ELSEVIER

Operations Research Letters 19 (1996) 59–64

operations research letters

# An O($pn^2$) algorithm for the $p$-median and related problems on tree graphs

Arie Tamir

*Department of Statistics and Operations Research, School of Mathematical Sciences, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Ramat-Aviv 69978, Israel*

Received 1 February 1995; revised 1 February 1996

**Abstract**

We improve the complexity bound of the $p$-median problem on trees by showing that the total running time of the "leaves to root" dynamic programming algorithm is O($pn^2$).

*Keywords:* Facility location; $p$-median problems; $p$-coverage problems; Tree graphs

## 1. Introduction

One of the classical problems in location theory is the $p$-median problem defined as follows: Given is a connected undirected graph $G = (V,E)$ with node set $V = \{v_1,\dots,v_n\}$ and edge set $E$. Each edge is associated with a nonnegative weight (length). The length of a path in $G$ is the sum of the weights of its edges. For each pair of nodes $v_i, v_j$ we let $d(v_i,v_j)$, the *distance* between $v_i$ and $v_j$, be the length of a shortest length path connecting $v_i$ and $v_j$. The problem is to select a subset $S$ of $p$ nodes (service centers) that will minimize the sum of the distances of all nodes (customers) to their respective nearest member (center) in $S$,

$$\sum_{v_i\in V} \min_{v_j\in S} d(v_i,v_j) .$$

Kariv and Hakimi [8] proved that the problem is NP-hard, and Lin and Vitter [10] presented several approximation schemes.

For tree graphs Kariv and Hakimi [8] described an O($p^2n^2$) algorithm, which is the lowest order method known. A different O($pn^3$) algorithm was given in Hsu [6]. In this paper we will show that the time needed to solve the $p$-median problem on a tree by the "leaves to root" dynamic programming algorithm is only O($pn^2$). For comparison purposes we note that Hassin and Tamir [5] achieved a complexity bound of O($pn$) for path graphs.

We consider a more general model which unifies the above $p$-median problem and related problems that have been discussed in the literature. With each node $v_i$ we associate a nonnegative weight $c_i$, and a real nondecreasing function $f_i$. In the general problem we wish to select a subset $S$ of at most $p$ nodes minimizing the following objective,

$$\sum_{v_j\in S} c_j + \sum_{v_i\in V} \min_{v_j\in S} f_i(d(v_i,v_j)) .$$

In the context of location theory modeling, the weight $c_j$ can be interpreted as the cost of setting up a service center at node $v_j$. The function $f_i$ can be viewed as a transportation cost function, depending on the distance between the customers at $v_i$


*PII* S0167-6377(96)00021-1

and their nearest center. With this interpretation, the objective is to minimize the sum of the setup costs of the centers and the transportation costs of the customers.

When $p = n$, and $f_i$ is linear for each $v_i \in V$, the model reduces to the classical Uncapacitated Facility Location problem, which is solved in $O(n^2)$ time by Cornuejols et al. [1]. Another special case is the $p$-coverage problem discussed by Megiddo et al. [12]. In the $p$-coverage problem each transportation cost function is a stepwise function taking on two values only. The complexity bound of the algorithm in Megiddo et al. [12] is $O(p^2n^2)$.

## 2. The algorithm

Suppose now that the given graph is a tree $T = (V,E)$, which is rooted at some distinguished node, say, $v_1$. For each pair of nodes $v_i$, $v_j$, we say that $v_i$ is a *descendant* of $v_j$ if $v_j$ is on the unique path connecting $v_i$ to the root $v_1$. If $v_i$ is a descendant of $v_j$ and $v_i$ is connected to $v_j$ with an edge, then $v_i$ is a *child* of $v_j$ and $v_j$ is the (unique) *father* of $v_i$. If a node has no children it is called a *leaf* of the tree.

To simplify the notation we assume without loss of generality that $f_j(0) = 0$ for each node $v_j \in V$. It is also convenient to transform the rooted tree into an equivalent binary tree, where each node which is not a leaf has exactly two children. The transformation is executed as follows:

1. First consider each non-leaf node $v_j$ of the original tree which has exactly one child, say $v_{j(1)}$. Introduce a new node, $u_{j(1)}$, and connect it to $v_j$ with a new edge. Assign a weight (length) of zero to this edge. ($u_{j(1)}$ will be the second child of $v_j$ in the new tree. It will also be a leaf of the new tree.)

2. Consider each node $v_j$ of the original tree which has at least three children, say $v_{j(1)},\ldots,v_{j(t)}$, $t \geqslant 3$. Add $t-2$ new nodes, $u_{j(2)},\ldots,u_{j(t-1)}$. For each $s = 2,\ldots,t-1$, replace the edge $(v_j, v_{j(s)})$ by an edge $(u_{j(s)}, v_{j(s)})$ . Also, replace the edge $(v_j, v_{j(t)})$ by an edge $(u_{j(t-1)}, v_{j(t)})$. The weight (length) of a new edge will be the weight of the edge it has replaced. Finally, add the following set of new zero weight (length) edges:

$$(v_j, u_{j(2)}), (u_{j(2)}, u_{j(3)}), \ldots, (u_{j(t-2)}, u_{j(t-1)}).$$

The weight (setup cost) associated with each new node will be sufficiently large, e.g., $1 + \sum_{v_j \in V}(c_j + f_j(\infty))$, and its transportation cost function will be identically zero.

After processing each node $v_j$ of the original tree, the new tree will have at most $2n-3$ nodes and it is binary. Moreover, solving the problem on the original tree is equivalent to solving it on the new binary tree.

In light of the above transformation we now assume without loss of generality that the original tree is a binary tree, where each non-leaf node $v_j$ has exactly two children, $v_{j(1)}$ and $v_{j(2)}$. The former is called the *left child*, and the latter is the *right child*. For each node $v_j$, $V_j$ will denote the set of its descendants, and $T_j$ will be the subtree induced by $V_j$. ($v_j$ is also viewed as the root of $T_j$.)

We are now ready to present the "leaves to root" dynamic programming algorithm.

In a preprocessing step, for each node $v_j$ we compute and sort the distances from $v_j$ to all nodes in $V$. Let this sequence be denoted by $L_j = \{r_j^1,\ldots,r_j^n\}$, where $r_j^i \leqslant r_j^{i+1}$, $i = 1,\ldots,n-1$, and $r_j^1 = 0$. For convenience, in order to handle a degenerate case, where the elements in $L_j$ are not distinct, we assume that there is a one to one correspondence between the elements in $L_j$ and the nodes in $V$, such that:

(i) If $v_k$ corresponds to $r_j^i$ then $r_j^i = d(v_j, v_k)$.

(ii) If $v_k$ and $v_m$ are two distinct nodes in $V_j$, and $v_m$ is a descendant of $v_k$, then the element of $L_j$ representing $v_k$ will precede the one representing $v_m$. In particular, $v_j$ corresponds to $r_j^1$.

(iii) If $v_j$ is not a leaf, and $v_k$ and $v_m$ are both in $V_{j(1)}$ ($V_{j(2)}$), where the element representing $v_k$ in $L_{j(1)}$ ($L_{j(2)}$) precedes the one representing $v_m$, then the element of $L_j$ representing $v_k$ will precede the one representing $v_m$.

(iv) If $v_k$ is in $V_j$, $v_m$ is in $V - V_j$, and $d(v_j, v_k) = d(v_j, v_m)$, then the element of $L_j$ representing $v_k$ will precede the one representing $v_m$.

For $i = 1,\ldots,n$, the node corresponding to $r_j^i$ is denoted by $v_j^i$.

We note that the total effort of the preprocessing step is $O(n^2)$. It can be achieved by using the centroid decomposition approach as in Kim et al. [9]. Alternatively, we can compute the lists $\{L_j\}$ as follows.

For each node $v_j$, let $L_j^+$ ($L_j^-$) be the sorted sequence of distances from $v_j$ to all nodes in $V_j$ ($V - V_j$).

Starting at the leaves of $T$, and proceeding recursively to the root, we first compute the sequences $\{L_j^+\}$.

If $v_j$ is a leaf, $L_j^+ = \{0\}$.

If $v_j$ is not a leaf, consider the sequences $L_{j(1)}^+$ and $L_{j(2)}^+$, associated respectively with $v_{j(1)}$ and $v_{j(2)}$, the two children of $v_j$. Add the distance $d(v_{j(1)}, v_j)$ $(d(v_{j(2)}, v_j))$ to each element in $L_{j(1)}^+$ $(L_{j(2)}^+)$ to obtain the sequence $L_{j(1)}^{++}$ $(L_{j(2)}^{++})$. Then merge $L_{j(1)}^{++}$ with $L_{j(2)}^{++}$, and augment the element zero to the merged sequence to obtain $L_j^+$.

To generate the sequences $\{L_j^-\}$, we start at the root $v_1$, and proceed recursively to the leaves.

For the root $v_1$, $L_1^-$ is empty.

Consider a node $v_j$ which is not the root. Suppose that $v_j$ is a child of $v_k$. Without loss of generality, suppose that $v_j = v_{k(1)}$. Consider the sequences $L_k^-$ and $L_{k(2)}^+$. Add the distance $d(v_{k(1)}, v_k)$ $(d(v_{k(1)}, v_{k(2)}))$ to each element in $L_k^-$ $(L_{k(2)}^+)$ to obtain the sequence $L_k^{-*}$ $(L_{k(2)}^{+*})$. Then, merge the sequences $L_k^{-*}$ and $L_{k(2)}^{+*}$, and augment the element $d(v_{k(1)}, v_k)$ to the merged sequence to obtain $L_j^-$.

It is clear that the total effort to generate the sequences $\{L_j^+\}$ and $\{L_j^-\}$ is $O(n^2)$. Finally, to obtain the sequence $L_j$ for a node $v_j$, merge the respective sequences $L_j^+$ and $L_j^-$.

For each node $v_j$, an integer $q = 1, \ldots, p$, and $r_j^i \in L_j$ let $G(v_j, q, r_j^i)$ be the optimal value of the subproblem defined on the subtree $T_j$, given that a total of at least 1 and at most $q$ nodes (service centers) can be selected in $T_j$, and that at least one of them has to be in $\{v_j^1, v_j^2, \ldots, v_j^i\} \cap V_j$. (In the above subproblem we implicitly assume no interaction between the nodes in $T_j$, and the rest of the nodes in $T$.) The function $G(v_j, q, r)$ is computed only for $q \leqslant |V_j|$. Also, for each node $v_j$ we define

$$G(v_j, 0, r) = \sum_{v_i \in V_j} f_i(\infty).$$

Similarly, for each node $v_j$ and an integer $q = 0, 1, \ldots, p$, we define $F(v_j, q, r)$ to be the optimal value of the subproblem defined on the subtree $T_j$, under the following two constraints:

(i) A total of at most $q$ nodes can be selected in $T_j$.

(ii) There are already some selected nodes (service centers) in $T - T_j$, and the closest amongst them to $v_j$ is at a distance of exactly $r$ from $v_j$. (The setup cost of this closest node is not incorporated into the value of $F(v_j, q, r)$.)

($F(v_j, q, r)$ is computed only for $q \leqslant |V_j|$, and $r = r_j^i$, where $r_j^i$ corresponds to a node $v_j^i$ in $V - V_j$.)

To motivate the above definitions, we note that if all the elements in $L_j$ are distinct, then $G(v_j, q, r_j^i)$ is the optimal value of the subproblem defined on $T_j$, given that at least 1 and at most $q$ nodes are selected in $T_j$, and the closest amongst them to $v_j$ is at a distance of at most $r_j^i$ from $v_j$. The conditions on $L_j$, required above, ensure that the same interpretation of $G(v_j, q, r_j^i)$ can be made for the "distinct" elements of $L_j$, i.e., the elements $r_j^i$, satisfying $r_j^i < r_j^{i+1}$.

The algorithm defines the functions $G$ and $F$ at all leaves of $T$, and then recursively, proceeding from the leaves to the root, computes these functions at all nodes of $T$. The optimal value of the problem will be given by $\min\{G(v_1, p, r_1^n), G(v_1, 0, r_1^n)\}$, where $v_1$ is the root of the tree.

Let $v_j$ be a leaf of $T$. Then,

$$G(v_j, 1, r_j^i) = c_j, \quad i = 1, \ldots, n\ .$$

For each $i = 1, \ldots, n$, such that $v_j^i \in V - V_j$,

$$F(v_j, 0, r_j^i) = f_j(d(v_j, v_j^i)),$$

and

$$F(v_j, 1, r_j^i) = \min\{F(v_j, 0, r_j^i), G(v_j, 1, r_j^i)\}\ .$$

Let $v_j$ be a non-leaf node in $V$, and let $v_{j(1)}$ and $v_{j(2)}$ be its left and right children respectively. The element $r_j^1$ corresponds to $v_j^1 = v_j$, which in turn corresponds to a pair of elements, say $r_{j(1)}^k$ and $r_{j(2)}^t$ in $L_{j(1)}$ and $L_{j(2)}$ respectively. Therefore,

$$G(v_j, q, r_j^1) = c_j + \min_{\substack{q_1+q_2=q-1 \\ q_1 \leqslant |V_{j(1)}| \\ q_2 \leqslant |V_{j(2)}|}} \{F(v_{j(1)}, q_1, r_{j(1)}^k) + F(v_{j(2)}, q_2, r_{j(2)}^t)\}.$$

Generally, for $i = 2, \ldots, n$, consider $r_j^i$. If $r_j^i$ corresponds to a node $v_j^i \in V - V_j$, then

$$G(v_j, q, r_j^i) = G(v_j, q, r_j^{i-1})\ .$$

If $v_j^i \in V_{j(1)}$, then $v_j^i$ corresponds to some element, say $r_{j(1)}^k$ in $L_{j(1)}$, and to some element, say $r_{j(2)}^t$ in $L_{j(2)}$.

Therefore,

$$G(v_j,q,r_j^i) = \min\{G(v_j,q,r_j^{i-1}),\ f_j(r_j^i) + \min_{\substack{1\leqslant q_1\leqslant |V_{j(1)}| \\ q_2\leqslant |V_{j(2)}| \\ q_1+q_2=q}} \times\{G(v_{j(1)},q_1,r_{j(1)}^k) + F(v_{j(2)},q_2,r_{j(2)}^t)\}\}.$$

If $v_j^i \in V_{j(2)}$, then $v_j^i$ corresponds to some element, say $r_{j(1)}^k$ in $L_{j(1)}$, and to some element, say $r_{j(2)}^t$ in $L_{j(2)}$. Therefore,

$$G(v_j,q,r_j^i) = \min\{G(v_j,q,r_j^{i-1}),\ f_j(r_j^i) + \min_{\substack{1\leqslant q_2\leqslant |V_{j(2)}| \\ q_1\leqslant |V_{j(1)}| \\ q_1+q_2=q}} \times\{G(v_{j(2)},q_2,r_{j(2)}^t) + F(v_{j(1)},q_1,r_{j(1)}^k)\}\}.$$

Having defined the function $G$ at $v_j$, we can compute the function $F$ at $v_j$ for all relevant arguments. Let $v_j^i$ be a node in $V - V_j$. Then $v_j^i$ corresponds to some elements, say $r_{j(1)}^k$ and $r_{j(2)}^t$ in $L_{j(1)}$ and $L_{j(2)}$, respectively. Therefore,

$$F(v_j,q,r_j^i) = \min\{G(v_j,q,r_j^i),\ f_j(r_j^i) + \min_{\substack{q_1\leqslant |V_{j(1)}| \\ q_2\leqslant |V_{j(2)}| \\ q_1+q_2=q}} \times\{F(v_{j(1)},q_1,r_{j(1)}^k) + F(v_{j(2)},q_2,r_{j(2)}^t)\}\}.$$

## 3. Complexity of the algorithm

It follows directly from the recursive equations that the total effort to compute the functions $G$ and $F$ at a given node $v_j$, for all relevant values of $q$ and $r$, is $O(n\min\{(|V_{j(1)}|,p)\}\min\{(|V_{j(2)}|,p)\})$.

Therefore, the total effort of the algorithm is clearly $O(p^2n^2)$. We apply a more careful analysis and improve the bound to $O(pn^2)$. We note that our analysis was obtained independently of an almost identical analysis used by Halldórsson et al. [4], to find a best subtree of a tree. (See the next section for a definition of the latter problem.)

The analysis is based on a partition of the node set into two subsets, according to the following definition.

A node $v_j$ is called *rich* if it is not a leaf, and both of its children $v_{j(1)}$ and $v_{j(2)}$ satisfy $|V_{j(1)}| \geqslant p/2$, $|V_{j(2)}| \geqslant p/2$. If a node is not rich it is called *poor*.

**Lemma 1.** *The number of rich nodes is bounded above by* $2n/p$.

**Proof.** Let $V'$ be the set of rich nodes in $V$, and let $T'$ be the minimal subtree of $T$ containing $V'$. For $k = 0,1,2$, let $V'_k$ be the subset of $V'$ consisting of all rich nodes that have exactly $k$ children in $T'$. In particular, $V'_0$ is the set of leaves of $T'$. From the definition of a rich node we have

$$n \geqslant (p+1)|V'_0| + (p/2+1)|V'_1| > p|V'_0| + (p/2)|V'_1|\ . \tag{1}$$

Let $T''$ be the tree obtained from $T'$ by deleting each poor node $v$ in $T'$ which has exactly one child in $T'$, and connecting its father and its child in $T'$ by an edge. $V'_0$ is the set of leaves of $T''$, and $V'_1$ is the set of non-leaf nodes of $T''$ which have exactly one child in $T''$. Let $W$ be the set of nodes of $T''$ which have exactly two children in $T''$, and let $E''$ be the edge set of $T''$. Using the facts that for any graph, the total degree of the nodes equals twice the number of edges, and for any tree, the number of edges is one less than the number of nodes, we obtain $|V'_0|+2|V'_1|+3|W|-1 = 2|E''| = 2(|V'_0|+|V'_1|+|W|-1)$. Thus, $|W| = |V'_0|-1$. Therefore, the total number of nodes in $T''$ is $2|V'_0|+|V'_1|-1$, which implies that

$$|V'| \leqslant 2|V'_0| + |V'_1| - 1\ . \tag{2}$$

The result follows from (1)–(2). □

We have already noted above that the effort to compute the functions $G$ and $F$ at a given node $v_j$ is $O(np^2)$. Therefore, by Lemma 1, the total effort to compute these functions at all rich nodes is $O(pn^2)$. We will now prove that the total effort to compute these functions at all poor nodes is also $O(pn^2)$.

For each node $v_j$ in $V$, let $H_j$ denote the total effort to compute the functions $G$ and $F$ at all poor nodes in

$V_j$. Then, if $v_j$ is rich

$$H_j \leqslant H_{j(1)} + H_{j(2)} \,, \tag{3}$$

and if $v_j$ is poor,

$$H_j \leqslant H_{j(1)} + H_{j(2)} + cn \min\{|V_{j(1)}|, p/2\} \min\{|V_{j(2)}|, p/2\} \,, \tag{4}$$

for some constant $c$.

**Lemma 2.** *For each node $v_j$, $H_j \leqslant cpn|V_j|$.*

**Proof.** We will prove inductively that $H_j \leqslant cn|V_j| \min\{|V_j|, p\}$. If $v_j$ is a leaf, then it clearly follows from the algorithm that $H_j \leqslant cn$, for some constant $c$. Next, consider a node $v_j$ which is not a leaf. Suppose first that $v_j$ is a rich node. Then, $|V_j| = |V_{j(1)}| + |V_{j(2)}| + 1$, $|V_{j(1)}| \geqslant p/2$ and $|V_{j(2)}| \geqslant p/2$. Using the induction hypothesis we obtain from (3)

$$\begin{aligned} H_j &\leqslant H_{j(1)} + H_{j(2)} \leqslant cnp(|V_{j(1)}| + |V_{j(2)}|) \\ &\leqslant cnp|V_j| = cn|V_j| \min\{|V_j|, p\}. \end{aligned}$$

Next, let $v_j$ be a poor node. By the induction hypothesis,

$$H_{j(k)} \leqslant cn|V_{j(k)}| \min\{|V_{j(k)}|, p\}, \quad k = 1,2.$$

Since $v_j$ is a poor node, we may assume without loss of generality that $|V_{j(1)}| \geqslant |V_{j(2)}|$ and $|V_{j(2)}| < p/2$. Consider the following cases:

1. $|V_j| \leqslant p$. In this case we obtain from (4)

$$\begin{aligned} H_j &\leqslant cn(|V_{j(1)}|^2 + |V_{j(2)}|^2) + cn|V_{j(1)}||V_{j(2)}| \\ &\leqslant cn(|V_{j(1)}| + |V_{j(2)}|)^2 \leqslant cn|V_j|^2 \\ &= cn|V_j| \min\{|V_j|, p\} \,. \end{aligned}$$

2. $|V_j| > p$. Since $p/2 > |V_{j(2)}|$, we must have $p/2 \leqslant |V_{j(1)}|$. Therefore,

$$\begin{aligned} H_j &\leqslant cn|V_{j(1)}| \min\{|V_{j(1)}|, p\} \\ &\quad + cn|V_{j(2)}|^2 + cn(p/2)|V_{j(2)}| \\ &\leqslant cnp|V_{j(1)}| + cn(p/2)|V_{j(2)}| + cn(p/2)|V_{j(2)}| \\ &= cnp(|V_{j(1)}| + |V_{j(2)}|) \leqslant cnp|V_j| \\ &= cn|V_j| \min\{|V_j|, p\} \,. \end{aligned}$$

This concludes the proof of the lemma. □

Since $H_1$ is the total effort to compute the functions $G$ and $F$ at all poor nodes in $V$, the above lemmas imply that the complexity of the above dynamic programming algorithm is $O(pn^2)$.

## 4. Related problems

When $p \geqslant n$, the model reduces to the classical Uncapacitated Facility Location problem. Since the bound $p$ is not effective the recursive equations of the algorithm can be simplified to achieve an $O(n^2)$ running time.

We have already noted above that our analysis is very similar to that used by Halldórsson et al. [4] to find the best subtree of a tree. In the latter model, the objective is to find a subtree of a tree of minimum (maximum) total edge weight, containing exactly $p$ edges. An $O(p^2n)$ algorithm to find a best subtree was presented in Maffioli [11] and Fischetti et al. [3]. Halldórsson et al. [4] improved the bound to $O(pn)$. A similar problem, where the selected subtree is required to contain a distinguished node, say $v_1$, is considered by Faigle and Kern [2], and solved in $O(n^4)$ time. The latter model can actually be solved in $O(pn)$ time by the general "left to right" dynamic programming algorithm in Johnson and Niemi [7]. However, the $O(pn)$ "leaves to root" algorithm in Halldórsson et al. [4], does more than that. For each node $v_j$ in the tree, it finds the best subtree of $V_j$ containing $v_j$.

Finally we note that the analysis presented above can also be applied to the "leaves to root" dynamic programming algorithm presented in Tamir and Lowe [13] to solve the generalized $p$-forest problem on trees. With the above analysis, it can be shown that the complexity bound reported there can also be reduced to $O(pn^2)$.

## References

[1] G. Cornuejols, G.L. Nemhauser and L.A. Wolsey, "The uncapacitated facility location problem", in: P.B. Mirchandani and R.L. Francis (eds.), *Discrete Location Theory*, Wiley, New York, 1990, pp. 119–171.

[2] U. Faigle and W. Kern, "Computational complexity of some maximum average weight problems with precedence constraints", *Oper. Res.* **42**, 688–693 (1994).

[3] M. Fischetti, H.W. Hamacher, K. Jornsten and F. Maffioli, "Weighted $k$-cardinality trees: complexity and polyhedral structure", *Networks* **24**, 11–21 (1994).
[4] M.M. Halldórsson, K. Iwano, N. Katoh and T. Tokuyama, "Finding subsets maximizing minimum structures", in: *Proc. 6th Ann. ACM–SIAM symp. on Discrete Algorithms*, San Francisco, California, 1995, pp. 150–159.
[5] R. Hassin and A. Tamir, "Improved complexity bounds for location problems on the real line", *Oper. Res. Lett.* **10**, 395–402 (1991).
[6] W.L. Hsu, "The distance-domination numbers of trees", *Oper. Res. Lett.* **1**, 96–100 (1982).
[7] D.S. Johnson and K.A. Niemi, "On knapsack, partitions, and a new dynamic programming technique for trees", *Math. Oper. Res.* **8**, 1–14 (1983).
[8] O. Kariv and S.L. Hakimi, "An algorithmic approach to network location problems, Part II: $p$-medians", *SIAM J. Appl. Math.* **37**, 539–560 (1979).
[9] T.U. Kim, T.J. Lowe, A. Tamir and J.E. Ward, "On the location of a tree shaped facility", *Networks*, to appear.
[10] J.-H. Lin and J.S. Vitter, "Approximations with minimum packing constraint violation", CS 92–29, Department of Computer Science, Brown University, 1992.
[11] F. Maffioli, "Finding a best subtree of a tree", Report No. 91.041, Dipartimento Di Elettronica, Politecnico Di Milano, 1991.
[12] N. Megiddo, E. Zemel and S.L. Hakimi, "The maximum coverage location problem", *SIAM J. Algebraic Discrete Methods* **4**, 253–261 (1983).
[13] A. Tamir and T.J. Lowe, "The generalized $p$-forest problem on a tree network", *Networks* **22**, 217–230 (1992).