COMPUTATION OF INVERSE 1-CENTER LOCATION PROBLEM ON THE WEIGHTED TRAPEZOID GRAPHS

BISWANATH JANA, SUKUMAR MONDAL, AND MADHUMANGAL PAL

ABSTRACT. Let T_{TRP} be the tree corresponding to the weighted trapezoid graph $G = (V, E)$. The eccentricity $e(v)$ of the vertex v is defined as the sum of the weights of the vertices from v to the vertex farthest from $v \in T_{TRP}$. A vertex with minimum eccentricity in the tree T_{TRP} is called the *1-center* of that tree. In an inverse 1-center location problem, the parameter of the tree T_{TRP} corresponding to the weighted trapezoid graph $G = (V, E)$, like vertex weights, have to be modified at minimum total cost such that a pre-specified vertex $s \in V$ becomes the 1-center of the trapezoid graph G. In this paper, we present an optimal algorithm to find an inverse 1-center location on the weighted tree T_{TRP} corresponding to the weighted trapezoid graph $G = (V, E)$, where the vertex weights can be changed within certain bounds. The time complexity of our proposed algorithm is $O(n)$, where n is the number of vertices of the trapezoid graph G.

1. INTRODUCTION

1.1. **Trapezoid Graph.** A *trapezoid graph* can be represented in terms of a *trapezoid diagram*. A *trapezoid diagram* consist of two horizontal parallel lines, named as the top line and the bottom line. Each line contains n intervals. The left end point and right end point of an interval i are a_i and b_i ($\geq a_i$) on the top line and c_i and d_i ($\geq c_i$) on the bottom line. A *trapezoid* i is defined by four corner points $[a_i, b_i, c_i, d_i]$ in the trapezoid diagram. Let $T = \{1, 2, \ldots, n\}$ be the set of n trapezoids. Let $G = (V, E)$ be an undirected graph with n vertices and m edges and let $V = \{1, 2, \ldots, n\}$. G is said to be a *trapezoid graph* if it can be represented by a trapezoid diagram such that each trapezoid corresponds to a vertex in V and $(i, j) \in E$ if and only if the trapezoids i and j intersect in the trapezoid diagram [12]. Two trapezoids i and $j(> i)$ intersect if and only if either $(a_j - b_i) < 0$ or $(c_j - d_i) < 0$ or both. We assume that the graph $G = (V, E)$ is connected. Without loss of generality we make the following assumptions.

(a) A trapezoid contains four different corner points and no two trapezoids share a common end point.

(b) Trapezoids in the trapezoid diagram and vertices in the trapezoid graph are one and the same.
(c) The trapezoids in the trapezoid diagram T are indexed by increasing right end points on the top line, i.e., if $b_1 < b_2 < \cdots < b_n$ then the trapezoids are indexed by $1, 2, 3, \ldots, n$, respectively.

Figure 2 represents a trapezoid graph shown in Figure 1. The class of trapezoid graphs includes two well-known classes of intersection graphs: the permutation graphs and the interval graphs [19]. The permutation graphs are obtained where $a_i = b_i$ and $c_i = d_i$ for all i and the interval graphs are obtained where $a_i = c_i$ and $b_i = d_i$ for all i. Trapezoid graphs can be recognized in $O(n^2)$ time [30]. Trapezoid graphs were first studied in [11, 12]. These graphs are a superclass of interval graphs, permutation graphs and a subclass of co-comparability graphs [29].

In G, a *walk* is defined as a finite alternating sequence of vertices and edges, beginning and ending with vertices, such that each edge is incident with the vertices preceding and following it. No edge appears more than once in a walk. A vertex; however, may appear more than once. An open walk, in which no vertex appears more than once, is called a *path*. A closed walk, in which no vertex (except the initial and the final vertex) appears more than once, is called a *circuit*. A *tree* T is a connected graph without any circuits, i.e., a tree is a connected acyclic graph. Clearly, there is one and only one path between every pair of vertices of T. A tree T is *weighted* if there is a non-negative real number associated with each edge or vertex of T. In an un-weighted tree $T = (V, E)$, where $|E| = |V| - 1$, the *eccentricity e(v)* of the vertex v is defined as the distance from v to the vertex farthest from $v \in T$, i.e.,

$$e(v) = \max\{d(v, v_i), \text{ for all } v_i \in T\},$$

where $d(v, v_i)$ is the number of the edges on the shortest path between v and v_i.

In vertex weighted tree $T = (V, E)$, the *eccentricity e(v)* of the vertex v is defined as the sum of the weights of the vertices from v to the vertex farthest from $v \in T$, i.e.,

$$e(v) = \max\{d_w(v, v_i), \text{ for all } v_i \in T\},$$

where $d_w(v, v_i)$ is the sum of the weights of the vertices on the path between v and v_i.

A vertex with minimum eccentricity in the tree T is called a *center*, i.e., if $e(s) = \min\{e(v), \text{ for all } v \in V\}$, then s is the *1-center*. It is clear that every tree has either one or two centers. The 1-center location problems occur when the best location of an emergency service, a hospital, a fire station, a

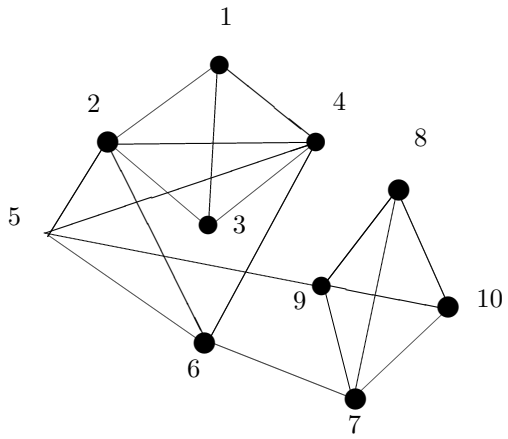

Figure 1. A Trapezoid graph G.

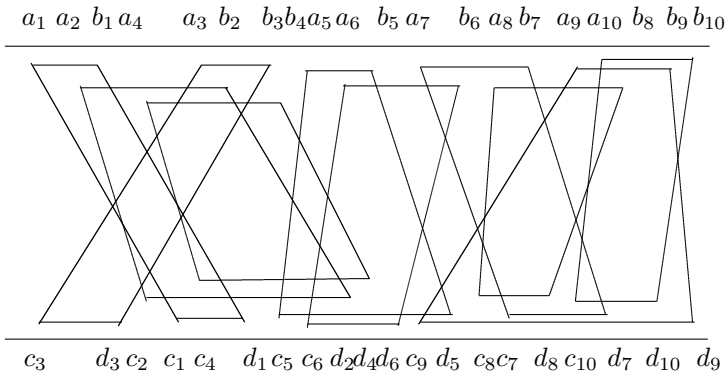

Figure 2. A trapezoid diagram T of the graph G of Figure 1.

police office, a bank branch, a train station, an airport, a shopping center, a city park, or another facility center has to be found.

The eccentricity of a center in a tree is defined as the *radius* of the tree and is denoted by $\rho(T)$, i.e.,

$$\rho(T) = \{\min_{v \in T} e(v)\}.$$

The *diameter* of an unweighted tree T is defined as the length of the longest path in T, i.e., the maximum eccentricity is the diameter. The *diameter* of a weighted tree T is defined as the sum of the weights of the vertices in the path of T, i.e., the maximum eccentricity of the weighted tree T is the diameter.

For the weighted tree T with n vertices and $n-1$ edges of the corresponding weighted trapezoid graphs, the inverse 1-center problem is concerned with a modifying parameter, like vertex weight, at minimum total cost within certain modification bounds such that, a pre-specified vertex becomes the 1-center. For example, consider the train station of a city which cannot be relocated. The mayor could change some parameters in the urban system (e.g., improving streets or urban transportation lines) at minimum cost, subject to evident length constraints such that the current location of the train station becomes the center (in a graph theoretic sense) of the city.

Our problem is to design an optimal algorithm to compute the inverse 1-center location problem on weighted trapezoid graphs.

1.2. **Survey of Relevant Literature.** In [9, 40, 41], we can work out the inverse problem of many combinatorial or network optimization in a strong or weak way of polynomial algorithms. In fact [38] shows a large class of combinatorial or network optimization problems, if we can solve the original problem in polynomial times, then its inverse problem can be solved in polynomial time by a uniform methodology. A detailed survey on inverse optimization problems has been compiled by Heuberger [25]. In the context of location problems, Cai et al. [10] proved that the inverse 1-center location problem with edge length modification on general unweighted directed graphs is NP-hard, while the underlying center location problem is solvable in polynomial time. In 2004, Burkard et al. [5] considered inverse p-median location problems could 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. Galavi [21] showed later that this problem can actually be solved in $O(n)$ time. Moreover, Burkard 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, [6], 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 [20] 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 et al. [7, 8].

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 pre-specified point that should become a 1-median, does not coincide with a given point in the plane. Galavii [21] showed that the 1-median on a

path with positve/negative weights lies in one of the vertices with positive weights, or lies in one of the end points of the path. This property allows us to solve the inverse 1-median problem on a path with negative weights in $O(n)$ time. Gassner [22] considered an inverse version of the convex ordered median problem and showed that this problem is NP-hard on general graphs, even on trees. Further, it was shown that the problem remained NP-hard for unit weights if the underlying problem was a K-centrum problem; if not 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 [39] 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. Recently, Alizadeh et al. [1] designed an algorithm for inverse 1-center location problem with edge length augmentation on trees in $O(n \log n)$ time, using a set of suitably extended AVL-search trees. In [2], Alizadeh et al. designed a combinatorial algorithm for inverse absolute on trees in $O(n^2)$ time, when topology is not allowed, and $O(n^2r)$ time, when topology is allowed.

Inverse optimization problems have recently attained significant theoretical interest due to their relevance in practice. For a comprehensive survey on inverse optimization problems, see [16, 23, 25, 28].

Network location problems belong to basic optimization models which are concerned with finding the best location of single or multiple new facilities in a network of demand points, such that a given function, which depends on the distance between the facilities and clients, becomes minimum. Depending on the model under investigation, facilities or clients may either be placed only at vertices, or may also lie on edges of the network. For further details on these problems, the reader is referred to the books of Daskin [13], Drezner et al. [15], Francis et al. [18], Mirchandani et al. [32], and Nickel et al. [35].

Burton and Toint [4] first investigated an inverse shortest paths problem in 1992. Since then, many problems have been considered by various authors, working at least partly independently. The notation of 'inverse optimization' is always similar, but not the same. Recently, Jana et al. [26, 27] designed a linear time algorithm to compute an inverse 1-center location problem on the edge weighted trees and weighted interval graphs, respectively.

In this paper, we design an algorithm to compute an inverse 1-center location problem on weighted trapezoid graphs in $O(n)$ time, where n is the number of vertices of the graph.

1.3. **Applications of the Problem.** One application derives from geophysical sciences and concerns predicting the movement of earthquakes.

Geologic zones are discretized into a number of cells with a view to achieving this. Adjacency relations can be modeled by arcs in a corresponding network (Moser [34]). Precise values are hard to obtain in comparison to some estimates of the transmission times being known. Earthquakes travel along the shortest paths and the problem is to refine the estimates of the transmission times between the cells on the basis of observation assumption of an earthquake, and the arrival times of the resulting seismic perturbations at various points. Actually, this is the inverse shortest path problem.

Another possible application actually changes the real costs. Assume that we are given a road network and some facility in it. The aim is to place the facility in such a way that the maximum distance to the customers is minimal. However, we are often faced with the situation that the facility already exists and cannot be relocated with reasonable costs. In such a situation, we may want to modify the network as little as possible (improving roads costs), such that the location of the facility becomes optimum (or such that the distances to the customers do not exceed some given bounds). This is an example of the inverse center location problem. When modeling traffic networks, a further option is to impose tolls in order to enforce an efficient use of the network (Dial [14]). The choice of the word 'inverse optimization' was motivated in part by the widespread use of inverse methods in other fields, see Marlow [31] and Engl et al. [17].

1.4. **Organization of the Paper.** In Section 2, we present the data structure and construction of the tree T_{TRP}. In Section 3, we discuss the inverse 1-center. Some notations are also presented in this section. In Section 4, we present an algorithm to get inverse 1-center of the modified vertex weighted tree corresponding to the trapezoid graph G. The time complexity is also calculated in this section. In Section 5, we give a conclusion.

2. Construction of the Tree

Let i be a pre-specified vertex which is to be an inverse 1-center. In this section our aim is to construct a minimum heighted tree, as root i, with two branches of level difference either zero or one.

Let the vertex i be the root of the tree. Then we find all adjacent vertices to i corresponding to the trapezoid and set them as child (leaves) of i. Next consider the vertices k and j, where $k = \max\{b_k \ \textit{or} \ d_k : (k,i) \in E\}$, $j = \max\{b_j \ \textit{or} \ d_j : (j,i) \in E, \ k \neq j \ \textit{and} \ b_j < b_k \ \textit{or} \ d_j < d_k\}$, and set them as a vertices on the main path and mark them. Next, find all adjacent trapezoids to the vertices k and j and set them as respective child (leaves). This process continues until all trapezoids are marked. In this way, we construct a rooted tree with two branches with level difference either zero or one.

The proposed combinatorial algorithm to construct the tree T_{TRP} is as follows:

Algorithm TRP-TREE

Input: Weighted trapezoid graph G with four corner points $[a_i, b_i, c_i, d_i]$, $i = 1, 2, \ldots, n$, and $T = 1, 2, \ldots, n$ be the set of n trapezoids.

Output: The rooted tree T_{TRP} with two branches of the trapezoid graph G.

Step 1. Set root $= i$ and compute $N(i) =$ the open neighborhood of $i = \{v : (v, i) \in E\}$.

Step 2. If $|N(i)| = 1$, then end.

If $|N(i)| > 1$ and i is the starting trapezoid, i.e., $i = 1$, then go to Step 3.

If $|N(i)| > 1$ and i is the end trapezoid, i.e., $i = n$, then go to Step 4.

If $|N(i)| > 1$ and i is an trapezoid between 1 and n, i.e., $1 < i < n$, then go to Step 5.

Step 3. Set $N(i)$ as the child of the root i and mark them.

Step 3.1. Set $k = \max\{b_k \ or \ d_k : (k, i) \in E\}$, $j = \max\{b_j \ or \ d_j : (j, i) \in E, \ k \neq j \ and b_j < b_k \ or \ d_j < d_k\}$.

Step 3.2. Find unmarked adjacent of j and k and if $N(j) \bigcap N(k) = \phi$, then $m_1 = \max\{b_{m_1} \ or \ d_{m_1} : (m_1, k) \in E, m_1 \in N(k)\}$ and set all unmarked $N(k)$ as the child of k and mark them and $m_2 = \max\{b_{m_2} \ or \ d_{m_2} : (m_2, j) \in E, m_2 \in N(j)\}$ and set all unmarked $N(j)$ as the child of j and mark them.

else $m_1' = \max\{b_{m_1'} \ or \ d_{m_1'} \in N(k) \cap N(j)\}$ set as child of j and $\{N(k) \cup N(j) - \{m_1'\}\}$ as child of k and mark and find $m_1'' = \max\{N(k) \cup N(j) - \{m_1'\}\}$.

Step 3.3. This process is continued until all trapezoids are marked.

Step 3.4. Compute the trapezoid tree T_{TRP}.

Step 4. Set $N(i)$ as the child of the root i and mark them.

Step 4.1. Set $j' = \min\{a_{j'} \ or \ c_{j'} : (j', i) \in E\}$, $k' = \min\{a_{k'} \ or \ c_{k'} : (k', i) \in E, k' \neq j' \ and \ a_j' < a_k' \ or \ c_j' < c_k'\}$.

Step 4.2. Find unmarked adjacent of j' and k' and if $N(j') \bigcap N(k') = \phi$, then $r_1 = \min\{a_{r_1} \ or \ c_{r_1} : (r_1, j') \in E, r_1 \in N(j')\}$ and set all unmarked $N(j')$ as the child of j' and mark them and $r_2 = \min\{a_{r_2} \ or \ c_{r_2} : (r_2, k') \in E, r_2 \in N(k')\}$ and set all unmarked $N(k')$ as the child of k' and mark them.

else $r_1' = \min\{a_{r_1'} \ or \ c_{r_1'} : r_1' \in N('k) \cap N(j')\}$ set as child of j' and $\{N(k') \cup N(j') - \{r_1'\}\}$ as child of k' and mark and find $r_1'' = \min\{N(k') \cup N(j') - \{r_1'\}\}$.

Step 4.3. This process is continued until all trapezoids are marked.

Step 4.4. Compute the trapezoid tree T_{TRP}.

Step 5. Set $N(i)$ as the child of the root i and mark them.

Step 5.1. Set $p = \max\{b_p \ or \ d_p : (p, i) \in E\}$, $q = \min\{a_q \ or \ c_q : (q, i) \in$

$E\}$ and $p \neq q$.
Step 5.2. Set $p' = \max\{b_{p'} \ or \ d_{p'} : (p', p) \in E, p' \in N(p)\}$ and set all unmarked $N(p)$ as the child of p and marked.
Step 5.3. Set $q' = \min\{a_{q'} \ or \ c_{q'} : (q', q) \in E, q' \in N(q)\}$ and set all unmarked $N(q)$ as the child of q and marked.
Step 5.4. This process is continued until all trapezoids are marked.
Step 5.5. Compute the interval tree T_{TRP}.
Step 6. Put weight $w_j(> 0)$ to the vertex j in T_{TRP} corresponding to the trapezoid j of the trapezoid graph G.
end TRP-TREE.

Illustration of the Algorithm TRP-TREE. Let $i = 1$ be the prespecified vertex which is the root whose level is 0. Next, the open neighborhood of 1 is $N(1) = \{2, 3, 4\}$, where the vertices of $N(1)$ as the child of the root 1 and put them at level 1. Next, 4 has the maximum b_i among the trapezoids of $N(1)$ corresponding to the vertices of the graph G and 2 has the next maximum d_i among the trapezoids of $N(1)$ corresponding to the vertices of the graph G. Next, the open neighborhoods of 4 and 2 are $N(4) = \{5, 6\}$ and $N(2) = \{5, 6\}$, respectively, where the vertices of $N(4)$ and $N(2)$ as the child of the roots 4 and 2 and put them at level 2. Next, 6 has the maximum b_i among the trapezoids of $N(4)$ corresponding to the vertices of the graph G and 5 has the next maximum b_i among the trapezoids of $N(2)$ corresponding to the vertices of the graph G. Next, the open neighborhoods of 6 and 5 are $N(6) = \{7\}$ and $N(5) = \{9\}$, respectively, where the vertices of $N(6)$ and $N(5)$ as the child of the roots 6 and 5 and put them at level 3. Next 7 has the maximum d_i among the trapezoids of $N(6)$ corresponding to the vertices of the graph G and 9 has the maximum d_i among the trapezoids of $N(5)$ corresponding to the vertices of the graph G. Next the open neighborhoods of 7 and 9 are $N(7) = \{8, 10\}$ and $N(9) = \{8, 10\}$, respectively, where the vertices of $N(7)$ and $N(9)$ as the child of the roots 7 and 9 and put them at level 4. Finally we construct the rooted tree T_{TRP} with root $i = 1$ (see Figure 3).

Now we have the following important observation on T_{TRP}.

Lemma 2.1. *The tree T_{TRP} formed by Algorithm TRP-TREE is a spanning tree.*

Proof. As per the construction of the graph T_{TRP} by maximum b_i or d_i, $i = 1, 2, \ldots, n$, in trapezoid diagram we get n vertices and $n - 1$ edges. Also, there is no repetition of the vertices, as we search only unmarked vertices, so this is a graph without any circuit. Therefore, the tree T_{TRP} is a spanning tree. Hence, we have the result. □

Lemma 2.2. *The tree T_{TRP} formed by Algorithm TRP-TREE is a BFS tree with minimum height.*

Proof. Actually, steps of the algorithm indicate the steps of BFS technique in the trapezoid graph. Thus, the tree formed by Algorithm TRP-TREE is the BFS tree. Again, we traverse the trapezoid graph with respect to maximum b_i or d_i until all unmarked trapezoids are marked. As in each step we move on the trapezoid, its height to be minimum. □

Also, the time complexity of Algorithm TRP-TREE to compute the tree T_{TRP} is given below.

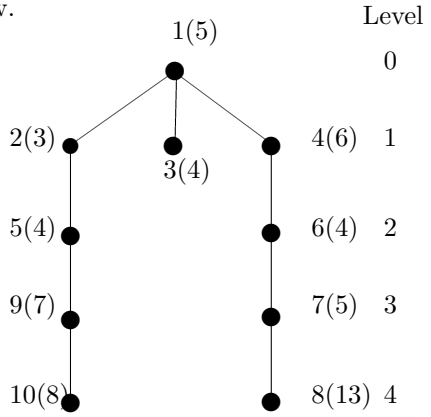

Figure 3. Tree T_{TRP} of the trapezoid graph G.

Theorem 2.3. *The time complexity of Algorithm TRP-TREE is $O(n)$, where n is the number of vertices of the tree.*

Proof. Step 1 and Step 2 each take $O(n)$ time, since the arcs are sorted and the root is selected from n arcs. Step 3 can be computed in $O(n)$ time, since the number of arcs is n. Since the end points of the arcs are sorted, the maximum element (vertex) from a set of vertices can be computed in $O(n)$ time. Again, the intersection of two finite sets of n elements (number of vertices) can be executed in $O(n)$ time. Thus, Step 4 and Step 5 can be computed in $O(n)$ time. Since the weight of each vertex in tree T_{TRP} corresponds to the weight of the trapezoids, then the trapezoid graph is placed on the corresponding vertex, so Step 6 can be executed in $O(n)$ time. Hence, overall time complexity of our proposed Algorithm TRP-TREE is $O(n)$ time, where n is the number of vertices of the weighted trapezoid graph. □

Thus, the tree T_{TRP} of the trapezoid graph is formed. The tree T_{TRP} of the trapezoid graph G (Figure 1) is shown in Figure 3.

3. Inverse 1-center

In this section we discuss inverse 1-center.

Now before going to our proposed algorithm, we introduce some notations for our algorithmic purpose. Let i be the pre-specified vertex in G.

R_i	:	Longest path to the vertex i.
L_i	:	Another longest path to the vertex i.
$w(R_i)$	:	Sum of weights of the vertices except the vertex i of the path R_i.
$w(L_i)$	:	Sum of weights of the vertices except the vertex i of the path L_i.
$w_{low}(v)$	:	Minimum weight of the vertex in the graph G.
$w_{upp}(v)$	:	Maximum weight of the vertex in the graph G.
w_{min}	:	$min\{w(L_i), w(R_i)\}$.
w_{max}	:	$max\{w(L_i), w(R_i)\}$.
w_1	:	$min\{w(v), v \in G\}$.
w_2	:	$max\{w(v), v \in G\}$.
k_1	:	The number of vertices in such a path between L_i, R_i whose weight is maximum, except the vertex i.
k_2	:	The number of vertices in such a path between L_i, R_i whose weight is minimum, except the vertex i.
T_{TRP}	:	Weighted tree corresponding to the circular-arc graph G.
T'_{TRP}	:	Modified tree of the tree T_{TRP} corresponding to the trapezoid graph G.
$w^*(R_i)$	:	Sum of weights of the vertices except the vertex i of the path R_i after modification.
$w^*(L_i)$	:	Sum of weights of the vertices except the vertex i of the path L_i after modification.

To find the inverse 1-center, we discuss the following cases.

Case 1. If the sum of weights of one side of the vertex i is equal to the sum of weights of other side, i.e., $w(L_i) = w(R_i)$, then i is the center as well as the inverse 1-center of the graph.

Case 2. If $w(L_i) \neq w(R_i)$, then we have the following six cases.

Case 2.1. w_{min} is equal to the product of the number of vertices except the vertex i in the path whose weight is maximum and minimum weight of the vertex in the graph, i.e., $w_{min} = k_1 w_1$.

Case 2.2. w_{min} is greater than the product of the number of vertices except the vertex i in the path whose weight is maximum and minimum weight of the vertex in the graph, i.e., $w_{min} > k_1 w_1$.

Case 2.3. w_{min} is less than the product of the number of vertices except the vertex i in the path whose weight is maximum and minimum weight of the vertex in the graph, $w_{min} < k_1 w_1$.

Case 2.4. w_{max} is equal to the product of the number of vertices except the vertex i in the path whose weight is minimum and maximum weight of the vertex in the graph, $w_{max} = k_2 w_2$.

Case 2.5. w_{max} is greater than the product of the number of vertices except the vertex i in the path whose weight is minimum and maximum weight of the vertex in the graph, $w_{max} > k_2 w_2$.

Case 2.6. w_{max} is less than the product of the number of vertices except the vertex i in the path whose weight is minimum and maximum weight of the vertex in the graph, i.e., $w_{max} < k_2 w_2$.

Under the above conditions, we modify the tree T_{TRP} with the help of the following non-linear semi-infinite (or nonlinear) optimization model.

$$\text{Minimize} \sum_{v \in V(T_{TRP})} \{c^+(w(v))x(w(v)) + c^-(w(v))y(w(v))\}$$

subject to

$\max_{v \in V(T_{TRP})} d_{\overline{w}}(v, i) \leq \max_{v \in V(T_{TRP})} d_{\overline{w}}(v, p)$,
for all $p \in T_{TRP}$(or $p \in V(T_{TRP})$),
$\overline{w}(v) = w(v) + x\{w(v)\} - y\{w(v)\}$ for all $v \in V(T_{TRP})$,
$x\{w(v)\} \leq w^+\{w(v)\}$, for all $v \in V(T_{TRP})$,
$y\{w(v)\} \leq w^-\{w(v)\}$, for all $v \in V(T_{TRP})$,
$x\{w(v)\}, y\{w(v)\} \geq 0$, for all $v \in V(T_{TRP})$,

where $\overline{w}(v)$ is the modified vertex weight, $w^+\{w(v)\} = w_{upp}(v) - w(v)$, and $w^-\{w(v)\} = w(v) - w_{low}(v)$ are the maximum feasible amounts by which $w(v)$ can be increased and reduced, respectively, i.e., $w_{low}(v) \leq \overline{w}(v) \leq w_{upp}(v)$ and $x\{w(v)\}$ and $y\{w(v)\}$ are the maximum amounts by which the vertex weight $w(v)$ is increased and reduced, respectively, $c^+(w(v))$ is the non-negative cost if $w(v)$ is increased by one unit, and $c^-(w(v))$ is the non-negative cost if $w(v)$ is reduced by one unit. Every feasible solution (x, y) with $x = \{x(w(v)) : v \in V(T_{CIR})\}$ and $y = \{y(w(v)) : e \in V(T_{TRP})\}$ is also called a feasible modification of the inverse 1-center location problem.

Now we prove the following results.

Lemma 3.1. *If $w_{min} = k_1 w_1$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices except the vertex i, i.e., root i up to the minimum weight, maintaining the bounding condition in the path whose weight is maximum and i is the inverse 1-center.*

Proof. If k_1 is the number of vertices in the maximum weighted path L_i or R_i and w_1 is the minimum weight of the vertex among the vertices in T_{TRP} as well as L_i or R_i, then there is a scope to reduce the weight of each vertex up to w_1. As k_1 vertices exist in the path L_i or R_i, we can reduce the weight at least $k_1 w_1$ and hence, reduce the weight of the path L_i or R_i

to become k_1w_1. Again, we have $w_{min} = k_1w_1$. In this way, we can balance the weights of both paths. So we get the modified tree of the tree T_{TRP}, say T'_{TRP}. Again, since the trapezoid graph is arbitrary, our assumption is true for any trapezoid graphs.

Finally in T'_{TRP}, we have $w^*(L_i) = w^*(R_i)$, which implies that i is the inverse 1-center of the given weighted trapezoid graph. Hence, the result follows. □

Lemma 3.2. *If $w_{min} > k_1w_1$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of some vertices except the root i and maintaining the bounding condition in the path whose weight is maximum and i is the inverse 1-center.*

Proof. Since we can decrease the weight of each vertex except the root up to minimum weight of the vertex in T_{TRP}, we can reduce the weight in the path whose weight is maximum in such a way that the least weight of the path becomes k_1w_1. Again, we have $w_{min} > k_1w_1$. Therefore, we can decrease the weights $(w_{max} - w_{min})$ from the vertices except the root i in the path whose weight is maximum using the conditions of the non-linear semi-infinite optimization model technique. In this way, we can balance the weights of both paths. So we get the modified tree of the tree T_{TRP}, say T'_{TRP}. Again, since the trapezoid graph is arbitrary, our assumption is true for any trapezoid graphs.

Finally in T'_{TRP}, we have $w^*(L_i) = w^*(R_i)$, which implies that i is the inverse 1-center of the given weighted trapezoid graph. Hence, we have the result. □

Lemma 3.3. *If $w_{min} < k_1w_1$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices up to the minimum weight, except the root i maintaining the bounding condition in the path whose weight is maximum, and enhance the weights of some vertices except the root i in the path whose weight is minimum and i is the inverse 1-center.*

Proof. Since we can decrease the weight of each vertex up to the minimum weight of the vertex in T_{TRP}, we can reduce the weights of the vertices except the root in the path whose weight is maximum in such a way that the least weight of the path becomes k_1w_1. Again, we have $w_{min} < k_1w_1$. Therefore, we can increase the weights $(k_1w_1 - w_{min})$ to the vertices, except the root, in the path whose weight is a minimum using the conditions of the non-linear semi-infinite optimization model technique. In this way, we can balance the weights of both paths. So we get the modified tree of the tree T_{TRP}, say T'_{TRP}. Again, since the trapezoid graph is arbitrary, our assumption is true for any trapezoid graphs.

Finally in T'_{TRP}, we have $w^*(L_i) = w^*(R_i)$, which implies that i is the inverse 1-center of the given weighted trapezoid graph. Hence, we have the result. □

Lemma 3.4. *If $w_{max} = k_2w_2$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of all vertices up to a maximum weight, except the root i, maintaining the bounding condition in the path whose weight is minimum and i is the inverse 1-center.*

Proof. If k_2 is the number of vertices in the minimum weighted path L_i or R_i and w_2 is the maximum weight of the vertex among the vertices in T_{TRP} as well as L_i or R_i, then there is a scope to increase the weight of each vertex up to w_2. Since there are k_2 vertices in the path L_i or R_i, we can enhance the weight at most k_2w_2 and hence, the weight of the path L_i or R_i becomes k_2w_2. Again, we have $w_{max} = k_2w_2$. In this way, we can balance the weights of both paths. So we get the modified tree of the tree T_{TRP}, say T'_{TRP}. Again, since the trapezoid graph is arbitrary, our assumption is true for any trapezoid graphs.

Finally in T'_{TRP}, we have $w^*(L_i) = w^*(R_i)$, which implies that i is the inverse 1-center of the given weighted trapezoid graph. Hence, the result follows. □

Lemma 3.5. *If $w_{max} > k_2w_2$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of all vertices up to the maximum weight except root i which maintains the bounding condition in the path whose weight is minimum and reducing the weights of some vertices except the root i in the path whose weight is maximum and i is the inverse 1-center.*

Proof. Since we can increase the weight of each vertex up to the maximum weight of the vertex in T_{TRP}, we can enhance the weights of all vertices except the root i in the path whose weight is minimum in such a way that its greatest weight of the path becomes k_2w_2. Again, we have $w_{max} > k_2w_2$. Therefore, we can reduce the weights $(w_{max} - k_2w_2)$ to some vertices except the root in the path whose weight is maximum, using the conditions of the non-linear semi-infinite optimization model technique. In this way, we can balance the weights of both paths. So we get the modified tree of the tree T_{TRP}, say T'_{TRP}. Again, since the trapezoid graph is arbitrary, our assumption is true for any trapezoid graphs.

Finally in T'_{TRP}, we have $w^*(L_i) = w^*(R_i)$, which implies that i is the inverse 1-center of the given weighted trapezoid graph. Hence, we have the result. □

Lemma 3.6. *If $w_{max} < k_2w_2$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of some vertices except the root i by maintaining the*

bounding condition in the path whose weight is minimum and i is the inverse 1-center.

Proof. Since we can increase the weight of each vertex up to the maximum weight of the vertex in T_{TRP}, we can enhance the weights of the vertices, except the root i, in the path whose weight is minimum in such a way that its greatest weight of the path becomes k_2w_2. Again, we have $w_{max} < k_2w_2$. Therefore, we can increase the weights $(w_{max} - w_{min})$ to some vertices, except the root i, in the path whose weight is minimum using the conditions of the non-linear semi-infinite optimization model technique. In this way, we can balance the weights of both paths. So we get the modified tree of the tree T_{TRP}, say T'_{TRP}. Again, since the trapezoid graph is arbitrary, our assumption is true for any trapezoid graphs.

Finally in T'_{TRP}, we have $w^*(L_i) = w^*(R_i)$, which implies that i is the inverse 1-center of the given weighted trapezoid graph. Hence, the result follows. □

4. Algorithm and its complexity

In this section, we propose a combinatorial algorithm for the inverse 1-center location problem on the weighted tree T_{TRP}. The main idea of our proposed algorithm follows.

Let T_{TRP} be a weighted tree corresponding to the trapezoid graph G with n vertices and $(n-1)$ edges. Let V be the vertex set and E be the edge set. Let i be any non-pendant specified vertex in the tree T_{TRP} which is the inverse 1-center. First, we calculate the path whose weight is maximum from i to any pendant vertex of T_{TRP}. Let L and R be the left and right paths from i in which weights are maximum with respect to the sides. Let $w(L_i)$, $w(R_i)$ be the sum of weights of the vertices except the root of the paths L_i, R_i, respectively, with respect to the vertex i. If $w(L_i) = w(R_i)$, then i is the center as well as the inverse 1-center of the graph. If $w(L_i) \neq w(R_i)$, then six cases may arise. In the first case, if $w_{min} = k_1w_1$ in T_{TRP}, where $w_1 = min\{w(v), v \in G\}$, $w_{min} = min\{w(L_i), w(R_i)\}$, k_1 is the number of vertices in such path between L_i, R_i whose weight is maximum, except the root i, and $w_{min} > 0$, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices up to minimum weight except the vertex i, i.e., root i maintaining the bounding conditions (Section 3) in the path whose weight is maximum and i is the inverse 1-center. In the second case, if $w_{min} > k_1w_1$ in T_{TRP}, where $w_1 = min\{w(v), v \in G\}$, $w_{min} = min\{w(L_i), w(R_i)\}$, k_1 is the number of vertices in such path between L_i, R_i whose weight is maximum, except the root i and $w_{min} > 0$, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of some vertices except the root i maintaining the bounding conditions (Section 3) in the path whose weight is maximum and

i is the inverse 1-center. In the third case, if $w_{min} < k_1 w_1$ in T_{TRP}, where $w_1 = min\{w(v), v \in G\}$, $w_{min} = min\{w(L_i), w(R_i)\}$, k_1 is the number of vertices in such path between L_i, R_i whose weight is maximum, except the root i and $w_{min} > 0$, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices up to minimum weight except the root i maintaining the bounding conditions (Section 3) in the path whose weight is maximum and enhance the weights of some vertices except the root i in the path whose weight is minimum and i is the inverse 1-center. In the fourth case, if $w_{max} = k_2 w_2$ in T_{TRP}, where $w_2 = max\{w(v), v \in G\}$, $w_{max} = max\{w(L_i), w(R_i)\}$, k_2 is the number of vertices in such path between L_i, R_i whose weight is minimum, except the root i and $w_{max} > 0$, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of all vertices up to maximum weight except the root i maintaining the bounding conditions (Section 3) in the path whose weight is minimum and i is the inverse 1-center. In the fifth case, if $w_{max} > k_2 w_2$ in T_{TRP}, where $w_2 = max\{w(v), v \in G\}$, $w_{max} = max\{w(L_i), w(R_i)\}$, k_2 is the number of vertices in such path between L_i, R_i whose weight is minimum, except the root i and $w_{max} > 0$, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of all vertices up to maximum weight except the root i maintaining the bounding conditions (Section 3) in path whose weight is minimum and reducing the weights of some vertices except the root i in the path whose weight is maximum and i is the inverse 1-center. In the sixth case, if $w_{max} < k_2 w_2$ in T_{CIR}, where $w_2 = max\{w(v), v \in G\}$, $w_{max} = max\{w(L_i), w(R_i)\}$, k_2 is the number of vertices in such path between L_i, R_i whose weight is minimum, except the root i and $w_{max} > 0$, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of some vertices except the root i maintaining the bounding conditions (Section 3) in the path whose weight is minimum and i is the inverse 1-center.

Our proposed algorithm to the inverse 1-center location problem of the tree corresponding to the trapezoid graph G follows.

Algorithm 1-INV-TRP-TREE
Input: Weighted trapezoid graph $G = (V, E)$ with its trapezoid representation $T_i = [a_i, b_i, c_i, d_i]$, $i = 1, 2, \ldots, n$.
Output: Vertex i as the inverse 1-center of the trapezoid graph $G = (V, E)$ with the help of its tree T'_{TRP}.
Step 1. Construction of the tree T_{TRP} with root i (Algorithm TRP-TREE).
Step 2. Compute the paths R_i and L_i.
Step 3. Calculate $w(L_i)$ and $w(R_i)$.
Step 4. Modification of the tree T_{TRP}
Step 4.1. If $w(L_i) = w(R_i)$, then i is the 1-center of T_{TRP}.
Step 4.2. If $w(L_i) \neq w(R_i)$, then

Step 4.2.1. If $w_{min} = k_1 w_1$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices except the vertex i, i.e., root i up to minimum weight maintaining the bounding condition in the path whose weight is maximum, then go to Step 4.3.
Step 4.2.2. If $w_{min} > k_1 w_1$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of some vertices except the root i maintaining the bounding condition in the path whose weight is maximum, then go to Step 4.3.
Step 4.2.3. If $w_{min} < k_1 w_1$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by reducing the weights of all vertices except the root i up to minimum weight maintaining the bounding condition in the path whose weight is maximum and enhance the weights of some vertices except the root i in the path whose weight is minimum, then go to Step 4.3.
Step 4.2.4. If $w_{max} = k_2 w_2$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of all vertices except the root i up to maximum weight maintaining the bounding condition in the path whose weight is minimum, then go to Step 4.3.
Step 4.2.5. If $w_{max} > k_2 w_2$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of all vertices except the root i up to maximum weight maintaining the bounding condition in path whose weight is minimum and reducing the weights of some vertices except the root i in the path whose weight is maximum, then go to Step 4.3.
Step 4.2.6. If $w_{max} < k_2 w_2$ in T_{TRP}, then $w^*(L_i) = w^*(R_i)$ by enhancing the weights of some vertices except the root i maintaining the bounding condition in the path whose weight is minimum, then go to Step 4.3.
Step 4.3. Modified tree T'_{TRP} of the trapezoid tree T_{TRP} with $w^*(L_i) = w^*(R_i)$, and i is the inverse 1-center.
end 1-INV-TRP-TREE.

Using Algorithm 1-INV-TRP-TREE, we can find the inverse 1-center location problem on any vertex weighted tree. Justification of this statement follows the following illustration.

Illustration of the Algorithm 1-INV-TRP-TREE to the tree T_{TRP} in Figure 3. Let $i = 1$ be the pre-specified vertex of the tree T_{TRP} which is to be inverse 1-center. Next, we find the longest path L_i from the vertex 1 to other the vertex 10, i.e., the path $1 \to 2 \to 5 \to 9 \to 10$ and find another longest path R_i from 1 to the vertex 8 does not contain any vertex of the path L_i except 1, i.e., the path $1 \to 4 \to 6 \to 7 \to 8$.

Next, calculate the weights of the paths L_i and R_i. Let $w(L_i)$ and $w(R_i)$ be the sum of the weights of the vertices except the root $i = 1$ of the paths L_i and R_i, respectively. Here, $w(L_i) = 22$ and $w(R_i) = 28$. Therefore, $w(L_i) \neq w(R_i)$. Therefore, $w_{min} = w(L_i) = 22$ and $w_{max} = w(R_i) = 28$. Again, $k_1 = 4$ and $w_1 = 3$, then $k_1 w_1 = 12$. Therefore, $w_{min} > k_1 w_1$. Next, calculate $(w_{max} - w_{min})$. Therefore, $(w_{max} - w_{min}) = (28 - 22) = 6$.

Therefore, we can decrease the weights $(w_{max} - w_{min})$ from the vertices except the root i in the path whose weight is maximum using the conditions of the non-linear semi-infinite optimization model technique (Section 3). Now, we subtract the weight 3 from the weight of the vertex 4 in R_i, again we subtract the weights 1, 2 from the weights of the vertices 6, 7 respectively in R_i, then we get $w^*(R_i) = \{(6-3)+(4-1)+(5-2)+13\} = 22$. Again, $w^*(L_i) = w_{min} = w(L_i) = 22$. Hence, we get $w^*(L_i) = w^*(R_i)$. Therefore, the vertex 1 is the inverse 1-center.

The above illustration gives the following table.

Now, we have the modified tree T'_{TRP} (Figure 4) with modified vertex weight.

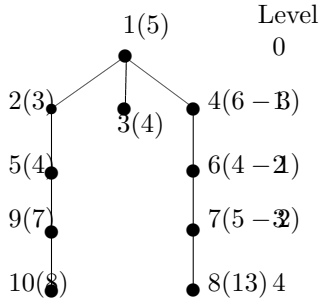

Figure 4. Modified tree T'_{TRP} of the tree T_{TRP}.

Pre-specified vertex of the tree T_{TRP}	$i = 1$
Longest path L_i from vertex 1 to vertex 10	$1 \to 2 \to 5 \to 9 \to 10$
Longest right path R_i from vertex 1 to vertex 8	$1 \to 4 \to 6 \to 7 \to 8$
Weight of path L_i except vertex $i = 1$	$w(L_i) = 22$
Weight of path R_i except vertex $i = 1$	$w(R_i) = 28$
$w_{min} = w(L_i)$	22
$w_{max} = w(R_i)$	28
Number of vertices in R_i except $i = 1$	$k_1 = 4$
Minimum weight of the vertex in the graph	$w_1 = 3$
$k_1 w_1$	12
w_{min}	$> k_1 w_1$
$w_{max} = w_{min}$	$30 - 27 = 3$
Reduce the weights of the vertices except vertex $i = 1$ in R_i	$(6 - 3) + (4 - 1) + (5 - 2) + 13$
$w * (L_i)$	22
$w * (R_i) = w(R_i)$	22
The weights of $w * (L_i)$ and $w * (R_i)$ are equal	$w * (L_i) = w * (R_i)$
The vertex which is inverse 1-center	$i = 1$

Next we shall prove the following important result.

Lemma 4.1. *Algorithm 1-INV-TRP-TREE correctly computes the inverse 1-center of the weighted trapezoid graph.*

Proof. Let i be the pre-specified vertex in T_{TRP}. We prove that i is the inverse 1-center. First, by Step 1, we constructed the tree T_{TRP} (as per Section 2) with root i, and by Step 2, we computed the longest paths R_i and L_i from i to the tree T_{TRP}. By Step 3, we calculated the weight of the paths L_i and R_i from i except i, i.e., $w(L_i)$ and $w(R_i)$. In Step 4, if $w(L_i) = w(R_i)$, then i is the vertex one center as well as inverse 1-center of T_{TRP} (Step 4.1). But if $w(L_i) \neq w(R_i)$, then we modified the tree T_{TRP} under the conditions of the non-linear semi-infinite optimization model (Step 4.2). By Step 4.3, we modified the circular-arc tree T_{TRP}, we get the weights $w^*(L_i)$ and $w^*(R_i)$ of both sides of i and we get $w^*(L_i) = w^*(R_i)$. Therefore, i is the inverse 1-center. Hence, Algorithm 1-INV-TRP-TREE correctly computes the inverse 1-center for any vertex weighted tree. □

We have another important observation in the tree T'_{TRP} given by the Algorithm 1-INV-TRP-TREE.

Lemma 4.2. *The specified vertex i in the modified tree T'_{TRP} is the inverse 1-center.*

Proof. By Algorithm 1-INV-TRP-TREE, we get $w^*(L_i) = w^*(R_i)$ in the modified tree T'_{TRP}. Therefore, the specified vertex i in the modified tree T'_{TRP} is the inverse 1-center. □

The following describe the time complexity of the algorithm to compute inverse 1-center problem on the weighted tree corresponding to the weighted trapezoid graph G.

Theorem 4.3. *The time complexity to find inverse 1-center problem on a given weighted trapezoid tree T'_{TRP} corresponding to the weighted trapezoid graph G is $O(n)$, where n is the number of vertices of the graph.*

Proof. Step 1 takes $O(n)$ time, since the adjacency relation of the trapezoid graph can be tested in $O(1)$ time. Step 2, i.e., the longest weighted path from i to v_i can be computed in $O(n)$ time, if T_{TRP} is traversed in a depth-first-search manner. Step 3 takes $O(n)$ time to compute the sum of the weights of the paths. Also, Step 4.1 takes $O(1)$ time. In the computation of k_1 and k_2, i.e., the number of vertices in R_i and L_i takes $O(n)$ time, so each repetition of Step 4.2 takes $O(n)$ time (since the comparison of two numbers and distribution of the excess weight takes $O(n)$ time, so, each repetition of Step 4.2.1 to 4.2.6 can be computed $O(n)$ time). Also, modification of weights in either R_i or L_i takes $O(n)$ time as T_{TRP} contains n vertices and $(n-1)$ edges, so Step 4.3 can be executed in $O(n)$ time. Hence, overall time complexity of our proposed Algorithm 1-INV-TRP-TREE is $O(n)$ time, where n is the number of vertices of the trapezoid graph. □

5. Concluding Remarks

In this paper, we investigated the inverse 1-center location problem with vertex weights on the tree corresponding to the weighted trapezoid graph G. First, we developed minimum height trees with two branches of level difference either zero or one of the trapezoid graphs. Second, we modified the tree maintaining the bounding conditions to get inverse 1-center. The time complexity of our proposed algorithm is $O(n)$, where n is the number of vertices of the trapezoid graph G. This idea can be applied to solve the 1-center location problem to other graphs.

Acknowledgement

We would like to thank the anonymous editors and reviewers for their insightful and constructive comments and suggestions that have been helpful for providing a better version of the present work.

References

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

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

[3] S. C. Barman, S. Mondal, and M. Pal, *A linear time algorithm to construct a tree 4-spanner on trapezoid graphs*, International Journal of Computer Mathematics, **87** (2008), 1–13.

[4] D. Burton and Ph. L. Toint, *On an instance of the inverse shortest paths problem*, Math. Programming, **53** (1992), 45–61.

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

[6] R. E. Burkard, C. Pleschiutschnig, and J. Zhang, *The inverse 1-median problem on a cycle*, Discrete Optimization, **16** (2007), 50–67.

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

[8] R. E. Burkard and B. Alizadeh, *Inverse center location problems*, International Symposium on Combinatorial Optimization, **36** (2010), 105–110.

[9] M. C. Cai and Y. J. Li, *Inverse matroid intersection problem*, ZOR Math. Meth. Oper. Res., **45** (1997), 235–243.

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

[11] D. G. Corneil and P. A. Kamula, *Extension of permutation and interval graphs*, Congressus Numerantium, **58** (1987), 267–275.

[12] I. Dagan, M. C. Golumbic, and R. Y. Pinter, *Trapezoid graphs and their coloring*, Discrete Applied Mathematics, **21** (1988), 35–46.

[13] M. S. Daskin, *Network and Discrete Location: Models, Algorithms, and Applications*, Wiley, New York, 1995.

[14] R. B. Dial, *Minimal-revenue congestion pricing part-I: A fast algorithm for the single-origin case*, Transportation Res. Part B, **33** (1999), 189–202.

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

[16] C. W. Duin and A. Volgenant, *Some inverse optimization problems under the Hamming distance*, European Journal of Operational Research, **170** (2006), 887–899.

[17] H. W. Engl, M. Hanke, and A. Neubauer, *Regularization of Inverse Problems*, Kluwer Academic Publishers Group, Dordrecht, 1996.

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

[19] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 2004.

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

[21] M. Galavi, *Inverse 1-median problems*, Ph. D. thesis, Institue of Optimization and Discrete Mathematics, Graz University of Technology, Graz, 2008.

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

[23] X. Guan and J. Zhang, *Inverse bottleneck optimization problems under weighted Hamming Distance*, Lecture Notes in Computer Science, **4041** (2006), 220–230.

[24] A. Hashimoto and J. Stevens, *Wire routing by optimizing channel assignment within large apertures*, in Proc. 8th IEEE Design Automation Workshop, (1971), 155–169.
[25] C. Heuberger, *Inverse combinatorial optimization: A survey on problems, methods, and results*, Journal of Combinatorial Optimization, **8** (2004), 329–361.
[26] B. Jana, S. Mondal, and M. Pal, *Computation of inverse 1-center location problem on the weighted trees*, CiiT International Journal of Networking and Communication Engineering, **4** (2012), 70–75.
[27] B. Jana, S. Mondal, and M. Pal, *Computation of inverse 1-center location problem on the weighted interval graphs*, Int. J. Computing Science and Mathematics, **8** (2017), 533–541.
[28] H. Kellerer, U. Pferschy, and D. Pisinger, *Knapsack Problems*, Springer, Berlin, 2004.
[29] Y. D. Liang, *Domination in trapezoid graphs*, Information Processing Letters, **52** (1994), 309–315.
[30] T. Ma, and J. P. Spinrad, *An $O(n^2)$ algorithm for 2-chain problem on certain classes of perfect graphs*, in Proc. 2nd ACM-SIAM Symp. on Discrete Algorithms, 1991.
[31] B. Marlow and N. Megiddo, *Inverse problems and linear-time algorithms for linear programming in R3 and related problems*, SIAM J. Comput., **12** (1983), 759–776.
[32] B. P. Mirchandani and R. L. Francis, *Discrete Location Theory*, Wiley, New York, 1990.
[33] S. Mondal, M. Pal, and T. K. Pal, *An optimal algorithm for solving all-pairs shortest paths on trapezoid graphs*, Intern. J. Comput. Engg. Sci., **3.2** (2002), 103–116.
[34] T. J. Moser, *Shortest paths calculation of seismic rays*, Geophysics, **56** (1991), 59–67.
[35] S. Nickel, and J. Puerto, *Location Theory, A Unified Approach*, Springer, Berlin, 2005.
[36] A. Saha, M. Pal, and T. K. Pal, *Selection of program slots of television channels for giving advertisement: A graph theoretic approach*, Information Sciences, **177.12** (2007), 2480–2492.
[37] R. E. Tarjan, *Depth first search and linear graph algorithm*, SIAM J. Comput., **2** (1972), 146–160.
[38] C. Yang and J. Z. Zhang, *Two general methods for inverse optimization problems*, Appl. Math. Lett., **12** (1999), 69–72.
[39] X. Yang and J. Zhang, *Inverse center location problem on a tree*, Journal of Systems Science and Complexity, **21** (2008), 651–664.
[40] J. Z. Zhang and Z. H. Liu, *Calculating some inverse linear programming problems*, J. Computational and Applied Mathematics, **72** (1996), 261–273.
[41] J. Z. Zhang and Z. F. Ma, *A network flow method for solving some inverse combinatorial optimization problems*, Optimization, **37** (1996), 59–72.

MSC2010: 54A40

Key words and phrases: Tree-networks, center location, 1-center location, inverse 1-center location, inverse optimization, tree, trapezoid graphs

Department of Applied Mathematics with Oceanology and Computer Programming, Vidyasagar University, Midnapore 721102, India
Email address: biswanathjana2012@gmail.com

Department of Mathematics, Raja N. L. Khan Women's College, Gope Palace,, Paschim Medinipur 721102, India
Email address: sm5971@rediffmail.com

Department of Applied Mathematics with Oceanology and Computer Programming, Vidyasagar University, Midnapore 721102, India
Email address: mmpalvu@gmail.com