Chapter 2

Computation of inverse 1-center location problem on the weighted trees*

2.1 Introduction

Let G be a connected graph, such that $|V_1| = n + 1, |E_1| = n$. Every edge $(u, v) \in E$ has different weight w_i. For a graph G, a *walk* can be defined as a finite alternating series of vertices and edges which is beginning and ending with vertices such that each edge is incident with those vertices which are preceding and following it. In a walk,there will be no edge which appears more than once. However, a vertex can appear more than once. A *path* can be defined as an open walk in which there will be no vertex appearing more than once. A *circuit* can be defined as a closed walk in which there will be no vertex (except the initial vertex and the final vertex) appearing more than once. A *tree* T is a graph which is connected, containing no circuits. That is, a *tree* T is a connected acyclic graph. Clearly there will be one and only one path between each and every pair of vertices of tree T. For an w-tree, there will be a non-negative real number attached with each edge of the tree. For an un-w-tree $T(V_1, E_1)$, where $|E_1| = |V_1| - 1$, the *eccentricity e(x)* of a vertex x can be defined as the distance from x to the vertex which is farthest from $x \in T$, i.e.

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

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

*A part of the work presented in this chapter is published in *CiiT International Journal of Networking and Communication Engineering*, 4 (2012) 70-75.

For a w-tree $T(V_1, E_1)$, the *eccentricity e(x)* of the vertex x can be defined as the sum of the weights of the edges from x to the vertex which is farthest from $x \in T$, i.e.

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

where the sum of the weights of the edges on the path between x and x_i is $d(x, x_i)$.

A *center* of a tree T can be defined by a vertex with minimum eccentricity i.e. if $e(s) = \min\{e(x), \text{ for all } x \in V_1\}$, then s is called *1-center*. We know very well that every tree is either monocentric or bicentric.

In a tree T, the eccentricity $e(x)$ of a center can be defined as the *radius* of the tree T which is denoted by $\rho(T)$, i.e.

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

For a tree T , the *diameter* can be defined as the length of the longest path that implies, the diameter is the maximum eccentricity.

Let the w-tree T with $(n + 1)$ vertices and n edges. For a tree, within certain bounds, Inv1C problem consists in changing edge weights such that a pre-specified vertex be 1-center.

Here, we propose an algorithm to compute Inv1C location problem on EdwT in $O(n)$ time, where the number of vertices of the tree is n.

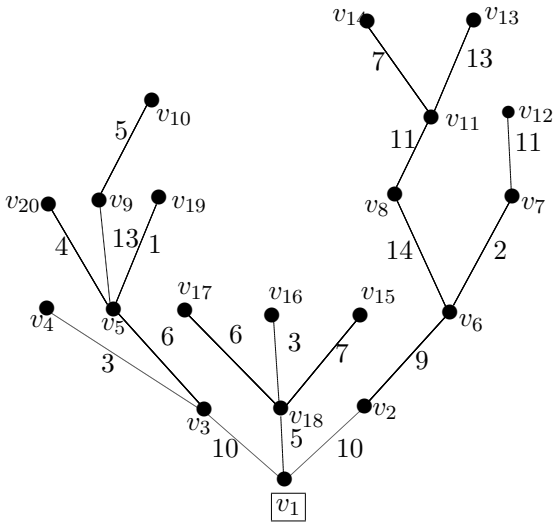

Figure 2.1: A tree T.

2.1.1 Applications of the problem

For example, an essential application appears from geophysical science and concerns foretelling the activity of earthquakes. To accomplish this aim, geologic zones have to discretized into a number of cells. In a corresponding network (Moser [77]), adjacency relations may

be modeled by arcs. Some estimates for the transmission times are well known but it is very hard to obtain precise values. After observing an earthquake and at various points, the arrival times of the resulting seismic perturbations and comprising that earthquakes move along shortest paths, the main problem is to filter the estimates of the transmission times between the cells. This problem is just an inverse shortest path problem.

There is another possible application which actually changes the real costs: Suppose, a road network is given and there are some facilities. The target is to set the facility in a way that maximum distance to the customers will be minimum. However often we face with such a situation that there is a facility, already exists which can not be relocated with equitable costs. In this situation, for improving roads costs, the modification of the network is needed as little as possible, so that the facility's location be optimum (or so that the distances to the customers will not overstep the given bounds). We have explained an instance of the inverse center location problem. For modeling traffic networks, to set tolls in order to insist an active use of the network (Dial [33]) is a further option. The 'inverse optimization' word was motivated in part by the widespread usage of inverse methods in so many other fields, for example Marlow [68] and Engl et al. [38].

2.1.2 Organization of the chapter

We shall discuss about basic terminologies and results in the next section i.e. the formulation of Inv1C problem of the EdwT. We design an algorithm to get Inv1C of the modified EdwT in Section 2.3. We have also presented some notations in this section. We also calculated the T-complexity in this section. In Section 2.4 we give the summary.

2.2 Basic terminologies and results

Let $v_0, v_1, v_2, \ldots, v_{(n-1)}, v_n$ be an unweighted path between v_0 and v_n of the tree T i.e. $(v_k, v_{k+1}) \in E_1$, for $k = 0, 1, 2, 3, \ldots, (n-1)$ and this path is denoted by $P(v_0, v_n)$. Clearly for a path $P(v_0, v_n)$, the path length, denoted by $\delta(P)$, is $d(v_0, v_n)$, i.e. $d(v_0, v_n) = \delta(P) = e(v_0) = e(v_n)$ and if this path length $P(v_0, v_n)$ is even, i.e. n is even, then radius of the path P is given by $\rho(P) = \delta/2 = d(v_0, v_n)/2$ and this is the eccentricity of the vertex $v_{\frac{n}{2}}$. So the center of the path $P(v_0, v_n)$ is $v_{\frac{n}{2}}$ and is at odd location when n is odd, then radius of the path P is given by $\rho(P) = (\delta + 1)/2$ and this is the eccentricity of each of the vertices $v_{\frac{n-1}{2}}$ and $v_{\frac{n+1}{2}}$, i.e. the two centers of the path $P(v_0, v_n)$ are $v_{\frac{n-1}{2}}$ and $v_{\frac{n+1}{2}}$ if n is odd.

Now we can introduce dummy vertex v_c in between $v_{\frac{n-1}{2}}$ and $v_{\frac{n+1}{2}}$ such that $(v_{\frac{n-1}{2}}, v_c) \in E_1$ and $(v_c, v_{\frac{n+1}{2}}) \in E_1$ so that v_c becomes the one center of the path $p(v_0, v_n)$.

Let v_l be the pre-specified vertex which is to be the center of the edge weighted path $P(v_0, v_n)$. To minimize the cost of changing the weights of the edges in order to v_l to become the center of the path $P(v_0, v_n)$ is our problem.

Now in the following, we give the steps to show the Inv1C problem on the edge weighted path $P(v_0, v_n)$.

i) Find $d_1 = d(v_j, v_0)$ and $d_2 = d(v_l, v_n)$.

ii) If $d_1 = d_2$, then v_l is 1-center.

iii) If $d_1 < d_2$ we increase the edge weights of the edges $(v_l, v_{(l-1)})$, $(v_{(l-1)}, v_{(l-2)}), \ldots, (v_1, v_0)$ and decrease the edge weights of the edges $(v_l, v_{(l+1)})$, $(v_{(l+1)}, v_{(l+2)}), \ldots, (v_{(n-1)}, v_n)$ maintaining the following three conditions:

(a) The vertex v_l becomes Inv1C of P for $\overline{w}(e)$, where $e \in E_1(P)$ i.e. for all $p \in V_1(P)$,

$$\max_{v \in V_1(P)} d_{\overline{w}}(v, v_l) \leq \max_{v \in V_1(P)} d_{\overline{w}}(v, p),$$

where $w = w(e)$ is the weight of the edge (positive real number) e and $\overline{w}(e)$ are the modified edge weights,

(b) The linear weighted(cost) mapping

$$\sum_{e \in E(P)} \{c_1^+(w)x_1(w) + c_1^-(w)y_1(w)\}$$

takes lowest cost, where $x_1(w)$ and $y_1(w)$ are the maximum amounts by which the edge weight $w(e)$ is grown and reduced gradually. $c_1^+(w)$ is the non-negative cost if $w(e)$ is grown by one unit and $c_1^-(w)$ is the non-negative cost if $w(e)$ is reduced by one unit,

(c) The new assigned edge weights stay within given modified bounds $w_{low}(e) \leq \overline{w}(e) \leq w_{upp}(e)$, for all edge in $E_1(P)$.

iv) If $d_1 > d_2$ we increase the edge weights of the edges $(v_l, v_{l+1}), (v_{l+1}, v_{l+2}), \ldots, (v_{n-1}, v_n)$ and decrease the edge weights of the edges $(v_l, v_{l-1}), (v_{l-1}, v_{l-2}), \ldots, (v_1, v_0)$ maintaining the above three conditions.

Hence, based on the above conditions, the Inv1C location problem on the EdwT T is formulated as the non-linear semi-infinite optimization model mentioned below:

$$\text{Minimize} \sum_{e \in E_1(T)} \{c_1^+(w(e))x_1(w(e)) + c_1^-(w(e))y_1(w(e))\}$$

subject to

$$\max_{v \in V_1(T)} d_{\overline{w}}(v, s) \leq \max_{v \in V_1(T)} d_{\overline{w}}(v, p), \ \forall \ p \in T (\text{or } p \in V_1(T)),$$

$$\overline{w}(e) = w(e) + x_1\{w(e)\} - y_1\{w(e)\} \ \forall \ e \in E_1(T),$$

$$x_1\{w(e)\} \leq w^+\{w(e)\}, \forall\ w \in E_1(T),$$
$$y_1\{w(e)\} \leq w^-\{w(e)\}, \forall\ w \in E_1(T),$$
$$x_1\{w(e)\}, y_1\{w(e)\} \geq 0, \forall\ w \in E_1(T),$$

where $w^+\{w(e)\} = w_{upp}(e) - w(e)$, $w^-\{w(e)\} = w(e) - w_{low}(e)$ are maximum feasible amounts for which $w(e)$ can be grown and reduced gradually. Each feasible solution (x, y) with $x = \{x_1(w(e)) : e \in E_1(T)\}$ and $y = \{y_1(w(e)) : e \in E_1(T)\}$ is known as a feasible modification of the Inv1C location problem.

2.3 Algorithm and its complexity

In this section we introduce a combinatorial algorithm for the Inv1C location problem on the EdwT T. The main idea of the introduced algorithm is as follows:

Suppose T be a w-tree with $(n+1)$ nodes and n edges. Suppose V_1 be the set of vertex and E_1 be the set of edges. Suppose s be any non-pendant specified vertex in the tree T which is to be 1-center. At first we calculate the longest weighted path from s to any pendant vertex of T. Let sv_i be the farthest weighted path from s to v_i and sv_l be the next farthest path from s to v_l such that there is no common vertex except s. Now calculate the weights of two paths sv_i and sv_l. If weight of sv_i and sv_l are equal then the vertex 1-center is s which is Inv1C of T. But our concentration is on unequal weights. If the weight of sv_i is greater than sv_l then we add the maximum weight with the pre weighted edge from s to v_l consecutively such that $w_{low}(e) \leq \overline{w}(e) \leq w_{upp}(e)$, for all $e \in E_1(T)$, where $w_{low}(e)$ and $w_{upp}(e)$ are the smallest and highest edge weights in T and $\overline{w}(e)$ be the modified edge weight. In this way if we seen the weight of sv_i and weight of sv_l are equal, then s be the Inv1C. But, if the weights of sv_i and sv_l are not equal, then we subtract the maximum weight from the pre weighted edge consecutively from the vertex s to v_i so that $w_{low}(e) \leq \overline{w}(e) \leq w_{upp}(e)$, $\forall\ e \in E_1(T)$ and in this way the weights of sv_i and sv_l must become equal. Therefore, the pre-specified node s be the Inv1C of the EdwT T.

Now, we introduce some notations for our algorithmic purpose.

R	:	Longest edge weighted path from s.
L	:	Another longest edge weighted path from s does not contain any vertex of the path R except s.
$w(R)$	:	Weight of the path R.
$w(L)$	:	Weight of the path L.
$w^*(R)$	:	Modified weight of the path R.
$w^*(L)$	:	Modified weight of the path L.

Our proposed algorithm is as follows :

Algorithm 1-INV-LOC-TREE

Input: Tree T with edge weight and specified vertex s.

Output: Vertex s as Inv1C of the tree T and modified tree T'.

Step 1. Set s as a pre-specified vertex in T.

Step 2. Compute the longest edge weighted path (only one) $R = sv_i$ from s to other vertex v_i on the given tree.

Step 3. Next compute another longest edge weighted path (only one) $L = sv_l$ from s to the vertex v_l does not contain any vertex of the path R except s.

Step 4. Calculate difference of the weights of two paths R and L i.e. $|w(R) - w(L)|$.

Step 5. //Computation of InvC//

Step 5.1. If $w(sv_i) = w(sv_l)$, $i \neq j$, then s is the vertex one center as well as Inv1C of T.

Step 5.2. If $w(sv_i) > w(sv_l)$, then distribute the weight $w(sv_i) - w(sv_l)$ on the path sv_l, i.e. L, from s such that the bounds rule holds good.

Step 5.2.1. If $w^*(L) = w(R)$, then s is the vertex 1-center.

Step 5.2.2. If $w^*(L) < w(R)$, then we decrease the excess weight of the path R from s with bounds rule until $w^*(L) = w^*(R)$.

Step 5.3. If $w(sv_i) < w(sv_l)$, then distribute the weight $w(sv_l) - w(sv_i)$ on the path sv_i, i.e. R, from s such that the bounds rule holds good.

Step 5.3.1. If $w^*(R) = w(L)$, then s is the vertex 1-center.

Step 5.3.2. If $w^*(R) < w(L)$, then we decrease the excess weight of the path L from s with bounds rule until $w^*(R) = w^*(L)$.

end 1-INV-LOC-TREE.

Using the **Algorithm 1-INV-LOC-TREE** we can find out the Inv1C location problem on any EdwT. Justification of this statement follows the following illustration.

Illustration of the algorithm to the tree T in Figure 2.1:

Let $s = v_1$ be the pre-specified vertex of the tree T which is to be one center. Next we find the longest path $R = v_1v_{13} = sv_{13}$ from s to other vertex v_{13} on the given tree and find next longest path $L = v_1v_{10} = sv_{10}$ from s to the vertex v_{10} does not contain any vertex of the path R except s. Next calculate the weights of the paths R and L. Let $w(R)$ and $w(L)$ be the weights of the paths R and L respectively. Here $w(R) = 57$ and $w(L) = 34$. Next calculate the difference of weights of two paths i.e. calculate $w(R) - w(L)$. Therefore $w(R) - w(L) = 57 - 34 = 23$. To get equal weights of $w(R)$ and $w(L)$ we add the weight 4 to the edge (v_1, v_3), 8 to the edge (v_3, v_5), 1 to the edge (v_5, v_9), 9 to the edge (v_9, v_{10})

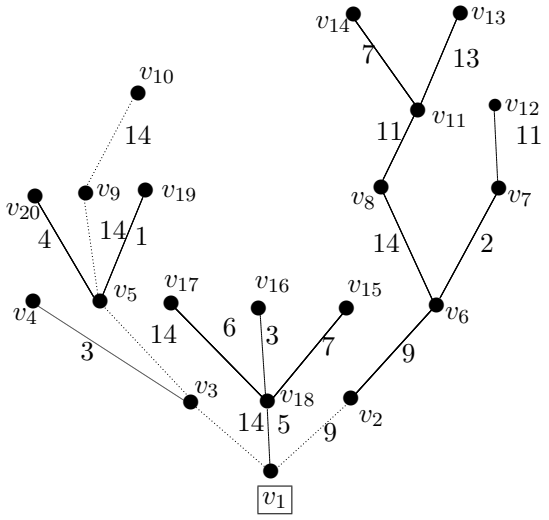

Figure 2.2: Modified tree T' of the tree T.

and decrease the weight 1 from the weight of the edge (v_1, v_2). After modification we get $w(L) = \{(10+4)+(6+8)+(13+1)+(5+9)\} = 56$ and $w(R) = \{(10-1)+9+14+11+13\} = 56$ i.e. $w(L) = w(R)$. Therefore the vertex $s = v_1$ is the inverse one center.
Now we have the modified tree T' (Figure 2.2) with modified edge weight.

Lemma 2.3.1 *The **algorithm 1-INV-LOC-TREE** correctly computes the Inv1C location on the EdwT.*

Proof. Let s be the pre-specified vertex in T. We have to prove that s is the Inv1C. At first, by Step 2 we have calculated the weight of the path $R = sv_i$ and by Step 3, the weight of the path $L = sv_l$. If $w(R) = w(L)$, then s is the vertex 1-center, also Inv1C (Step 5.1). But if $w(R) > w(L)$, then by Step 5.2 we have distributed the excess weight $w(R) - w(L)$ on the path L from s obeying the bounds conditions $w_{low}(e) \leq \overline{w}(e) \leq w_{upp}(e)$, $\forall\ e \in E_1(T)$ and if $w(R) < w(L)$, then we have distributed the excess weight $w(L) - w(R)$ on the path R obeying the same bounds rule (Step 5.3). By this process we get $w^*(R) = w^*(L)$, which is the condition of Inv1C. Therefore s is the Inv1C. Hence **algorithm 1-INV-LOC-TREE** correctly computes the Inv1C for any EdwT. □

We have another important observation in the tree T' given by the **algorithm 1-INV-LOC-TREE**.

Lemma 2.3.2 *The specified vertex s in the modified tree T' is the 1-center.*

Proof. This result directly follows from Lemma 2.3.1. By **algorithm 1-INV-LOC-TREE**, finally we get $w^*(R) = w^*(L)$ in the modified tree T'. Therefore the specified vertex s in the modified tree T' is the 1-center. □

We describe the total time needed to compute **algorithm 1-INV-LOC-TREE** on the EdwT, follows below.

Next we shall prove an important result.

Theorem 2.3.1 *The T-complexity to find Inv1C problem on a given EdwT T is $O(n)$, where the number of vertices of the tree is n.*

Proof. $O(1)$ time is needed in Step 1. If tree T is traversed in a DFS manner in Step 2, longest edge weighted path from s to v_i can be computed in $O(n)$ time . Similarly $O(n)$ time is needed to compute Step 3. Step 4 takes $O(1)$ time as comparing two numbers needs $O(1)$ time. Also Step 5 needs $O(1)$ time. Comparing two numbers and distribution of the excess weight takes $O(n)$ time. So, $O(n)$ time is needed to complete Step 5.2 and Step 5.3. Hence overall T-complexity of our proposed **algorithm 1-INV-LOC-TREE** is $O(n)$ time. □

2.4 Summary

In graph theory, for a tree, Inv1C location problem with different edge weights is an essential real life problem. In this chapter, we investigated the Inv1C location problem with different edge weights on the tree. We developed exact sequential algorithm for the *tree* with fast running time $O(n)$, where the total number of vertices of the tree is n.