closest vertices to \( x \) are 15 and 1, respectively. Thus, \( d_{max1} = 15 \), \( d_{min1} = 1 \) and \( f_2 = 14 \). Also \( P_1 = \{p_1\} \) and \( P_2 = \{p_2\} \) where \( p_1 = \{e_3, e_4, e_5\} \) and \( p_2 = \{e_6\} \). Table 3 contains the iterations result for this example. In the first iteration there isn’t any saturated path and the path \( p_1 \) does not contain \( p_2 \). Therefore, in this iteration we should calculate both \( C_{P_1} \) and
Among edges in $p_1$ the edge $e_1$ has lowest changing cost, So $h_1 = 3$ and $RL_1 = min\{l_3 - d_{min}, l_3 - l_3\} = 4$. Then we obtain $CP_1 = c_3$, $CP_2 = c_6^+$. Since $CP_1 = 2 < CP_2 = 5$, then we set $K_1 = min\{15 - 10, 4\} = 4$, $l_3 = 1$, $f_1 = 8$ and $f_2 = 10$.

| Iteration | $P_1$       | $P_2$       | $CP_1$ | $CP_2$ | $f_1$ | $f_2$ |
|-----------|-------------|-------------|--------|--------|-------|-------|
| 1         | $\{p_1\}$  | $\{p_2\}$  | $c_3 = 2$ | $c_6^+ = 5$ | 8     | 10    |
| 2         | $\{p_1\}$  | $\{p_2, p_3\}$ | $c_4 = 3$ | -        | 11    | 9     |
| 3         | $\{p_1, p_4\}$ | $\{p_2, p_3\}$ | $c_5^+ + c_8 = 6$ | -        | 23    | 7     |
| 4         | $\{p_1, p_4\}$ | $\{p_2, p_3\}$ | $c_5^+ + c_8 = 16$ | -        | 39    | 5     |

Table 3. The iterations result for Example 2.

In the second iteration the path $p_3 = \{e_3\}$ has been added to $P_2$. This path is a saturated path, however the condition $(C_2)$ does not hold. Since $p_1$ contains $p_3$, then we should only find $CP_1$. We see $h_1 = 4$, therefore $l_4$ should be reduced. Thus $CP_1 = c_4 = 3$ and $K_1 = min\{11 - 10, 2 - 1\} = 1$. Therefore, we will find $l_4 = 1$, $f_1 = 11$ and $f_2 = 9$.

In the third iteration the path $p_4 = \{e_8, e_9, e_{10}\}$ has been added to $P_1$. Again the path $p_3$ is a saturated path, but the condition $(C_2)$ does not hold. Also, again $p_1$ contains $p_3$, and we should only find $CP_1$. We see $h_1 = 5$ and $h_4 = 8$, therefore $l_5$ and $l_8$ should be reduced. Thus $RL_1 = 4$, $RL_4 = 2$, $CP_1 = c_5^+ + c_8^+ = 6$, $K_1 = min\{10 - 7, 4, 2\} = 2$, $l_3 = 5$, $l_8 = 1$, $f_1 = 23$ and $f_2 = 7$.

In the 4th iteration we will obtain $l_5 = 4$, $l_9 = 2$, $f_1 = 39$ and $f_2 = 5$. Then $p_1 = \{e_3, e_4, e_5\}$ is saturated and the current solution is optimal. The length of edges in the optimal solution are given in Table 4.

5. **Summary and conclusion.** In this paper, we investigated the inverse balanced distances of single facility location problem with variable edge lengths on trees. The goal in this problem is minimizing the difference distances between farthest and closest clients to the given facility by modifying the length of edges with minimum cost. A model and some properties are proposed for this problem. Then two algorithms with $O(n \log n)$ and $O(n^2)$ time complexity are developed for solving the unbounded and bounded edge lengths models, respectively.

The reverse model of the problem, in which we have a limited budgeting and we want to modify the length of edges such that the difference distances between farthest and closest clients to the given facility reduced as much as possible, can be
Table 4. The optimal edge lengths for Example 2.

| $e_i$          | optimal $l_i$ | $t_3$ | $t_4$ |
|----------------|---------------|-------|-------|
| $e_1 = (v_1, v_2)$ | 3             | 1     | 4     |
| $e_2 = (v_2, x)$   | 3             | 2     | 6     |
| $e_3 = (x, v_3)$   | 1             | 1     | 7     |
| $e_4 = (v_3, v_4)$ | 1             | 1     | 5     |
| $e_5 = (v_4, v_5)$ | 4             | 4     | 10    |
| $e_6 = (x, v_6)$   | 1             | 1     | 4     |
| $e_7 = (v_6, v_7)$ | 2             | 1     | 5     |
| $e_8 = (x, v_8)$   | 1             | 1     | 7     |
| $e_9 = (v_8, v_9)$ | 2             | 2     | 8     |
| $e_{10} = (v_9, v_{10})$ | 3       | 1    | 6     |

considered in the future works. Also, considering the other cases of inverse models are interesting fields for future works.

REFERENCES

[1] B. Alizadeh, E. Afrashteh and F. Baroughi, Combinatorial algorithms for some variants of inverse obnoxious median location problem on tree networks, Journal of Optimization Theory and Applications, 178 (2018), 914–934.

[2] B. Alizadeh, R. E. Burkard and U. Pferschy, Inverse 1-center location problems with edge length augmentation on trees, Computing, 86 (2009), 331–343.

[3] B. Alizadeh and R. E. Burkard, Combinatorial algorithms for inverse absolute and vertex 1-center location problems on trees, Networks, 58 (2011), 190–200.

[4] B. Alizadeh and R. Etemad, Optimal algorithms for inverse vertex obnoxious center location problems on graphs, Theoretical Computer Science, 707 (2018), 36–45.

[5] M. Barbati and C. Piccolo, Equality measures properties for location problems, Optimization Letters, 10 (2015), 903–920.

[6] O. Berman, Z. Drezner, A. Tamir and G. O. Wesolowsky, Optimal location with equitable loads, Annals of Operations Research, 167 (2009), 307–325.

[7] R. W. Bulterman, F. W. Van-Der-Sommen, G. Zwaan T. Verhoeff, A. J. M. Van-Gasteren and W. H. J. Feijen, On computing a longest path in a tree, Information Processing Letters, 81 (2002), 93–96.

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

[9] R. E. Burkard, C. Pleschiutschnig and J. Z. Zhang, The inverse 1-median problem on a cycle, Discrete Optimization, 5 (2008), 242–253.

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

[11] M. Davoodi, k-Balanced Center Location problem: A new multi-objective facility location problem, Computers & Operations Research, 105 (2019), 68–84.

[12] H. A. Eiselt and G. Laporte, Objectives in Location Problems, In: Facility Location: A Survey of Applications and Methods (eds. Z. Drezner), Springer, Berlin, (1995), 151–180.

[13] J. Fathali and M. Zaferanieh, On the balanced 2-median and 2-maxian problems on a tree, arXiv preprint, arXiv:1803.10332, 2018.

[14] M. Galavii, The inverse 1-median problem on a tree and on a path, Electronic Notes in Discrete Mathematics, 36 (2010), 1241–1248.

[15] M. Gavalec and O. Hudec, Balanced location on a graph, Optimization, 35 (1995), 367–372.

[16] X. C. Guan and B. W. Zhang, Inverse 1-median problem on trees under weighted Hamming distance, Journal of Global Optimization, 54 (2012), 75–82.

[17] G. Y. Handler, Minimax location of a facility in an undirected tree networks, Transportation Sci., 7 (1973), 287–293.

[18] C. Heuberger, Inverse combinatorial optimization: a survey on problems, methods, and results, J. Comb. Optim., 8 (2004), 329–361.
[19] M. Landete and A. Marin, Looking for edge-equitable spanning trees, *Computers & Operations Research*, 41 (2014), 44–52.
[20] M. A. Lejeune and S. Y. Prasad, Effectiveness-equity models for facility location problems on tree networks, *Networks*, 62 (2013), 243–254.
[21] A. Marin, The discrete facility location problem with balanced allocation of customers, *European Journal of Operational Research*, 210 (2011), 27–38.
[22] M. T. Marsh and D. A. Schilling, Equity measurement in facility location analysis: a review and framework, *European Journal of Operational Research*, 74 (1994), 1–17.
[23] P. B. Mirchandani and R. Francis, *Discrete Location Theory*, J. Wiley, 1990.
[24] M. Nazari, J. Fathali, M. Nazari and S. M. Varedi-Koulaei, Inverse of backup 2-median problems with variable edge lengths and vertex weight on trees and variable coordinates on the plane, *Production and Operations Management*, 9 (2018), 115–137.
[25] K. T. Nguyen, Inverse 1-median problem on block graphs with variable vertex weights, *Journal of Optimization Theory and Applications*, 168 (2016), 944–957.
[26] K. T. Nguyen and A. R. Sepasian, The inverse 1-center problem on trees with variable edge lengths under Chebyshev norm and Hamming distance, *Journal of Combinatorial Optimization*, 32 (2016), 872–884.
[27] S. Omidi, J. Fathali and M. Nazari, Inverse and reverse balanced facility location problems with variable edge lengths on trees, *OPSEARCH*, 57 (2020), 261–273.
[28] V. H. Pham, K. T. Nguyen and T.T. Le, A linear time algorithm for balance vertices on trees, *Discrete Optimization*, 32 (2018), 37–42.
[29] T. Sayar, J. Fathali and M. Ghiyasi, The problem of balancing allocation with regard to the efficiency of servers, *Iranian Journal of Operations Research*, 9 (2018), 29–47.
[30] A. R. Sepasian and F. Rahbarnia, An O(nlogn) algorithm for the inverse 1-median problem on trees with variable vertex weights and edge reductions, *Optimization*, 64 (2015), 595–602.

Received June 2020; revised November 2020.

E-mail address: shahedeo@yahoo.com
E-mail address: fathali@shahroodut.ac.ir