A Summary of Kariv and Hakimi's Solution to the p-Median Problem on a Weighted Tree

Adam Lefaivre

CPSC 5110 Final Project I
– Dr. Robert Benkoczi –

December 6th, 2017

Introduction

The p-Median problem is a facility location problem on some graph network $G(V, E)$, with a set of edges E, and a set of vertices V. In particular, this paper solely focuses on graphs that are trees. The p-median problem assigns p facilities to be placed in the network, so that the weighted distance sum to all of these facilities is minimized. Note that all of the solutions to the p-median problem will fall at a vertex of the tree, since moving the facility closer and closer to a vertex produces a smaller sum.

The distance function between any two nodes is given as:

$$d(v, X) = \min_{v' \in X} \{d(v, v')\}$$

where X is just a set of points at G's vertices. This means that we will always try to find the minimum path between two nodes. The minimum sum of all of the points X is then defined as:

$$H(X) = \sum_{v \in V} w(v) d(v, X)$$

The p-median solutions are said to be in the set V_p^*, where

$$H(V_p^*) = \min \{H(V_p) | V_p \text{ a set of } p \text{ vertices}\} = \min_{X \text{ on } G} \{H(X)\}$$

$p > n$ needs to be maintained, or else the overall cost would be trivially 0.

In this paper Kariv and Hakimi also show that the 1-median solution vertex is the weighted centroid of a tree, that there is a simple $O(n)$ algorithm presented by Goldman[2] that enables one to achieve this 1-median solution in a tree, and that the p-median problem is an NP-hard problem. We will simply describe these proofs and this algorithm to give some background of the problem before moving forwards; the task at hand is to simply summarize, to some extent, the solution that Kariv and Hakimi proposed in 1979.

The proof of the p-median solution being a weighted centroid relies on showing that it cannot be otherwise, or else there would be a subtree that has greater than half the weight of the tree, without loss of generality. The proof of p-median for being NP-hard uses the dominating set problem. If we convert the set of vertices into a graph representation and solve the p-median problem we can obtain the optimal cost z. Let there be n vertices with $p = k$ solutions, where k is the number of vertices in the dominating set. Then if we have $z \leq n - k$ then we can return the k-medians as a certificate to the covering problem. This shows that finding the dominating set is in fact a polynomial time problem, and shows further that the p-median problem is NP-hard. Lastly, the algorithm for finding the 1-median, as given by Goldman[2] (and explained by Kariv and Hakimi) is to traverse upwards from the leaves towards the root, summing the weights of all the previous nodes as well as the

current one. We stop when we get to a point where the total weight is greater than half. This ensures that we find a weighted centroid and thus the 1-median.

These are some basic ideas that should be familiar. However, Kariv and Hakimi's solution to the p-median problem goes far beyond these simple notions, and is actually one of the first algorithms to be proposed for the solution of the p-median problem. Note that several solutions that were more time-intensive were known at the time, including the $O(\binom{n}{p}np)$ enumeration of all the combinations of p medians. However, Kariv and Hakimi's solution actually solves the algorithm in $O(n^2p^2)$ time.

Fundamental Functions and Variables

We will strictly use the same notation as Kariv and Hakimi do. This is in part because the notation is already daunting, and changing it would not be beneficial to the clarity of the ensuing explanations. The same notation is given below as in their *Notations* section.

1. Let our tree be called T, and for now, some arbitrary subtree of T be called T'.
2. Let v_0 be the root of the total tree and all of the subtrees. Note that $T = T_{v_0}$.
3. Let $V^*(T', k)$ be the k-median solution of a tree T'. The distance to the k solutions must be minimal.
4. $H(T', k)$ is the total weighted distance-sum, for some arbitrary tree T'. Note that $H(T', k) = \sum_{v' \in T'} w(v')d(v', V^*(T', k))$, and of course the whole tree's solution cost can be found by: $H(T_{v_0}, k)$.
5. Let $c(v, k)$ be a vertex that belongs to $V^*(T_v, k)$ which covers v.
6. Let $e(v, l)$ be an edge connecting vertex v to another vertex, where l is the branching factor number (i.e. the edge connecting v to its lth son)
7. Let $T_{e(v,l)}$ be a subtree of T_v that is maximally connected. If we let all of the sons of v be t then we get the complete tree, that is: $T_{e(v,t)} = T_v$. Recall that we can think of l denoting the lth son of v. This notation is somewhat confusing so it will be elaborated on in further detail, below.

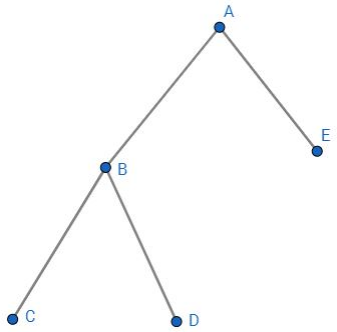

Figure 1: *A short example of the meaning of the notation* $T_{e(v,l)}$

So, in the above figure, we note that:

$$T(C,0) = \{C\}$$
$$T(D,0) = \{D\}$$
$$T(E,0) = \{E\}$$
$$T(B,0) = \{B\}$$
$$T(B,1) = \{B,C\}$$
$$T(B,2) = \{B,C,D\}$$
$$T(A,0) = \{A\}$$
$$T(A,1) = \{A,B,C,D\}$$
$$T(A,2) = \{A,B,C,D,E\}$$

Notice here that $T_{e(v,l-1)} \subset T_{e(v,l)}$. This is used in many different places by this algorithm. Especially for placing a covering facility and finding a partial sum solution with respect to the subtree that it is covering.

There is just one more function to discuss. It is a fundamental function to Kariv and Hakimi's dynamic algorithm solution. It is given as:

$$R(k, e(v,l), v_r) = \sum_{v' \in T_{e(v,l)}} w(v') d(v', V_p^*)$$

Where $e(v,l)$ denotes some edge and v_r is some vertex that belongs to the set of p-median vertices V_p^* which covers v. k is the number of points of V_p^* that cover the subtree $T_{e(v,l)}$. The above expression denotes the partial distance summation to any such median location for that subtree $T_{e(v,l)}$. So now we have a way to show the summation of the cost for some subtree.

The Relationship Between the Functions R and H

We must now elaborate on the relationship between the $R(k, e(v,l), v_r)$ and $H(T', k)$ functions. As they occur frequently together throughout Kariv and Hakimi's solution. If we let some V_1 be a set of vertices that belong to that tree $T_{e(v,l)}$ covered by v_r, then we get:

$$R(k, e(v,l), v_r) = \sum_{v' \in V_1} w(v')d(v', v_r) + \sum_{v' \in T_{e(v,l)} - V_1} w(v')d(v', V_k^* - \{v_r\})$$

Let us elaborate on this equation. First recall that the function R gives the partial summation solution for a certain subtree, where the vertex v_r is the vertex of coverage for that subtree. So since V_1 belongs to $T_{e(v,l)}$ we can find the whole subtree partial distance summation by taking the left summand, which is the partial sum for that subtree V_1 and add it to the right summand, which is the partial sum for everything else in the tree of interest except for the subtree V_1. Thus giving us the total partial summation for $T_{e(v,l)}$. Note, however, that we can exploit the relationship between H and R here.

In the above equation, the right summand excludes a covering vertex. So the total summation accounts for the total solution of $k-1$ facilities covering that subtree $T_{e(v,l)} - V_1$. That is to say, there are at most $k-1$ facilities that can be in that summand. So here we can just use the H function, since it is the total cost of a subtree. Then, substituting this function in, we get the expression:

$$R(k, e(v,l), v_r) = \sum_{v' \in V_1} w(v')d(v', v_r) + H(T_{e(v,l)} - V_1, k-1)$$

This is the recursive definition that Kariv and Hakimi's solution is based on. They supply one more step and say this summation should be minimized for all V_1 subsets, or rather, that we can find a subset that minimizes the cost of placing k-facilities. This is shown by:

$$R(k, e(v,l), v_r) = \min_{V_1} \Big\{ \sum_{v' \in V_1} w(v')d(v', v_r) + H(T_{e(v,l)} - V_1, k-1) \Big\}$$

So after all of the covering nodes v_2 have been enumerated in the cases that we will discuss below we can find, for a whole subtree rooted at v_s, with t_s number of sons (sons being the direct descendants only), the minimum total cost for the k medians covering subtree T_{v_s}. This gives:

$$H(T_{v_s}, k) = \min_{v_2 \in T_{v_s}} \{R(k, e(v_s, t_s), v_2)\}$$

This is given here, since we need some way of showing the purpose of applying all of the partial sum calculations for R. Remember that $H(T_{v_0}, p)$ gives us our final cost.

The Algorithm

A Broad Overview

The algorithm itself can be broken down into several steps. The first step is a precomputation step, the second step finds the cost of the p-median solution and the final step returns all of the facilities by which the optimal cost was achieved. We will focus mainly on the precomputation and finding the cost of the p-median solution.

How the algorithm works at a very high level will be explained, to gain some intuition of the algorithm in general before continuing forwards. For every level of the tree we choose a node v_s, which is the current "root" of all the subtrees to be dealt with. We can then imagine fixing a coverage point v_2 inside of v_s's subtree, and calculating the partial sum, given by R, for all of the k facilities that account for the coverage cost. We check v_2's influence over some other node v_1. Then we need to account for the cost of all of v_1's subtrees and its sons subtrees too (son, in this case, refers to the immediate descendants only). Since we are performing everything in a bottom-up approach, these sums should already be calculated. Finally, after calculating the partial sums at that level, we perform an assignment to the total sum for T_{v_s} using the H function. One can imagine various cases arising when we do need to fix that coverage point, one example includes the coverage point v_2 being in the same subtree as that which it is covering. Most of the rest of this paper attempts to reconcile these cases against one another, so that the algorithm can be explained thoroughly. We will now explain the precomputation steps necessary for the algorithm.

Precomputation

This step is very important for the algorithm to execute in the running time that it was meant to. The values stored in this step are used so frequently throughout the program, that without them the algorithm would have many more computations if they were repeated at every step.

The list of computations that initially need to be completed are:

1. Choosing a root v_0
2. Finding the greatest level of the tree L_m
3. Finding a list of subtrees $T_e(v, l)$ for each level of T_{v_0}.
4. Calculating a distance matrix for every node to every other node
5. A list of leaves
6. Other helpful steps not given by Kariv and Hakimi include:

(a) Finding a list of sons for each node

(b) Filling the H, R, and c tables with initial undefined values

(c) Finding the levels of every node

The operation that gives the largest running time above is calculating the distance matrix for each node, this takes $O(n^2)$ time, since we need to find the shortest path from one node to another which is an $O(n)$ operation that we perform n times. We can calculate everything else in a bottom-up fashion starting at the highest level for the other precomputation steps if need be, however these steps themselves will not take longer than the running time of the main algorithm, being $O(n^2p^2)$.

Since we have a list of leaves now, we can perform the trivial cases of the functions on each leaf. This yields:

$$H(T_v, 1) = 0$$

$$c(v, 1) = v$$

$$R(1, e(v, 0), v) = 0$$

For any leaf, $H(T_v, 1)$ implies that we are placing one facility directly on the leaf v, giving an immediate cost of 0. $c(v, 1) = v$ implies that we are placing a facility at the location of that node itself. Finally, $R(1, e(v, 0), v) = 0$ implies that we look to the subtree that is just the leaf itself (i.e. $T_{e(v,0)}$), and with a node that covers itself we get the weighted distance of 0.

Now that we have seen all of the precomputation steps, we can go on to explaining how to exploit the relationship between H and R at the ith stage of the algorithm.

Finding the Cost of The p-Median Solution

For the ith stage of the algorithm, we consider some vertex v_s, and some other two nodes v_1 and $v_2 \in T_{v_s}$. The level of v_s is $L_m - i$. We also need to have a path between v_1 and v_2 pass through v_s to consider the cases that will ensue, that is, v_s is found on the path $p(v_1, v_2)$. If v_s is indeed the root of the subtree then the path needs to go through the root of that subtree. Let $e(v_1, j_1)$ denote the branch that connects to v_3.

We then need to compute $R(k, e(v_1, j_1), v_2)$ for $1 \leq l \leq t_s$. Where t_s is the maximum number of children belonging to the node v_1. We perform it in this fashion: Compute $R(k, e(v_1, j_1), v_2)$ for $e(v_1, j_1)$, v_2 belonging to $T_{e(v_s,1)}$ but not $T_{e(v,0)}$, compute $R(k, e(v_1, j_1), v_2)$ for $e(v_1, j_1)$, v_2 belonging to $T_{e(v_s,2)}$, but not $T_{e(v,1)}$, and so on, until we hit $T_{e(v_s,t_s)}$.

Kariv and Hakimi refer to $T_{e(v_s,l)} - T_{e(v_s,l-1)}$ as a way to focus on the current subtree at the lth son, and consider all of the new unique vertices in that subtree. Now note that v_1 can truly be in any of the previous subtrees of one of the sons that was already considered. This generates several cases, but before describing them, let us observe what is happening in the diagram below.

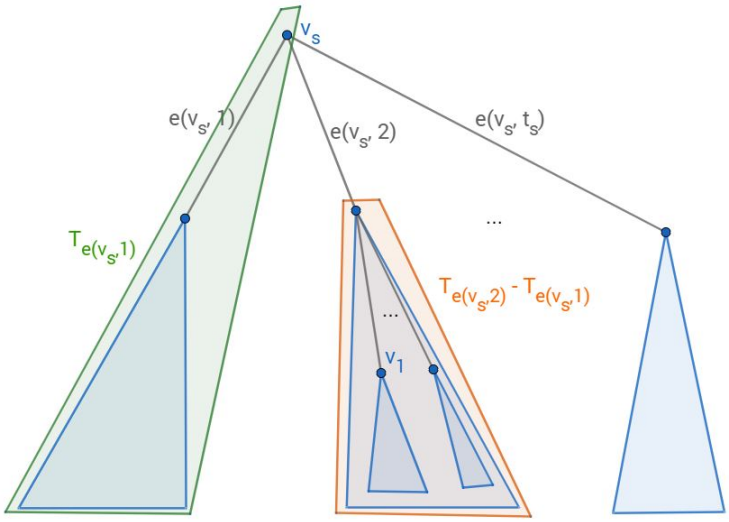

Figure 2: *Considering the* $T_{e(v_s,l)} - T_{e(v_s,l-1)}$ *subtrees*

Continuing forwards, if we generalize the above mentioned process, there are four cases that might arise for $e(v_1, j_1)$ and $v_2 \in T_{v_s}$:

i. $v_1 \in T_{e(v_s,l)} - T_{e(v_s,l-1)}$ and $v_2 \in T_{e(v_s,l-1)}$

ii. $e(v_1, j_1) = e(v_s, l)$ and $v_2 \in T_{e(v_s,l-1)}$

iii. $e(v_1, j_1) = e(v_s, l)$ and $v_2 \in T_{e(v_s,l)} - T_{e(v_s,l-1)}$

iv. $e(v_1, j_1) \in T_{e(v_s,l-1)}$ and $v_2 \in T_{e(v_s,l)} - T_{e(v_s,l-1)}$

Remember that for every case we need to have v_s as a node found in the path $p(v_1, v_2)$. Think of the v_2 node in all of the cases $R(k, e(v_1, j_1), v_2)$ (for cases i and iv) and $R(k, e(v_s, l), v_2)$ (for cases ii and iii) as a felt pen pushing down on a piece of paper, with its ink bleeding outwards. This is the same as for the coverage of v_2, the cases above just give some way of computing that influence mathematically with respect to the k facilities of that partial sum that cover the subtree $T_{e(v_1,j_1)}$.

We will now briefly discuss each case and the special computations done for each case before presenting the final algorithm identifying the optimal p-median cost. We will focus mostly on case i, since case iv is very similar to it, and case ii and iii are somewhat special cases.

Cases i and iv

Case i

Think of v_2 being placed in one of any of the left subtrees of some v_s and v_1 being in only the most right subtree at that iteration (we see this is the opposite case for iv). Then,

the computation step for case i entails choosing some covering vertex $v_2 \in T_{(v_s,l-1)}$ and enumerating the edges of $T_{e(v_s,l)} - T_{e(v_s,l-1)}$, and then repeating this process for different choices of $v_2 \in T_{(v_s,l-1)}$, which we go about in level-wise order. We skip the edge $e(v_s, l)$ since it is covered in case ii.

Note that we can assign the cost of v_1 itself by having:

$$R(1, e(v_1, 0), v_2) = w(v_1)d(v_1, v_2)$$

This is just simply the weighted distance, from v_1 to our covering vertex v_2, assuming that we just have one median placed on the covering vertex v_2. This is all we need to do if v_1 is a leaf, however if it is not a leaf we have more work to do, since we can start enumerating all of v_1's sons subtrees. Note that the letter t denotes the maximum number of sons of a node. If we let t_1 be the number of sons of v_1, then we can find the partial sum of all of the sons of v_1, if $1 \leq j_1 \leq t_1$, by the following:

$$R(k, e(v_1, j_1), v_2)$$

This means that we enumerate all of v_1's sons. We also need to account for the sons of sons of v_1 as well. That means if there is some node at the end of $e(v_1, j_1)$ which we call v_3 then we need to account for all of v_3's sons subtrees too. Note that we can use the maximally connected subtree of $T_{e(v_3,t_3)}$ here to just consider all of v_3's subtrees. Please see the diagram below. It will illuminate what Kariv and Hakimi are discussing:

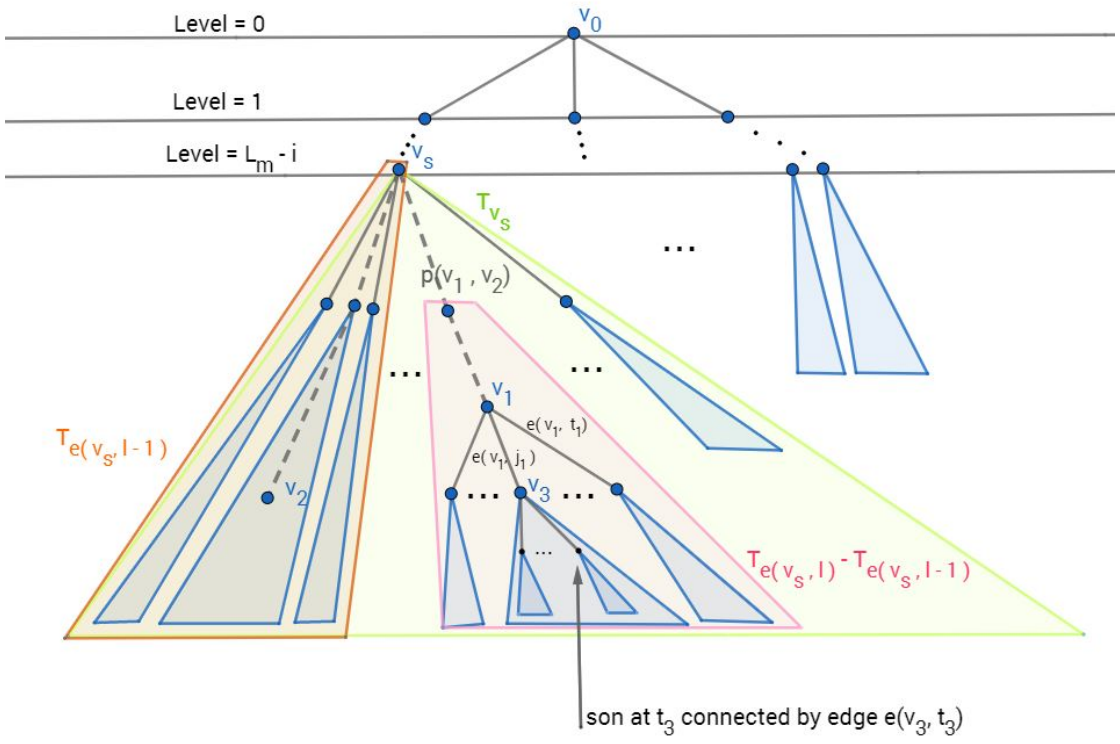

Figure 3: *Considering case i, so far, and the subtrees discussed*

Kariv and Hakimi then give the following equation to deal with everything having to do with case i. It is given below, and will be discussed:

$$R(k, e(v_1, j_1), v_2) = \min_{k_1,k_2} \Big\{ R(k_1, e(v_1, j_1 - 1), v_2) + R(k + 1 - k_1, e(v_3, t_3), v_2),$$

$$R(k_2, e(v_1, j_1 - 1), v_2) + H(T_{v_3}, k - k_2) \Big\}$$

So we fix v_2 to be some covering node of v_3, as discussed earlier. That is why we see v_2 as the covering node in all of the summands above. There are two cases by which we are picking the minimum from. The first case is where v_3 is covered by v_2 and the second case is where v_3 is not covered by v_2. This is reflected in the two terms to pick from in the minimum computation above.

If v_3 is covered by v_2 then we express this by taking the partial sum of all the nodes in the $v_1, j_1 - 1$ subtree to some number of k_1 facilities plus the cost of all of the associated children of that subtree that happen to not have the partial distances accounted for by that number of k_1 facilities, but are still under the coverage of v_2.

If v_3 is not covered by v_2 (i.e. the second case of the minimum), then we can exploit the H and R relationship discussed earlier and find the whole sum of the facilities not accounted for in T_{v_3} by taking the difference between all of the facilities and the k_2 number of facilities. This takes care of the partial sum of T_{v_3} not accounted for by k_2 facilities then. The R component of this second portion of the minimum defines the partial sum distance for the v_2 covering node for the rest of the $T_{e(v_1,j_1-1)}$ subtrees, plus the total sum of the $k - k_2$ facilities, for the k_2 facilities that do account for the cost in the T_{v_3} subtree. Remember this is how the R and H tables sum together to give us the total coverage cost. This is important to the recursive definitions of the algorithm and is how we find the minimum possible coverage for any facilities to be placed.

We enumerate all of the costs of all of the possible facilities that can be placed here, hence the usage of k_1 and k_2 here. k_1 is the number of facilities responsible for the partial sum of v_2 covering v_3, and k_2 is the number of facilities responsible for the partial sum of v_2 not covering v_3. Notice that in each of the summands given above, k_1 and k_2, are always either between 1, and the total number of nodes in the subtree being considered. Kariv and Hakimi then solve for the ranges by which k_1 and k_2 can be enumerated, These ranges are:

$$\forall k_1 \in \Big[\max(1, k + 1 - \big|T_{v_3}\big|), \min(k, \big|T_{e(v_1,j_1-1)}\big|) \Big]$$

$$\forall k_2 \in \Big[\max(1, k - \big|T_{v_3}\big|), \min(k - 1, \big|T_{e(v_1,j_1-1)}\big|) \Big]$$

These ranges, and the others that follow, ensure that we find the minimum cost for any given number of facilities (i.e. $k_1 + k_2$ in this case) covering the subtrees of interest.

Case iv

So for case i just described, we can also note how similar case iv will be to this case. These cases are given again below:

i. $v_1 \in T_{e(v_s,l)} - T_{e(v_s,l-1)}$ and $v_2 \in T_{e(v_s,l-1)}$

iv. $e(v_1, j_1) \in T_{e(v_s,l-1)}$ and $v_2 \in T_{e(v_s,l)} - T_{e(v_s,l-1)}$

Notice how v_2 would then just be in the opposite subtree in case iv as it was in case i. Furthermore notice in iv that $e(v_1, j_1)$ (and therefore v_1) is now in the opposite subtree as it was in case i. So this is why we can apply all of the same steps above, but just for v_1 and v_2 in the exact opposite subtrees. We will now discuss the two special cases ii and iii.

Cases ii and iii

Cases ii and iii are given as:

ii. $e(v_1, j_1) = e(v_s, l)$ and $v_2 \in T_{e(v_s,l-1)}$

iii. $e(v_1, j_1) = e(v_s, l)$ and $v_2 \in T_{e(v_s,l)} - T_{e(v_s,l-1)}$

We stated that these cases were special considerations above, this is firstly because we skip over the $e(v_s, l)$ edge in cases i and iv, and do not consider it, however we need some way of dealing with this edge as well. Furthermore, recall that v_s needs to be on the path $p(v_1, v_2)$. Well, in these cases we treat v_1 as v_s, then have v_2 being at the end of an edge that connects to v_2. So basically we are saying the covering node is the root v_s. Thus, we see why these are special cases.

Case ii

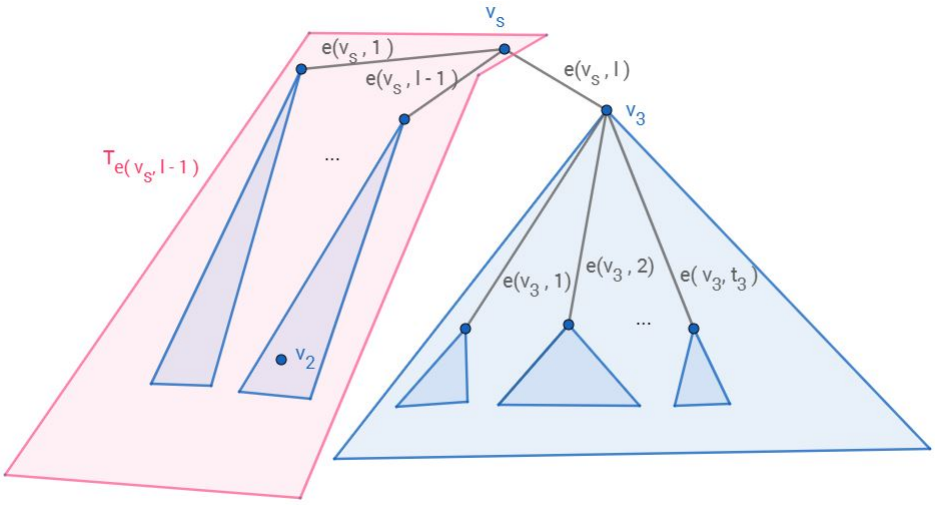

Figure 4: *Case ii*

So, the first thing that we can notice is that $p(v_s, v_2)$ is inside the subtree rooted at v_s (i.e. the subtree $T_{e(v_s,l)}$, but more specifically $T_{e(v_s,l-1)}$). With t_3 as the number of sons of v_3, we just simply take the first equation for case i and apply it, but change the context of the variables to represent $e(v_1, j_1) = e(v_s, l)$. Note that we still need to consider facilities in the partial sum of v_3's subtree, thus giving:

$$R(k, e(v_s, l), v_2) = \min_{k_1,k_2} \Big\{ R(k_1, e(v_s, l-1), v_2) + R(k+1-k_1, e(v_3, t_3), v_2),$$
$$R(k_2, e(v_s, l-1), v_2) + H(T_{v_3}, k-k_2) \Big\}$$

This computation is done for every single $v_2 \in T_{e(v_s,l-1)}$. Like mentioned above, notice how all of the $e(v_1, j_1)$ edges in the equation have been replaced with $e(v_s, l)$. The computation then becomes the same as it was for case i. The first component of the minimum is the sum of k_1 facilities covering the $T_{e(v_s,l-1)}$ subtree plus the cost of not having k_1 facilities covering the v_s subtree. The second component of the minimum uses the same relationship R and H relationship that we already discussed in case i. That is the partial sum for $T_{e(v_s,l-1)}$, given k_2 facilities, plus everything else in T_{v_3} not accounted for by k_2 facilities.

Kariv and Hakimi give the bounds for this case, which are found in the same way that they are for case i. They are:

$$\forall k_1 \in \Big[\max(1, k+1-\big|T_{v_3}\big|), \min(k, \big|T_{e(v_s,l-1)} + Lev(v_s) - Lev(v_2)\big|) \Big]$$

$$\forall k_2 \in \Big[\max(1, k-\big|T_{v_3}\big|), \min(k-1, \big|T_{e(v_s,l-1)} + Lev(v_s) - Lev(v_2)\big|) \Big]$$

Notice that the difference here is that we replaced $T_e(v_1, j_1 - 1)$ with $T_e(v_1, l-1)$ and that we added $Lev(v_s) - Lev(v_2)$ to the furthest that we can enumerate through. This is because we notice that the number of nodes in the path from v_s to v_2 is $Lev(v_s) - Lev(v_2) + 1$, and the number of nodes on the path from v_2 to v_3 is $Lev(v_3) - Lev(v_2) + 1$. Since for k_2 we can consider placing any number of facilities on the path from v_s to v_2. If we look at $R(k_1, e(v_s, l-1), v_2)$ or $R(k_2, e(v_s, l-1), v_2)$ in the above minimum calculation, $T_{e(v_s,l-1)}$ does in fact include path $p(v_s, v_2)$.

Case iii

So in case ii we considered $v_2 \in T_{e(v_s,l-1)}$, but then of course we need to consider the case of v_2 belonging to the subtree $T_{e(v_s,l)} - T_{e(v_s,l-1)}$ also. Notice how this is the same idea of how i and iv are related, in that they are opposite considerations of v_2's coverage, as we discussed in case i.

Case iii is seen below:

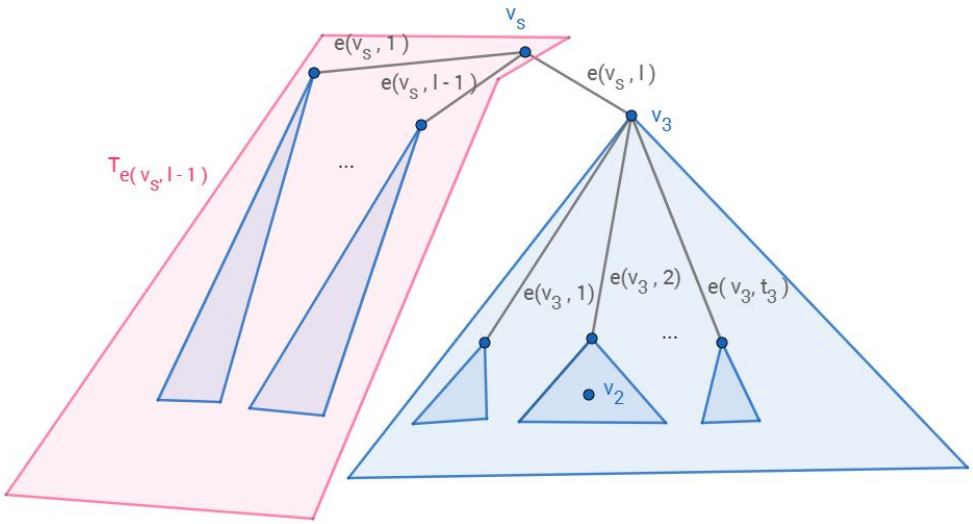

Figure 5: *Case iii*

The partial sum equation is given as:

$$R(k, e(v_s, l), v_2) = \min_{k_1} \Big\{ R(k_1, e(v_s, l-1), v_2) + R(k+1-k_1, e(v_3, t_3), v_2) \Big\}$$

We notice that this is exactly the same as the first component of the minimum in case ii. This gives the cost of some k_1 facilities for the subtree $T_{e(v_s,l-1)}$ where v_2 covers v_3, in addition to the facility costs of all of the rest of the subtree $T_{e(v_3,t_3)}$ without being covered by k_1 facilities. We only need this one component, because v_2 is in the same subtree as v_3.

The range of the facilities of k_1 are given by Kariv and Hakimi as:

$$1 \leq k_1 \leq \min \left\{ k, |T_{e(v_s,l-1)}| \right\}$$

Notice here that this range reflects the fact that we will either have the maximum number of facilities possible, or all of the nodes in $T_{e(v_s,l-1)}$, which makes sense because there may be k_1 facilities covering $T_{e(v_s,l-1)}$, according to the first summand.

The range for the facilities that are not in the set k_1 are given by Kariv and Hakimi as:

$$1 \leq k + 1 - k_1 \leq \left\{ p, |T_{v_3}| + 1 + Lev(v_s) - Lev(v_2) \right\}$$

Which is because the maximum number of facilities not represented by k_1 that can be placed has to be a maximum of the number of nodes in the subtree T_{v_3}, as well as those nodes on the path from v_s to v_2.

This is then solved for the general boundaries for all k_1 facilities by Kariv and Hakimi, giving:

$$\forall k_1 \in \Big[\max(1, k + Lev(v_2) - Lev(v_s) - \left|T_{v_3}\right|), \min(k, \left|T_{e(v_s,l-1)}\right|) \Big]$$

Above is a brief explanation of all of the cases presented by Kariv and Hakimi that might occur when we choose some v_s as a root to some subtree and make it such that $p(v_1, v_2)$ has to include v_s, where v_2 is a covering node. The cases where v_1 is v_s are given by cases ii and iii, and the functions are altered accordingly to update the partial sum costs given by R. We notice that the minimum coverage cost is being achieved by checking for the suitable amount of k_1 and k_2 covering facilities.

Updating H and c

After all of the cases have been accounted for in the algorithm, the last step is to update the H value for that v_s vertex. This was mentioned in the section describing the relationship between H and R. We need to find the cost for the whole subtree of v_s for all of the possible covering v_2 nodes, this is given by:

$$H(T_{v_s}, k) = \min_{v_2 \in T_{v_s}} \left\{ R(k, e(v_s, t_s), v_2) \right\}$$

This ensures that we achieve the minimum sum for k vertices in the subtree of v_s where v_2 covers v_s. Which was the objective of the cases above. Using the function $c(v, k)$ we can assign the vertex that covers v_s minimally. We can use the relationship between H and R to do so, this is given by:

$$d(v_s, c(v_s, k)) = \min_{v'_r \in T_{v_s}} \left\{ d(v_s, v'_r) | R(k, e(v_s, t_s), v'_r) = H(T_{v_s}, k) \right\}$$

This states that we are choosing the vertex with the smallest distance from v_s to cover v_s, given the condition that the node that covers v_s has the minimum cost, given by H.

Discussion on How to Retrieve the p-median Vertices

We will briefly discuss how Kariv and Hakimi dealt with obtaining the p-facilities. Firstly, we know from calculating the cost, all of the $H(T_v, k)$, $c(v, k)$ and $R(k, e(v, l), v_r)$ values. We can apply a depth first search, and at each point, we know from our R table, the v_2 covering vertex for the subtree rooted at v_3, remember that $e(v_1, j_1) = (v_1, v_3)$. Refer to figure 3 here, if need be. We can use the H and R tables to compute the number of medians we need to cover the total subtree of the node we are on in the depth first search (this is denoted by k_1).

There are two cases at any point in the depth first search. Our tree T_{v_3} is either covered by v_2 given by the R table, or it is not. If it is covered then we need to continue the depth first search down into the subtree T_{v_3} with $k - k_1 + 1$ facilities left (since we have potentially placed k_1 facilities). We also need to remove the cost of the partial sum to reflect the k_1 facilities being applied before we visit T_{v_3} and continue our search. If T_{v_3} is not covered by v_2 in the R table, then we can repeat the whole process via recursion by treating v_3 as v_1,

except with this recursion we will have $k - k_1$ facilities in total.

Final Discussion of the Overall Algorithm

For reference, the overview of steps needed to compute the p-median cost have been attached on the next page. In it we can see how the v_2 coverage nodes belong to either the subtree $T_{e(v_s,l-1)}$, or $T_{e(v_s,l)} - T_{e(v_s,l-1)}$, and how the node v_1 is fixed to be somewhere in either of those subtrees as well, given one of the four cases discussed. We can also see how the sons within v_1 are enumerated as well, and how this applies to the assignments that were discussed.

If we look at how the loops are structured in the algorithm overview, we can see that when we consider case i and iv, we loop through all of the v_2 and v_1 combinations, and then inside of these loops there could be p facilities under consideration. Then, within each assignment for all of the cases both k_1 and k_2 have ranges up to p as well, which gives an overall complexity of $O(n^2p^2)$, as promised by Kariv and Hakimi. This still holds as the maximum complexity, because the second phase is only of $O(n^2)$ time complexity. Since for every edge we traverse we will need to traverse a maximum number of $O(n)$ times as well, therefore the complexity of finding the overall cost outweighs the $O(n^2)$ complexity of retrieving the facilities, thus giving a maximum running time of $O(n^2p^2)$.

This algorithm makes use of calculating a subset of p-medians on various subtrees, where the partial cost and total cost functions are dependent on each other. Four important cases were examined in order to account for the coverage of one subtree by a specific number of facilities. The algorithm is quite complex, but was one of the first algorithms to ever solve the p-median problem on a tree.

An Overview of the p-Median Cost Calculating Algorithm

Here we will briefly mention the steps needed, in a very high level fashion, to obtain the minimum cost of the p-median solution. These steps are given below:

Perform all precomputation
Assign all initial values to leaves
Find the maximum level of the tree

loop through all levels of the tree:
  loop through all unprocessed v_s nodes:
    loop through all sons of v_s:
      compute trees: $T_{e(v_s,l)}$, $T_{e(v_s,l-1)}$, and $T_{e(v_s,l)} - T_{e(v_s,l-1)}$

      case i:
      loop through all v_2 nodes in $T_{e(v_s,l-1)}$:
        loop from the max level of $T_{e(v_s,l-1)}$ to $Lev(v_2)$:
          loop through all potential v_1 nodes in $T_{e(v_s,l-1)}$ at that level:
            loop through the sons of v_1:
              perform the assignment for case i

      case iv:
      loop through all v_2 nodes in $T_{e(v_s,l)} - T_{e(v_s,l-1)}$:
        loop from the max level of $T_{e(v_s,l)} - T_{e(v_s,l-1)}$ to $Lev(v_2)$:
          loop through all potential v_1 nodes in $T_{e(v_s,l)} - T_{e(v_s,l-1)}$ at that level:
            loop through the sons of v_1:
              perform the assignment for case iv (same as for case i)

      case ii:
      loop through all v_2 nodes in $T_{e(v_s,l-1)}$ at that level:
        perform the assignment for case ii

      case iii:
      loop through all v_2 nodes in $T_{e(v_s,l)} - T_{e(v_s,l-1)}$ at that level:
        perform the assignment for case iii

    update H **and** c **tables**:
    perform the assignments for H and c

References

[1] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. ii: The p-medians. *SIAM Journal on Applied Mathematics*, 37(3):539–560, 1979.

[2] A. J. Goldman. Optimal center location in simple networks. *Transportation Science*, 5(2):212–221, 1971.