AN ALGORITHMIC APPROACH TO NETWORK LOCATION PROBLEMS.
I: THE p-CENTERS*

O. KARIV† AND S. L. HAKIMI‡

Abstract. Problems of finding p-centers and dominating sets of radius r in networks are discussed in this paper. Let n be the number of vertices and |E| be the number of edges of a network. With the assumption that the distance-matrix of the network is available, we design an O(|E| · n · lg n) algorithm for finding an absolute 1-center of a vertex-weighted network and an O(|E| · n + n² · lg n) algorithm for finding an absolute 1-center of a vertex-unweighted network (the problem of finding a vertex 1-center of a network is trivial). We show that the problem of finding a (vertex or absolute) p-center (for 1 < p < n) of a (vertex-weighted or vertex-unweighted) network, and the problem of finding a dominating set of radius r are NP-hard even in the case where the network has a simple structure (e.g., a planar graph with maximum vertex degree 3). However, we describe an algorithm of complexity O(|E|² · n²/p−1/(p−1)!) for finding an absolute p-center in a vertex-weighted network and an O(|E|² · n²/p−1/(p−1)!) algorithm for finding a vertex p-center in a network. We proceed by discussing the problems of finding p-centers and dominating sets of networks whose underlying graphs are trees. When the network is a vertex-weighted tree, we obtain the following algorithms: An O(n · n · lg n) algorithm for finding the (vertex or absolute) 1-center; an O(n·lg n) algorithm for finding a (vertex or absolute) dominating set of radius r; an O(n² · lg n) algorithm for finding a (vertex or absolute) p-center for any 1 < p < n. Some generalizations of these problems are discussed. When the network is a vertex-unweighted tree, O(n·lg n) algorithms for finding the (vertex or absolute) 1-center and an absolute 2-center are already known; we extend these results by giving an O(n · lg n) algorithm for finding an absolute p-center (where 3 ≤ p < n) and an O(n · lg n) algorithm for finding a vertex p-center (where 2 ≤ p < n).

In part II we treat the p-median problem.

1. Introduction. A network is a connected undirected graph G(V, E) with a nonnegative number w(v) (called the weight of v) associated with each of its |V| = n vertices, and a positive number l(e) (called the length of e) associated with each of its |E| edges. Let X_p = {x_1, x_2, ..., x_p} be a set of p points on G, where by a point on G we mean a point along any edge of G which may or may not be a vertex of G. We define the distance d(v, X_p) between a vertex v of G and a set X_p on G by

\[ d(v, X_p) = \min_{1 \leq i \leq p} \{d(v, x_i)\} \]

where d(v, x_i) is the length of a shortest path in G between vertex v and point x_i. Let

\[ F(X_p) = \max_{v \in V} \{w(v) \cdot d(v, X_p)\}. \]

Let X^*_p be such that

\[ F(X^*_p) = \min_{X_p \text{ on } G} \{F(X_p)\}. \]

Then X^*_p is called an absolute p-center of G and F(X^*_p) is called the absolute p-radius of G and is usually denoted by r_p(G), or simply r_p. If X_p and X^*_p in (1.3) are restricted to be sets of p vertices of G, then X^*_p is called a vertex p-center and F(X^*_p) is called the vertex p-radius of G. Throughout the paper, by "p-center" ("p-radius") of G we shall usually mean "absolute p-center" ("absolute p-radius").

* Received by the editors January 5, 1977 and in revised form October 9, 1978. This work was supported by Air Force Office of Scientific Research, Air Force Systems Command under Grant AFOSR-76-3017.
† Weizmann Institute, Rehovot, Israel. Now at Department of Computer Science, Technion-Israel Institute of Technology, Haifa, Israel.
‡ Department of Electrical Engineering and Computer Science and the Department of Engineering Science and Applied Mathematics, Northwestern University, Evanston, Illinois 60201.
If all the vertices of the network $G(V, E)$ have the same weight $c$, then without loss of generality we shall assume that $c = 1$ and we refer to this case as the vertex-unweighted case. Otherwise, we say that $G(V, E)$ is a vertex-weighted network.

We shall assume that $p < n$, since if $p = n$ then $X_p = V, r_p(G) = 0$, while $p > n$ has no mathematical significance. We further assume as usual and without the loss of generality that graph $G$ contains neither loops nor multiple edges. Finally, we assume that for each edge $e = (v_r, v_s)$ the length of $e$ is equal to the distance between $v_r$ and $v_s$ (i.e., $l(e) = d(v_r, v_s)$); because otherwise, the edge $e$ could be eliminated without affecting the $p$-radius of $G$.

The problem of finding a $p$-center of $G$ was originated by Hakimi [1], [2] and is discussed in a number of papers [3]–[8]. In a recent paper, Hakimi, Schmeichel and Pierce [9] discussed improvements and generalizations of various existing algorithms for finding $p$-centers of networks and gave the corresponding orders of complexity.

The "inverse" of the $p$-center problem is defined as follows: Given a network $G(V, E)$ and a positive integer $r$, find the smallest positive integer $p$ such that the $p$-radius of $G$ is not greater than $r$. This number $p$ is called the (absolute) domination number of radius $r$ of $G$ while a corresponding $p$-center is called an (absolute) dominating set of radius $r$. The vertex domination number of radius $r$ and the vertex dominating set of radius $r$ are similarly defined. If $G$ is a vertex-unweighted graph, all of whose edges are of length 1, then a vertex dominating set of radius 1 and the vertex domination number of radius 1 are referred to simply as a dominating set and the domination number, respectively. The problems of finding the domination number and a dominating set are NP-hard [18]. However, the problem of finding the domination number and a dominating set when $G$ is a tree was solved in linear time by Cockayne, Goodman, and Hedetniemi [19].

Obviously, if one knows how to find a $p$-center, then by performing a binary search over the $n$ possible values of the domination number, one can find a dominating set of radius $r$. On the other hand, information on the domination numbers and on dominating sets can be used to draw conclusions on $p$-centers and the $p$-radii.

In this paper we show that the various algorithms presented in [9] can be further refined leading to significant improvements in their orders of complexity. After introducing some basic notions (§2.1), we describe an $O(|E| \cdot n \cdot \lg n)$ algorithm for finding an absolute 1-center in a vertex-weighted graph (§2.2), and an $O(|E| \cdot n + n^2 \cdot \lg n)$ algorithm for finding an absolute 1-center in a vertex-unweighted graph (§2.3). We assume that the distance-matrix, which gives the distance $d(v_i, v_j)$ between every pair of vertices $v_i$ and $v_j$, is known; actually it takes $O(n^3)$ steps to compute the distance-matrix in a general graph, and thus the complexities of finding absolute 1-centers in a vertex-weighted and a vertex-unweighted network are in fact $O(|E| \cdot n \cdot \lg n + n^3)$ and $O(|E| \cdot n + n^3)$ respectively.\(^1\) With the assumption that the distance-matrix of the network is known, the problem of finding a vertex 1-center becomes trivial. The complexity of finding the distance-matrix has no influence on the complexities of the other algorithms presented in this paper.

Based on results of Garey and Johnson [18] we show that the problem of finding a (vertex or absolute) $p$-center (for $1 < p < n$) of a network, and the problem of finding a (vertex or absolute) dominating set of radius $r$, are NP-equivalent even in the case where the network is a vertex-unweighted planar graph of maximum vertex degree 3, all

---

\(^1\) Since the problem of finding a 1-center of a vertex-unweighted network is equivalent to that of finding a least diameter tree of an edge-weighted graph [4], [5], this gives an $O(|E| \cdot n + n^3)$ algorithm for the latter problem.
of whose edges are of length 1 (§ 3.1). However, we describe an $O((|E|^p \cdot n^{2p-1}/(p-1)) \cdot \lg n)$ algorithm (respectively, an $O(|E|^p \cdot n^{2p-1}/(p-1))$ algorithm) for finding an absolute $p$-center ($1 < p < n$) in a vertex-weighted (respectively, vertex-unweighted) network (§ 3.2). These algorithms are relatively efficient for small values of $p$. For finding a vertex $p$-center, one can use the exhaustive $O(n^{p+1}/(p-1))$ algorithm.

In the remainder of the paper we discuss the case of a network whose underlying graph is a tree, and first we treat the case of a vertex-weighted tree. We begin by presenting an $O(n \cdot \lg n)$ algorithm for finding the (vertex or absolute) 1-center of a vertex-weighted tree (§ 4.1). Then an $O(n)$ algorithm for finding a dominating set of radius $r$ in a tree is given (§ 4.2), and this algorithm implies an $O(n^2 \cdot \lg n)$ algorithm for finding a (vertex or absolute) $p$-center ($1 < p < n$) of a vertex-weighted tree (§ 4.3). Then some generalizations are discussed (§ 4.4). We conclude by treating the case of a network which is a vertex-unweighted tree: First, the $O(n^{p-1})$ algorithm of Hakimi et al. [9] for finding an absolute $p$-center ($3 \leq p < n$) in a vertex-unweighted tree is modified, leading to an $O((n \cdot \lg n)^{p-2})$ algorithm for the same problem, and finally an $O(n \cdot \lg n)$ algorithm for finding a vertex $p$-center ($2 \leq p < n$) in a vertex-unweighted tree is described (§ 5).

Some authors have proposed generalizations of the $p$-center problems. Minieka [20] introduced the notion of general $p$-centers and general $p$-medians which are centers and medians which serve all the points of the graph not merely the vertices. Handler [5] and Halpern [21] defined some combinations of centers and medians. When the network is a tree, both the finding of the "medi-center" as defined by Handler, and the "cent-median" as defined by Halpern involve finding of the 1-center and the 1-median of the tree, for which purpose our algorithms can be used.

The complexity of the algorithms presented in this paper is evaluated according to their asymptotic worst case behavior. No attempt was made to evaluate the average (expected) complexity of these algorithms. It also should be noted that in some practical cases, where the size of the input data is small and the actual behaviors of the algorithms deviate from their asymptotic behaviors, some existing algorithms may be as efficient as the algorithms presented here.

2. 1-center of a network

2.1. The local center. Finding a 1-center of a network $G$ involves finding a "local" center on each edge of $G$ (see, Hakimi's algorithm [1]). A local-center of $G$ on edge $e \in E$ is a point $x^*(e)$ on $e$, such that $F(x^*(e)) = \min_{x(e) \in e} \{F(x(e))\}$. $F(x^*(e))$ is called the local-radius of $G$ on $e$, and it is denoted by $r(e)$. By the definition of 1-center (see (1.2) and (1.3)), it is clear that if

$$r_1 = r(e_1) = \min_{1 \leq i \leq |E|} \{r(e_i)\},$$

then $r_1$ is the 1-radius of $G$ and the corresponding point $x^*(e_1)$ is a 1-center of $G$. Thus, in order to find a 1-center of $G$ it is enough to find a local-center on each of its $|E|$ edges.

To find a local-center of $G$ on edge $e = (v, u)$, we first examine for each vertex $v \in V$ its weighted distance to an arbitrary point $x(e)$ on $e$. If we let $t = t(x(e))$ denote the distance from $x(e)$ to $v$, along edge $e$, then the weighted distance from $v$ to $x(e)$ is

$$D_v(v, t) = w(v) \cdot \min \{t + d(v, u), (l(e) - t + d(v, u))\}.$$ 

The typical plots of $D_v(v, t)$ as it varies with $t$, $0 \leq t \leq l(e)$, are shown in Fig. 1. Each plot consists of either one or two straight line segments, where the slopes of these segments are $\pm w(v)$. Notice that if the plot consists of two segments, then the "break point" (where these segments meet) is the point where $D_v(v, \cdot)$ assumes its maximum.
Next, we define

$$D_e(t) = \max_{v \in V} \{D_e(v, t)\} \quad \text{for } 0 \leq t \leq l(e).$$

A typical plot of $D_e(\cdot)$ is shown in Fig. 2. It is easy to see that any point $x^*(e)$ where $D_e(\cdot)$ assumes its minimum is a local-center on $e$ and the value of $D_e(\cdot)$ at this point is the local-radius on $e$. Furthermore, let $t^* = t(x^*(e))$, (i.e., let $t^*$ be the value of $t$ at the point $x^*(e)$); then it can be shown that $t^*$ is any point on $e$ which satisfies the following two conditions:

(i) Either $t^* = 0$ or $t^* = l(e)$, or $t^*$ is a point where two functions $D_e(v_1, \cdot)$ and $D_e(v_2, \cdot)$ intersect such that their slopes at $t^*$ have opposite signs. [In Fig. 2, the set of points satisfying condition (i) are $t_0^* = 0, t_1^*, t_2^*, t_2^*, t_4^*, t_5^*, t_6^*$, and $t_6^* = l(e)$.]

(ii) If $T^*$ is the set of all points that satisfy condition (i), then $D_e(t^*) = \min_{t \in T^*} \{D_e(t)\}$.

Condition (i) implies that in a graph with $n$ vertices, there are at most $(n(n-1)/2) + 2$ points on edge $e$ which are suspected of being a local-center on $e$. In what follows, we will refer to these points as the "suspected points."

Assuming that the distance-matrix is available, and using relations (2.2) and (2.3), it takes $O(n)$ steps to compute the value of $D_e(\cdot)$ at each of the $O(n^2)$ suspected points. Thus, by a straightforward implementation of condition (ii), a local-center on $e$ can be found in $O(n^3)$ steps. The algorithm of Hakimi, Schmeichel and Pierce [9] finds a local-center in $O(n^2 \cdot \lg n)$ steps. In the next section, we describe an algorithm which finds a local-center on $e$ in $O(n \cdot \lg n)$ steps. Such an algorithm, clearly leads to a procedure for finding a 1-center of $G$ in $O(|E| \cdot n \cdot \lg n)$ steps. For a vertex-unweighted network we present an $O(n)$ algorithm for finding a local-center on an edge, and thus we obtain an $O(|E| \cdot n)$ algorithm for finding a 1-center of the network.

**Remark.** Assuming that the distance-matrix is available, it takes $O(n^2)$ time to find a vertex 1-center of a (vertex-weighted or vertex-unweighted) network. (One should find a maximum entry in each row of the weighted distance-matrix (whose $j$th entry is $w(v_j) \cdot d(v_i, v_j)$) and then find a minimum over these maxima.)
2.2. A 1-center of a vertex-weighted network. To begin with, we shall assume that the vertices of $G$ are ordered such that $w(v_1) \leq w(v_2) \leq \cdots \leq w(v_n)$. This requires a sorting which is achieved in $O(n \cdot \lg n)$ steps.

Let $e$ be an arbitrary edge in $E$. We would like to compute a local-center $x^*(e)$ on $e$. For each integer $i$, $1 \leq i \leq n$, let $D^{(i)}_e(\cdot)$ be the function defined by

$$D^{(i)}_e(t) = \max_{1 \leq i \leq n} \{D_e(v_j, t)\} \quad 0 \leq t \leq l(e).$$

We note that $D^{(n)}_e(t) = D_e(t)$ (see, (2.3)). Each function $D^{(i)}_e(\cdot)$ is piece-wise linear, with the slopes of its line segments belonging to the set $\{\pm w(v_1), \pm w(v_2), \cdots, \pm w(v_i)\}$. Let $L^{(i)} = (t_0^{(i)}, t_1^{(i)}, \cdots, t_n^{(i)})$ be the ordered list of the abscissas of the points where $D^{(i)}_e(\cdot)$ breaks, i.e., $0 = t_0^{(i)} < t_1^{(i)} < \cdots < t_n^{(i)} = l(e)$. Then, for each $j, 1 \leq j \leq n$, the value of $D^{(i)}_e(t)$ for $t \in [t_{j-1}^{(i)}, t_j^{(i)}]$ is given by

$$D^{(i)}_e(t) = D^{(i)}_e(t_{j-1}^{(i)}) + \frac{D^{(i)}_e(t_j^{(i)}) - D^{(i)}_e(t_{j-1}^{(i)})}{t_j^{(i)} - t_{j-1}^{(i)}}(t - t_{j-1}^{(i)}).$$

Thus, the function $D^{(i)}_e(\cdot)$ is fully described by the list $L^{(i)}$ and by its values at the points of $L^{(i)}$.

We denote by $L^{(i)}(t_1, t_2)$ the ordered partial-list of $L^{(i)}$ which contains all the points $t_j^{(i)}$ of $L^{(i)}$ such that $t_1 \leq t_j^{(i)} \leq t_2$.

We shall now describe the derivation of the function $D^{(i+1)}_e(\cdot)$ from the function $D^{(i)}_e(\cdot)$ (i.e., the derivation of the list $L^{(i+1)}$ and the corresponding values of $D^{(i+1)}_e(\cdot)$ at the points of $L^{(i+1)}$, from the list $L^{(i)}$ and the values of $D^{(i)}_e(\cdot)$ at the points of $L^{(i)}$). First we note that by (2.4), $D^{(i+1)}_e(t) = \max\{D^{(i)}_e(t), D_e(v_{i+1}, t)\}$. However, by the order of the vertices, for each $j, 1 \leq j \leq i$, we have $w(v_{i+1}) \leq w(v_j)$, and thus the (absolute) slope of $D^{(i)}_e(v_{i+1}, \cdot)$ is greater than or equal to the (absolute) slope of any segment of $D^{(i)}_e(\cdot)$. Therefore, unless the functions $D_e(v_{i+1}, \cdot)$ and $D^{(i)}_e(\cdot)$ coincide along a whole segment, they can intersect at most at two points. Let $t_{i+1}^{(i)}$ be the point where $D^{(i)}_e(v_{i+1}, \cdot)$ assumes its maximum in the interval $[0, l(e)]$ (thus $t_{i+1}^{(i)}$ is either 0, or $l(e)$, or a point where $D^{(i)}_e(\cdot)$ breaks), and define $\hat{D}_{i+1}$ by the relation:

$$\hat{D}_{i+1} = D_e(v_{i+1}, t_{i+1}^{(i)}) = \max_{t \in [0, l(e)]} D_e(v_{i+1}, t).$$

Then, only one of the following cases could occur:

(a) If $\hat{D}_{i+1} \leq D^{(i)}_e(t_{i+1}^{(i)})$ then clearly for all $t, t \in [0, l(e)]$, $D_e(v_{i+1}, t) \leq D^{(i)}_e(t)$.

Therefore in this case $D^{(i+1)}_e(\cdot) = D^{(i)}_e(\cdot)$ and $L^{(i+1)} = L^{(i)}$.

(b) If $\hat{D}_{i+1} > D^{(i)}_e(t_{i+1}^{(i)})$, we consider the two closed intervals $[0, t_{i+1}^{(i)}]$ and $[t_{i+1}^{(i)}, l(e)]$ separately in each of the following two subcases:

(i) If $D_e(v_{i+1}, \cdot)$ does not intersect $D^{(i)}_e(\cdot)$ in the interval $[0, t_{i+1}^{(i)}]$ (respectively, interval $[t_{i+1}^{(i)}, l(e)]$), then in this interval $D^{(i+1)}_e(t) = D_e(v_{i+1}, t)$ and $L^{(i+1)}(0, t_{i+1}^{(i)}) = (0, t_{i+1}^{(i)})$ (respectively, $L^{(i+1)}(t_{i+1}^{(i)}, l(e)) = (t_{i+1}^{(i)}, l(e))$).

(ii) If $D_e(v_{i+1}, \cdot)$ intersects $D^{(i)}_e(\cdot)$ in the interval $[0, t_{i+1}^{(i)}]$ (respectively, interval $[t_{i+1}^{(i)}, l(e)]$), then it either intersects $D^{(i)}_e(\cdot)$ at a single point $t_{i+1}^{(i)}$ (respectively, $t_{i+1}^{(i)}$) or it coincides with $D^{(i)}_e(\cdot)$ along a line segment. In the latter case let $t_{i+1}$ be the smallest value $t$ in the interval $[0, t_{i+1}^{(i)}]$ such that $D_e(v_{i+1}, t) = D^{(i)}_e(t)$ (respectively, $t_{i+1}$ is the greatest value $t$ in the interval $[t_{i+1}^{(i)}, l(e)]$ such that $D_e(v_{i+1}, t) = D^{(i)}_e(t)$). In both cases, whether $D_e(v_{i+1}, \cdot)$ and $D^{(i)}_e(\cdot)$ intersect at one point or along a line segment, we have:

$$D^{(i+1)}_e(t) = \begin{cases} D^{(i)}_e(t) & \text{for } 0 \leq t \leq t_{i+1} \\ D_e(v_{i+1}, t) & \text{for } t_{i+1} \leq t \leq t_{i+1}^{(i)} \end{cases}$$
and

\[ L^{(i+1)}(0, \hat{t}_{i+1}) = L^{(i)}(0, \hat{t}_{i+1}) \parallel (\hat{t}_{i+1}, \hat{t}_{i+1}) \]

(respectively,

\[ D_e^{(i+1)}(t) = \begin{cases} D_e(v_{i+1}, t) & \text{for } \hat{t}_{i+1} \leq t \leq \hat{t}_{i+1}^+ \\ D_e^{(i)}(t) & \text{for } \hat{t}_{i+1}^+ \leq t \leq l(e) \end{cases} \]

and

\[ L^{(i+1)}(\hat{t}_{i+1}, l(e)) = (\hat{t}_{i+1}, \hat{t}_{i+1}) \parallel L^{(i)}(\hat{t}_{i+1}^+, l(e)) \].

**Data structure.** Before we formally state the algorithm, we describe the data structure to be used. An ordered 2–3 tree [12] is used to store at each stage the ordered list \( L^{(i)} \). Thus, each vertex of the 2–3 tree contains a value \( t \), where \( t \) is a point of the interval \([0, l(e)]\) in which the appropriate function \( D_e^{(i)}(\cdot) \) breaks. For each such point \( t \), we attach a variable \( D(t) \) which gives the actual value of the function \( D_e^{(i)}(\cdot) \) at the point \( t \). As it was mentioned above, this data structure gives a full description of the function \( D_e^{(i)}(\cdot) \) using (2.5). We note that each time function \( D_e^{(i+1)}(\cdot) \) is derived from \( D_e^{(i)}(\cdot) \), at most three points \( (\hat{t}_{i+1}^+, \hat{t}_{i+1}, \hat{t}_{i+1}^-) \) can be added to the ordered list \( L^{(i)} \). Thus, at no stage does the list \( L^{(i)} \) contain more than \( 3n \) points, and the depth of the 2–3 tree which describes the list \( L^{(i)} \) is at most of order \( O(\log n) \). Therefore each of the operations INSERT, DELETE, and FIND which are executed on the 2–3 tree is performed in no more than \( O(\log n) \) steps.

In the following algorithm we denote the 2–3 tree by \( L \). The initial 2–3 tree contains the two points 0 and \( l(e) \) where the initial values of \( D(\cdot) \) are assumed to be 0. Operations on \( L \) are written in capital letters. We assume that the vertices of the graph are listed according to a nondecreasing order of their weights.

**Algorithm 2.1.** Local-center in a vertex-weighted network.

1. [Initialization]. Assign: \( i \leftarrow 0, L(0, l(e)), D(0) \leftarrow 0, D(l(e)) \leftarrow 0 \).
2. [Proceed to the next vertex]. If \( i = n \) then go to 10. Else, assign \( i \leftarrow i + 1 \).
3. Find the point \( \hat{t} \) such that \( D_e(v_i, \hat{t}) = \max_{0 \leq t \leq l(e)} \{D_e(v_i, t)\} \) [\( \hat{t} \) is either 0 or \( l(e) \) or the point where \( D_e(v_i, \cdot) \) breaks].
4. FIND in \( L \) the point \( t^- \) such that \( t^- = \min \{t \mid L(t) \leq s^-\} \). FIND and assign: \( t^- \leftarrow \hat{t} \).
5. [Treatment of the point \( \hat{t} \)]. If \( t^+ = \hat{t} \) then assign: \( D(\hat{t}) \leftarrow D_e(v_i, \hat{t}) \) and go to 6. Else \( t^- < \hat{t} < t^+ \), INSERT \( \hat{t} \) into \( L \) and assign: \( D(\hat{t}) \leftarrow D_e(v_i, \hat{t}), t^- \leftarrow \hat{t}, t^+ \leftarrow \hat{t} \).
6. [Treatment of the interval \((0, \hat{t})\)]. If \( t^- = 0 \) then go to 7. Else, assign: \( s^- \leftarrow t^- [s^- \text{ is a temporary storage}] \). FIND and assign: \( t^- \leftarrow \max \{t \mid L(t) \leq s^-\} \). If \( D(t^-) < D_e(v_i, t^-) \) then DELETE \( t^- \) from \( L \) and repeat 6. If \( D(t^-) = D_e(v_i, t^-) \), then go to 8. If \( D(t^-) > D_e(v_i, t^-) \), then find the point \( t_0 \) in the interval \((t^-, s^-)\) where \( D(\cdot) \) and \( D_e(v_i, \cdot) \) intersect. INSERT the point \( t_0 \) into \( L \), assign: \( D(t_0) \leftarrow D_e(v_i, t_0) \) and go to 8.
7. [Treatment of the point 0]. If \( \hat{t} = 0 \) then go to 8. Else, INSERT the point \( t = 0 \) into \( L \) and assign: \( D(0) \leftarrow D_e(v_i, 0) \).
8. [Treatment of the interval \((\hat{t}, l(e))\)]. If \( t^+ = l(e) \) then go to 9. Else, assign: \( s^+ \leftarrow t^+ [s^+ \text{ is a temporary storage}] \). FIND and assign: \( t^+ \leftarrow \min \{t \mid L(t) \geq s^+\} \). If \( D(t^+) < D_e(v_i, t^+) \) then DELETE \( t^+ \) from \( L \) and repeat 8. If \( D(t^+) = D_e(v_i, t^+) \) then repeat 8.

\(^2\) It should be understood that if in this expression the point \( t_{i+1}^- \) (resp. \( t_{i+1}^+ \)) appears more than once, all but one appearance must be omitted. The operation \( \parallel \) means concatenation of two lists.
then return to 2. If $D(t^*) > D_e(v_i, t^*)$ then (by (2.5)) find the point $t_0$ (in the interval $(s^+, t_0)$) where $D(\cdot)$ and $D_e(v_i, \cdot)$ intersect. INSERT the point $t_0$ into $L$, assign: $D(t_0) = D_e(v_i, t_0)$ and return to 2.

9. [Treatment of the point $(l(e))$. If $l = l(e)$ then return to 2. Else, INSERT the point $l(e)$ into $L$, assign: $D(l(e)) = D_e(v_i, l(e))$ and return to 2.

10. Find in $L$ a point $t^*$ for which $D(\cdot)$ is the minimum. Each such point is a local-center on $e$, and the value $D(t^*)$ is the local-radius.

Each of the operations INSERT, DELETE and FIND requires at most $O(\log n)$ steps. The operation INSERT is performed at most five times per vertex (steps 5, 6, 7, 8, 9) and thus it is performed at most $5n$ times. Clearly the operation DELETE is not performed more often than the operation INSERT. The operation FIND is performed once before each DELETE, plus at most four times per vertex (twice in step 4, and once before we leave each of steps 6 and 8)—a total of $O(n)$ times. Thus, the complexity of Algorithm 2.1 (finding a local-center on an edge) is $O(n \cdot \log n)$.

**Algorithm 2.2. 1-center of a vertex-weighted network.**
1. Arrange the vertices of the network according to nondecreasing order of their weights.
2. For each edge $e$ of the network apply Algorithm 2.1 to find the local-radius on $e$ and a local-center.
3. The minimum local-radius on the edges of the network is the 1-radius of the network. Any local-center having the minimum local-radius is a 1-center.

The complexity of Algorithm 2.2 is $O(|E| \cdot n \cdot \log n)$, provided the distance-matrix of the graph is available.

### 2.3. A 1-center of a vertex-unweighted network

In this section, we present an algorithm for finding a 1-center of a vertex-unweighted network $G(V, E)$, namely a network in which $w(v) = c$ for all $v \in V$. We shall lose no generality by assuming $c = 1$.

As before, we begin by giving an algorithm for finding a local-center on an arbitrary edge $e \in E$ of the network $G(V, E)$. We shall see that this algorithm has complexity $O(n)$ in contrast with Hakimi, Schmeichel, and Pierce’s algorithm [9], which requires $O(n \cdot \log n)$ steps. Thus, assuming that the distance matrix is available, one can find a 1-center of the network in $O(|E| \cdot n)$ steps.\(^3\)

As in the previous case, a local-center $x^*(e)$ on edge $e = (v_i, v_s)$ is either at the end-points $v_i$ or $v_s$ of the edge, or at a point $t^*(t^* \in [0, l(e)])$ where two functions $D_e(v_i, \cdot)$ and $D_e(v_s, \cdot)$ intersect. Notice that due to the equal weights of the vertices, all the functions $D_e(v_i, \cdot)$, $i = 1, 2, \ldots, n$, have the same (absolute) slope, and thus any two functions $D_e(v_i, \cdot)$ and $D_e(v_s, \cdot)$ which do not coincide along a whole line segment can intersect at most at one point, where at this point these two functions have slopes of opposite signs.

For each vertex $v_i \in V$, we define:

$$V_i = \{v \in V | D_e(v, 0) \leq D_e(v_i, 0)\}; \quad \bar{V}_i = V - V_i.$$

If $\bar{V}_i \neq \emptyset$, we also define a vertex $\bar{v}_i$ by:

$$D_e(\bar{v}_i, l(e)) = \max_{v \in \bar{V}_i} \{D_e(v, l(e))\}.$$

\(^3\) In fact, our algorithm requires a preprocessing of $O(n^2 \cdot \log n)$ steps. Therefore, in case of sparse graphs (when $|E| \leq O(n \cdot \log n)$), the complexity of our algorithm is rather $O(n^2 \cdot \log n)$ than $O(|E| \cdot n)$. However, even if $|E| = O(n)$ (as in planar graphs) our algorithm is no worse than that of Hakimi et al.
(If there are more than one vertex which satisfy (2.8) we choose the one with minimum
index to be \(v_i\)).

Using (2.7) and (2.8), we can prove the following lemmas:

**Lemma 2.1.** Let \(t_o\) be the abscissa of the point where two functions \(D_e(v_i, \cdot)\) and
\(D_e(v_j, \cdot)\) intersect, such that at the point \(t_o\), the function \(D_e(v_i, \cdot)\) has a positive slope and,
thus, \(D_e(v_j, \cdot)\) has a negative slope. Then \(D_e(v_i, t) < D_e(v_j, t)\) for \(0 \leq t < t_o\), and \(D_e(v_i, t) > D_e(v_j, t)\) for \(t_o < t \leq l(e)\).

**Proof.** The proof is implied directly by the fact that the slopes of both functions
\(D_e(v_i, \cdot)\) and \(D_e(v_j, \cdot)\) are \(\pm 1\).

**Corollary 2.1.** Under the conditions of Lemma 2.1, \(v_i \in \tilde{V}_n, v_j \in V_f\).

**Lemma 2.2.** A local-center of \(G\) on edge \(e\) is either at the end-points of \(e\) (namely, at
\(t = 0\) or at \(t = l(e)\)), or it is at some point \(t_i\), where \(t_i\) is the abscissa of the intersection point of
two functions, \(D_e(v_i, \cdot)\) and \(D_e(v_j, \cdot)\).

**Proof.** The fact that a local-center could be at the points \(0\) or \(l(e)\) is obvious.
Assume a local-center on \(e\) is at a point \(t^*\), where \(t^* \neq 0, l(e)\). As before, \(t^*\) is the
intersection point of two functions \(D_e(v_i, \cdot)\) and \(D_e(v_j, \cdot)\) whose slopes at the point \(t^*\)
have opposite signs. Without loss of generality, assume that at \(t^*\), \(D_e(v_i, \cdot)\) has a positive
slope; then by Corollary 2.1 this implies that \(v_j \in \tilde{V}_i\). Thus, \(\tilde{V}_i \neq \emptyset\) and \(v_i\) is defined. By
the definition of a local-center, \(D_e(v_i, t^*) \leq D_e(v_j, t^*)\). If the equality is satisfied, then the
lemma holds. Therefore, assume that \(D_e(v_i, t^*) < D_e(v_j, t^*)\), which implies that
\(D_e(v_i, t^*) < D_e(v_j, t^*)\). If at \(t^*\) the function \(D_e(v_i, \cdot)\) has a positive slope, then \(D_e(v_i, t^*) < D_e(v_j, t^*)\)
implies \(D_e(v_j, 0) < D_e(v_i, 0)\) which contradicts the choice of \(v_i\) (2.7) and (2.8). If at \(t^*\)
the function \(D_e(v_j, \cdot)\) has a negative slope, then \(D_e(v_i, t^*) < D_e(v_j, t^*)\) implies
\(D_e(v_i, t^*) < D_e(v_j, t^*)\) which contradicts (2.8). Thus, the assumption \(D_e(v_i, t^*) < D_e(v_j, t^*)\) is not correct; hence \(D_e(v_i, t^*) = D_e(v_j, t^*)\), and the lemma holds. **Q.E.D.**

**Remark.** Lemma 2.2 reduces the number of possible candidates for a local-center
on \(e\) to at most \(n + 2\).

**Lemma 2.3.** If the functions \(D_e(v_i, \cdot)\) and \(D_e(v_j, \cdot)\) intersect, let \(t_i\) be the abscissa of
their intersection point. Then, \(D_e(v, t_i) = \max_{v \in V} \{D_e(v, t_i)\}\) (or in short, \(D_e(t_i) = D_e(v_i, t_i)\)).

**Proof.** Assume that the lemma does not hold and there exists a vertex \(v'\) such that
\(D_e(v_i, t_i) < D_e(v', t_i)\). Since at the point \(t_i\) the function \(D_e(v_i, t_i)\) has a positive slope, that
inequality implies \(D_e(v_i, 0) < D_e(v', 0)\) —namely: \(v' \in \tilde{V}_i\). On the other hand \(D_e(v_i, t_i) < D_e(v', t_i)\)
implies \(D_e(v_i, t_i) < D_e(v', t_i)\), and since at the point \(t_i\) the function \(\tilde{v}\) has a negative
slope we have \(D_e(v_i, l(e)) < D_e(v', l(e))\) which is in contradiction to the
definition of \(\tilde{v}\) (2.8). **Q.E.D.**

As a direct consequence of Lemmas 2.2 and 2.3, we have

**Corollary 2.2.** Let \(t_0 = 0, t_{n+1} = l(e)\) and for those values of \(i\), \(1 \leq i \leq n\), where
\(D_e(v_i, \cdot)\) and \(D_e(v_j, \cdot)\) intersect, define \(t_i\) to be the point of intersection. Let \(D_0 = \max_{v \in V} \{D_e(v, 0)\}, D_{n+1} = \max_{v \in V} \{D_e(v, l(e))\}\) and for those values of \(i\), \(1 \leq i \leq n\),
where \(t_i\) is defined, denote \(D_i = D_e(v_i, t_i)\). Let \(j\) be an index for which \(D_j = \min_{0 \leq i \leq n} \{D_i|D_i|\}\) is defined. Then \(D_j\) is the local-radius on \(e\) and the corresponding \(t_j\) is a
local-center on \(e\).

**Lemma 2.4.** If \(v_i\) and \(v_j\) are two vertices such that \(D_e(v_i, 0) = D_e(v_j, 0)\) and
\(D_e(v_i, l(e)) = D_e(v_j, l(e))\), then in Corollary 2.2, the point \(t_i\) (where \(D_e(v_i, \cdot)\) and \(D_e(v_j, \cdot)\)
intersect) need not be considered.

**Proof.** By the conditions of the lemma we have (for every \(t\), \(0 \leq t \leq l(e)\)), \(D_e(v_i, t) \equiv D_e(v_j, t)\). Suppose that \(t_i\) is a local-center on \(e\), then the definition of local-center implies that
\(D_e(v_i, t_i) \equiv D_e(v_j, t_i)\). Thus, we obtain \(D_e(v_i, t_i) = D_e(v_j, t_i)\). On the other hand, definition (2.8) implies that \(\tilde{v} = \tilde{v}\) and thus \(D_e(\tilde{v}, t_i) = D_e(\tilde{v}, t_i)\). Therefore the inter-
section point \( t_i \) of \( D_e(v_i, \cdot) \) and \( D_e(\tilde{v}_i, \cdot) \) is also the intersection point \( t_i \) of \( D_e(v_i, \cdot) \) and \( D_e(\tilde{v}_i, \cdot) \), and thus in Corollary 2.2 it is sufficient to consider the point \( t_i \) while \( t_j \) need not be considered.

The following algorithm for finding a local-center on edge \( e \) in a vertex-unweighted network is based on Corollary 2.2 as amended by Lemma 2.4. As a pre-procedure for the algorithm, we construct for each \( v \in V \), a list \( L(v) \) in which all the vertices of the network are listed according to a nonincreasing order of their distances from \( v \) (notice that the vertex \( v \) itself is the last vertex in the list \( L(v) \)). On the assumption that the distance-matrix of the network is available, the construction of each such list requires \( O(n \cdot \log n) \) steps, and thus it would take \( O(n^2 \cdot \log n) \) steps to construct all the lists for all of the vertices.

Let \( e = (v_n, v_1) \), and let \( v_i \) be the \( i \)th vertex in the list \( L(v_i) \). Thus: \( D_e(v_1, 0) \geq D_e(v_2, 0) \geq \cdots \geq D_e(v_n-1, 0) > D_e(v_n, 0) = 0 \) (notice that by this notation, vertex \( v_n \), an end-point of \( e \), is denoted by \( v_n \)). The following algorithm for finding a local-center on \( e \) is performed in stages, where in the \( i \)th stage we examine the functions \( D_e(v_i, \cdot) \) and \( D_e(\tilde{v}_i, \cdot) \), and find their intersection point \( t_i \) if necessary. Notice that if for some \( i \) and \( j \) (\( 1 < j \leq k < n \)), we have \( D_e(v_{i-1}, 0) > D_e(v_i, 0) = \cdots = D_e(v_k, 0) > D_e(v_{k+1}, 0) \), then by (2.7) and (2.8), \( \tilde{v}_i = \tilde{v}_{i+1} = \cdots = \tilde{v}_k \). Furthermore, if \( v_m \) (where \( j \leq m \leq k \)) is a vertex of minimum index such that \( D_e(v_m, l(e)) = \max_{1 \leq i \leq k} \{D_e(v_i, l(e))\} \), then Lemma 2.4 implies that out of the whole set of vertices \( \{v_{j-1}, v_{j+1}, \cdots, v_k\} \), only the vertex \( v_m \) should be considered (namely, only the point \( t_m \) where \( D_e(v_m, \cdot) \) and \( D_e(\tilde{v}_m, \cdot) \) intersect, is suspected as being a local-center). Moreover, by (2.7) and (2.8) and by the special order of the list \( L(v_i) \), it follows that the vertex \( \tilde{v}_{k+1} \) is that vertex (of the two vertices \( \tilde{v}_i \) and \( v_{m} \)) which satisfies:

\[
D_e(\tilde{v}_{k+1}, l(e)) = \max \{D_e(\tilde{v}_i, l(e)), D_e(v_m, l(e))\}.
\]

Let \( k \) be the integer such that \( D_e(v_1, 0) = \cdots = D_e(v_k, 0) > D_e(v_{k+1}, 0) \) (clearly, \( 1 \leq k < n \)). Since (2.7) and (2.8) imply that the vertices \( \tilde{v}_1, \cdots, \tilde{v}_k \) are not defined, these vertices must not be considered in Corollary 2.2. Notice, however, that if we define \( v_m \) as before (namely, \( v_m \), where \( 1 \leq m \leq k \), is a vertex of minimum index such that \( D_e(v_m, l(e)) = \max_{1 \leq i \leq k} \{D_e(v_i, l(e))\} \)), then the special order of the list \( L(v_i) \) implies that \( \tilde{v}_{k+1} = v_m \).

In the following algorithm, the variables \( r^* \) and \( r \) represent a local-center and the local-radius on edge \( e = (v_n, v_1) \) respectively. The vertices \( v_i \) and \( \tilde{v}_i \) are represented throughout the algorithm by variables \( v^* \) and \( \tilde{v}^* \) respectively.

**Algorithm 2.3.** Local-center in a vertex-unweighted network.

1. [Treatment of the points \( t = 0 \) and \( t = l(e) \).] Let \( v^* \) be the first vertex of the list \( L(v_i) \) and let \( \tilde{v} \) be the first vertex of the list \( L(v_i) \). If \( D_e(v^*, 0) = D_e(\tilde{v}, l(e)) \) then assign: \( r^* \leftarrow 0, r \leftarrow D_e(v^*, 0) \); else, assign: \( r^* \leftarrow D_e(\tilde{v}, l(e)) \). If \( v^* \neq \tilde{v} \) then Halt. [For every \( v \in V \) and \( t \in [0, l(e)] \), \( D_e(v, t) \leq D_e(v^*, t) \), and thus \( D_e(v^*, \cdot) = D_e(\cdot, \cdot) \) and \( r^* \) and \( r \) are a local-center and the local-radius on \( e \), respectively.]

2. [Initialization of the stages.] Assign: \( i \leftarrow 1, v_m \leftarrow v^* \).

3. [Treatment of all the vertices \( v \) such that \( D_e(v, 0) = D_e(v_1, 0) \).] Assign: \( i \leftarrow i + 1 \). Let \( v^* \) be the \( i \)th vertex in the list \( L(v_i) \). If \( D_e(v^*, 0) = D_e(v_m, 0) \) then go to 4. Else, if \( D_e(v^*, l(e)) > D_e(v_m, l(e)) \) assign: \( v_m \leftarrow v^* \). Repeat 3. [Notice that since \( D_e(v_1, 0) = D_e(v_{m}, 0), i < n \).]

4. Assign: \( \tilde{v}^* \leftarrow v_m \). If \( i = n \) then go to 8. Else assign: \( v_m \leftarrow v^* \) and go to 5.

5. [Find all the vertices \( v \) such that \( D_e(v, 0) = D_e(v_1, 0) \), and find the corresponding \( v_m \).] Assign: \( i \leftarrow i + 1 \). Let \( v^* \) be the \( i \)th vertex in the list \( L(v_i) \). If \( D_e(v^*, 0) \neq D_e(v_m, 0) \) then
$D_e(v_{m}, 0)$ then go to 6. Else, if $D_e(v^{*}, l(e)) > D_e(v_{m}, l(e))$ assign: $v_m \leftarrow v^{*}$. Repeat 5. [Notice that since $D_e(v_{i}, 0) = D_e(v_{i-1}, 0)$, $i < n$.]

6. [Treatment of the point $t_m$.] If the functions $D_e(v_m, \cdot)$ and $D_e(v^{*}, \cdot)$ do not intersect, or if they coincide along a whole line-segment, then go to 7. Else, let $t_m$ be the point where $D_e(v_m, \cdot)$ and $D_e(v^{*}, \cdot)$ intersect. If $D_e(v_m, t_m) < r$ then assign: $t^{*} \leftarrow t_m, r \leftarrow D_e(v_m, t_m)$.

7. [Proceed to the next vertex.] If $D_e(v_{m}, l(e)) > D_e(v^{*}, l(e))$, then assign $v^{*} \leftarrow v_m$. If $i = n$ then go to 8. Else assign $v_m \leftarrow v^{*}$ and go to 5.

8. [Treatment of the vertex $v_n$.] If the functions $D_e(v^{*}, \cdot)$ and $D_e(t^{*}, \cdot)$ do not intersect, or if they coincide along a whole line-segment, then Halt. Else, let $t_m$ be the point where $D_e(v^{*}, \cdot)$ and $D_e(t^{*}, \cdot)$ intersect. If $D_e(v^{*}, t_m) > r$ then Halt. Else, assign: $t^{*} \leftarrow t_m, r \leftarrow D_e(v^{*}, t_m)$ and Halt.

Provided the distance-matrix and the lists $L(v_r)$ and $L(v_s)$ are available, it is easy to see that the complexity of Algorithm 2.3 is $O(n)$.

**Algorithm 2.4.** 1-center of a vertex-unweighted network.

1. For each vertex $v \in V$, construct a list $L(v)$ of all the vertices of the graph arranged according to a nonincreasing order of their distances from $v$.
2. For each edge $e$ of the network apply Algorithm 2.3 to find the local-radius on $e$ and a local-center.
3. The minimum local-radius on the edges of the network is the 1-radius of the network. Any local-center having the minimum local-radius is a 1-center.

Step 1 of Algorithm 2.4 requires $O(n^2 \cdot \lg n)$ operations, while step 2 requires $O(|E| \cdot n)$. Thus, provided the distance-matrix is available, the complexity of Algorithm 2.4 is $O(|E| \cdot n + n^2 \cdot \lg n)$.

**3. p-center ($p > 1$) of a network.** In the next two sections we treat the problems of finding a $p$-center ($p > 1$) of a vertex-weighted and a vertex-unweighted network. In §3.1, we prove that the NP-completeness of the dominating set problem implies that the problems of finding (absolute or vertex) $p$-centers are NP-complete even when the network is a planar graph of maximum vertex degree 3. In §3.2, however, we describe an algorithm for the (absolute) $p$-center problem which although exponential in $p$, is polynomial in $n$, and thus is relatively efficient for small values of $p$. Unfortunately, it seems that similar ideas cannot lead to reduction of the time bound which is set by the direct (exhaustive) algorithm for the vertex $p$-center problem.

**3.1. The general p-center problem is NP-complete.** We now show that the general $p$-center problems (i.e., finding an absolute $p$-center or a vertex $p$-center for a vertex-weighted or a vertex-unweighted network) belong to the family of combinatorial problems which are NP-complete [10], [12] (see the remarks in the next paragraph). This implies that it is highly unlikely that there exist $O(f(n, p))$ algorithms for the $p$-center problems where $f(n, p)$ is a polynomial function in each of the two variables. (In fact, we show that the $p$-center problems as well as the dominating set problems are NP-complete, even when the network is a planar graph of maximum vertex degree 3.)

[Actually, by the definition of NP-completeness [10], [12], a problem can be "NP-complete" only if in the first place it belongs to the set NP, namely, only if it is a decision problem for which a positive answer ("Yes") can be found in polynomial time by a nondeterministic Turing machine. Clearly, according to this definition, the $p$-center problem, which is an optimization problem, cannot be considered an "NP-complete problem." However, we shall prove that the $p$-center problem is an "NP-hard" problem, namely, we shall prove that the dominating set problem which is an NP-complete problem, is polynomial time reducible to the $p$-center problem, and]
therefore, there exists a (deterministic) polynomial time algorithm for the p-center problem only if \( P = NP \). Moreover, based on some results of the next section, we shall show that the p-center problem is in fact \("NP-equivalent," \) namely, that it is (also) polynomial time reducible to some problem in \( NP \), and therefore, there is a polynomial-time solution for the p-center problem if (and only if) \( P = NP \).

**Lemma 3.1.** Let \( G(V, E) \) be a graph and let \( p \) be an integer, \( 1 < p < n \). The problem of whether there exists in \( G \) a dominating set of cardinality \( \leq p \) (i.e., whether the domination number of \( G \) is \( \leq p \)) is \( NP \)-complete even in the case when \( G \) is a planar graph of maximum vertex degree 3.

*Proof.* The proof of Lemma 3.1 was given by Garey and Johnson [18] and is based on their previous result about the \( NP \)-completeness of the vertex cover problem. The vertex cover problem is defined as follows: Given a graph \( G(V, E) \) and an integer \( k \), find a subset \( V' \) of the vertices of \( G \) such that \( |V'| \leq k \) and each edge of \( G \) is incident at a vertex of \( V' \). While the general vertex cover problem is known to be \( NP \)-complete [10], Garey and Johnson have shown that the problem is \( NP \)-complete even when \( G \) is a planar graph of maximum vertex degree 3 [17]. Lemma 3.1 is proved by performing the following reduction from the vertex cover problem to the dominating set problem:

Replace each edge \( XY \) of \( G \) by \( XY \) and set \( p = |E| + 1 \).

**Theorem 3.1.** The problems of finding a vertex p-center and an absolute p-center are \( NP \)-hard even in the case when the network is vertex-unweighted planar graph of maximum degree 3, all whose edges are of length 1.

*Proof.* We first observe that the \( NP \)-complete problem of Lemma 3.1 is polynomial time reducible to the problem of finding a vertex p-center under the conditions of Theorem 3.1. For, suppose that we can find a vertex p-center \( V^*_p \) and the vertex p-radius \( r^{(e)}_p \) of \( G \). Then there exists a dominating set of cardinality \( \leq p \) if and only if \( r^{(e)}_p \leq 1 \). Thus, the vertex p-center problem is \( NP \)-hard.

We now show that the absolute p-center problem is \( NP \)-hard by similar arguments: Assume that under the conditions of Theorem 3.1 we can find an absolute p-center \( X^*_p \) and the absolute p-radius \( r_p \). Clearly, if \( r_p > 1 \), then there exists in the graph no dominating set of cardinality \( \leq p \). On the other hand, if \( r_p \leq 1 \), then by replacing each nonvertex point of \( X^*_p \) by the closest vertex to it (where ties are broken arbitrarily), we obtain a vertex p-center of radius \( \leq 1 \) — i.e., there exists a dominating set of cardinality \( \leq p \). Thus, the problem of Lemma 3.1 is polynomial time reducible to the problem of finding an absolute p-center, which means that the latter problem is \( NP \)-hard.

**Corollary 3.1.** The problem of finding a vertex or an absolute dominating set of radius \( r \) is \( NP \)-hard even in the case when the network is a vertex-unweighted planar graph of maximum vertex degree 3, all whose edges are of length 1.

The following theorem concludes the discussion on the \( NP \)-completeness of the p-center problems.

**Theorem 3.2.** The problems of finding a vertex p-center and an absolute p-center of a network are \( NP \)-equivalent.

*Proof.* Since by Theorem 3.1 the p-center problems are \( NP \)-hard, it remains to show that if \( P = NP \), then there exist (deterministic) polynomial-time algorithms which solve these problems.

---

4 The term "\( NP \)-equivalent" was suggested to the authors by David S. Johnson [18]. The authors wish to thank D. Johnson for his illuminative remarks on this section which helped to clarify the terminology and to remove possible confusion.
We first prove this assertion for the vertex $p$-center problem. We note that a priori there are at most $O(n^2)$ possible values for the vertex $p$-radius—these are exactly all the elements $r_{ij}$ of the (weighted) distance-matrix of the graph. It is not difficult to build a nondeterministic Turing machine which finds in polynomial time for each such possible value $r_{ij}$ whether there exists a subset $V_p$ of $p$ vertices of the graph such that for each vertex $v \in V$, $w(v) \cdot d(v, V_p) \leq r_{ij}$. Hence, if $P = NP$, then a vertex $p$-center can be found in (deterministic) polynomial time.

For the absolute $p$-center problem, we use a result of the next section which shows that without loss of generality, we can restrict the points of an absolute $p$-center $X^*_p$ of a network $G$ to be chosen out of a set $X$ of no more than $O(|E| n^2)$ points. Moreover, each point $x_i$ of $X$ is associated with a “range” $r^{(i)}$, such that if $X^*_p$ is an absolute $p$-center of $G$, then the absolute $p$-radius $r_p$ of $G$ is given by $r_p = \max_{x_i \in X^*_p} \{r^{(i)}\}$. Thus, a priori, there are no more than $O(|E| \cdot n^2)$ possible values for the absolute $p$-radius of $G$. In the next section, we also show that the set $X$ of all the possible points and their corresponding ranges can be found in time $O(|E| \cdot n^2)$ (see step 1 of Algorithm 3.1). Using a nondeterministic Turing machine, we can find in polynomial time for each such value $r^{(i)}$ whether there exists a subset $X_p$ of $p$ points of $X$, such that for each $v \in V$, $w(v) \cdot d(v, X_p) \leq r^{(i)}$. Hence, if $P = NP$, then an absolute $p$-center can be found in (deterministic) polynomial time which completes the proof that the problem of finding an absolute $p$-center is NP-equivalent.

3.2. An algorithm for finding a $p$-center ($p > 1$). In this section, we analyze the properties of an (absolute) $p$-center ($p > 1$) of a general network and we present an $O((|E|^p \cdot n^{2p-1}/(p-1)! \lg n)$ algorithm for finding an (absolute) $p$-center of a vertex-weighted network and an $O(E^p \cdot n^{2p-1}/(p-1)!)$ algorithm for a vertex-unweighted network. Some of the ideas expressed in this section have already appeared in the earlier works of Minieka [3] (the restriction of the search to a finite set of “suspected points”), and Handler [5] (the concept and usage of the “range” of the “suspected points”). Our approach, however, is more general and comprehensive; our arguments are based on the discussion in the previous sections (rather than on integer or constrained programming) and our algorithms are more efficient. Unfortunately, these ideas cannot improve the $O(n^{p+1}/(p-1)!)$ exhaustive algorithm for finding a vertex $p$-center, and at the end of this section we explain why.

Let $X^*_p = \{x_1, x_2, \ldots, x_p\}$ be an (absolute) $p$-center of a network $G$ and let $r_p$ be its (absolute) $p$-radius. We say that a vertex $v \in V$ is covered by the point $x_i$ of the $p$-center $X^*_p$ if $w(v) \cdot d(v, x_i) \leq r_p$. For each point $x_i \in X^*_p$, we define $V_i = \{v \in V | d(v, x_i) = d(v, X^*_p)\}$.

For each point $x_i \in X^*_p$, we define $V_i = \{v \in V | d(v, x_i) = d(v, X^*_p)\}$. The sets $V_1, V_2, \ldots, V_p$ may not be disjoint and some may be empty, but $\bigcup_{i=1}^n V_i = V$. If $V_i$ is not empty, we define $G_i$ to be the subgraph of $G$ induced by $V_i$ (i.e., $G_i$ is the graph consisting of all vertices in $V_i$ and those edges of $G$ which connect two vertices in $V_i$). Clearly, $G_i$ is connected. Let $x_i$ be a 1-center of $G_i$ and $r^{(i)}$ be its 1-radius. Then, $r^{(i)} = \max_{x_i \in V_i} \{w(v) \cdot d(v, x_i)\} \leq \max_{x_i \in V_i} \{w(v) \cdot d(v, x_i)\} \leq r_p$. If $V_i$ is empty, then we define $G_i$ to be an arbitrary single vertex $v_i$, and in this case, let $x_i = v_i$ and $r^{(i)} = 0$. Let $v$ be a vertex of $G$; then $v$ belongs to some set $V_i$ and thus: $w(v) \cdot d(v, x_i) \leq r^{(i)} \leq r_p$. Therefore, the set $X^*_p = \{x_1, x_2, \ldots, x_p\}$ is a $p$-center of $G$, and the $p$-radius of $G$ is given by:

\[ r_p = \max_{1 \leq i \leq p} \{r^{(i)}\}. \]
We call $r(i)$ the 1-radius of the subgraph $G_i$ for which $x_i$ is a 1-center the range of $x_i$.

Thus, without loss of generality, we may restrict the search for a $p$-center of $G$ to sets of the form $X_p^* = \{x_1, x_2, \ldots, x_p\}$ where $x_i(1 \leq i \leq p)$ is either a 1-center of a subgraph $G_i$ or a vertex of $G$ whose range is 0. We may further assume that the $p$ points of $X_p^*$ are distinct.

Let $x_i$ be a point of a $p$-center $X_p^*$. If the range of $x_i$ is not 0, then $x_i$ is a 1-center of a subgraph $G_i$, and thus it is also a local-center on some edge $e_i$ of $G_i$. Therefore, by condition (i) of § 2.1, $x_i$ is located at a point on $e_i$ where two functions $D_e(v_{i1}, \cdot)$ and $D_e(v_{i2}, \cdot)$ of opposite slopes intersect, and the range $r(i)$ of $x_i$ is equal to the value of these functions at their intersection point. [Notice that even if $x_i$ is located at a vertex of $G_i$, then since $x_i$ is a 1-center of $G_i$, there must be at least two edges which are incident to $x_i$, for which the above statement is correct.] It follows that on each edge $e_i \in E$ there are at most $O(n^2)$ "suspected" points at which the members of a $p$-center $X_p^*$ may be located, and thus the whole graph contains at most $O(|E| \cdot n^2)$ points which are suspected of being members of a $p$-center $X_p^*$ (where this count also includes the $n$ points of range 0 which are located at the vertices of $G$). Moreover, once one identifies a point $x_i$ as a "suspected point," one also knows the range $r(i)$ of $x_i$. If $x_i$ is a point where two functions $D_e(v_{i1}, \cdot)$ and $D_e(v_{i2}, \cdot)$ of opposite slopes intersect, then $r(i)$ is the value of those functions at their intersection. If $x_i$ is not such an intersection point, then $x_i$ is a point which is located at a vertex of $G$ and its range $r(i)$ is 0.

Let $X_p^* = \{x_1, x_2, \ldots, x_{p-1}, x_p\}$ be a $p$-center of $G$, where $x_i$'s belong to the set of the "suspected points." For each $i$ (1 $\leq i \leq p$) define: $V_i = \{v \in V : w(v) \cdot d(v, x_i) \leq r(i)\}$. Let $\tilde{V} = \bigcup_{i=1}^{p-1} V_i$, then $V - \tilde{V} \subseteq V_p$. Let $\tilde{G}$ be the network obtained from the (original) network $G$ by assigning a zero weight to all the vertices of the set $\tilde{V}$, and preserving the original weights for all the vertices of the set $V - \tilde{V}$. If $\tilde{x}_p$ is a 1-center of the network $\tilde{G}$, then it is not difficult to see that the set $\tilde{X}_p^* = \{x_1, x_2, \ldots, x_{p-1}, \tilde{x}_p\}$ is a $p$-center of $G$. Thus, without the loss of generality, we may restrict our search for a $p$-center of $G$ to a set of the form $\tilde{X}_p^*$ (namely, a set where the $p$th point is a 1-center of all the vertices which are not covered by the first $p-1$ points). The following algorithms finds a $p$-center of $G$ by constructing all possible sets of the form $\tilde{X}_p^*$.

**Algorithm 3.1.** A $p$-center of a general network.

1. Find the location $x_i$ and the corresponding range $r(i)$ of each of the $O(|E| \cdot n^2)$ "suspected points" [i.e., the vertices of $G$ and the points where two functions $D_e(v_{i1}, \cdot)$ and $D_e(v_{i2}, \cdot)$ of opposite slopes intersect].
2. For each possible set $X_{p-1} = \{x_1, x_2, \ldots, x_{p-1}\}$ of $p-1$ suspected points, do steps 3–4–5, then go to 6. [There are, at most, \(\binom{|E| \cdot n^2}{p-1}\) such sets.]
3. For each $i$, 1 $\leq i \leq p-1$, compute $V_i = \{v \in V : w(v) \cdot d(v, x_i) \leq r(i)\}$. Let $\tilde{V} = \bigcup_{i=1}^{p-1} V_i$.
4. Find a 1-center $\tilde{x}_p$ of the network $\tilde{G}$ [where $\tilde{G}$ is the network $G$ with all vertices of $\tilde{V}$ having weight 0]. Let $r(p)$ be the 1-radius of $\tilde{G}$.
5. Let $\tilde{X}_p = X_{p-1} \cup \{\tilde{x}_p\}$. Assign: $r(\tilde{X}_p) = \max_{1 \leq i \leq p} \{r(i)\}$. [Go back to 2.]
6. Let $\tilde{X}_p^*$ be a set such that $r(\tilde{X}_p^*) = \min_{\tilde{X}_p} \{r(\tilde{X}_p)\}$. Then, $\tilde{X}_p^*$ is a $p$-center of $G$ and $r(\tilde{X}_p^*)$ is the $p$-radius.

For evaluating the complexity of Algorithm 3.1 we note that if $G$ is a vertex-weighted network, then step 4 requires $O(|E| \cdot n \cdot \lg n)$ operations, while if $G$ is a vertex-unweighted network, then step 4 can be accomplished in $O(|E| \cdot n)$ operations (using slight modifications of algorithms 2.3 and 2.4). Since step 3 is carried out in $O(n \cdot p)$ operations, while step 5 requires $O(p)$ operations, the complexity of the Loop
3-4-5 is determined by the complexity of step 4. The Loop 3-4-5 is repeated (by step 2) at most
\[
\binom{|E| \cdot n^2}{p-1} = O\left(\frac{|E|^{p-1} \cdot n^{2p-2}}{(p-1)!}\right)
\]
times, and thus we conclude that the total complexity of Algorithm 3.1 is
\[
O\left(\frac{|E|^p \cdot n^{2p-1}}{(p-1)!} \log n\right)
\]
in the vertex-weighted case and
\[
O\left(\frac{|E|^p \cdot n^{2p-1}}{(p-1)!}\right)
\]
in the vertex-unweighted case (clearly steps 1 and 6 have no influence on the complexity of the algorithm).

**Remark.** Let \(X^*_p = \{x_1, x_2, \ldots, x_p\}\) be a p-center of \(G\). It can be shown that all the points of nonzero range are the local-centers of the appropriate subgraphs \(G_i\) on distinct edges of \(E\). Thus one might think of improving Algorithm 3.1 as follows: In step 2, instead of choosing a set \(X_{p-1}\) of \(p-1\) points out of the \(O(|E| \cdot n)\) "suspected" points, choose a set \(E_{p-1} = \{e_1, e_2, \ldots, e_{p-1}\}\) of \(p-1\) distinct edges out of the \(|E|\) edges of \(G\), and then on each edge \(e_i\) of \(E_{p-1}\) choose a point \(x_i\) out of the \(O(n^2)\) "suspected" points of \(e_i\). However, the complexity of such a procedure is
\[
O\left(n^{2(p-1)} \cdot \binom{|E|}{p-1}\right) = O\left(\frac{|E|^{p-1} \cdot n^{2p-2}}{(p-1)!}\right),
\]
which is the same as the complexity of step 2 in Algorithm 3.1, and thus no better result is achieved.

The above ideas, when applied to the vertex p-center problem, cannot improve the time bound which is set by the exhaustive algorithm. According to the exhaustive algorithm, for each subset \(V_p\) of \(p\) vertices of the graph, we find the "weighted distance" of the vertices of the graph from \(V_p\) (where this distance is given by \(\max_{v \in V} \min_{v' \in V_p} \{w(v) \cdot d(v, v')\}\)) and clearly, the minimum over these distances is the vertex \(p\)-radius of the graph, while a corresponding subset \(V_p\) is a vertex \(p\)-center of the graph. Since each such distance can be computed in time \(O(n \cdot p)\), the whole exhaustive algorithm requires
\[
\binom{n}{p} \cdot O(n \cdot p) = O\left(\frac{n^{p+1}}{(p-1)!}\right)
\]
time. This time bound can also be expressed as
\[
\binom{n}{p-1} \cdot O(n \cdot (n-p)),
\]
which shows that it is inconceivable that a better algorithm can be obtained by a strategy of searching through all the \(\binom{n}{p-1}\) subsets \(V_{p-1}\) of \(p-1\) vertices, and finding the corresponding \(p\)th vertex according to the selection of the first \(p-1\), in time less than \(O(n \cdot (n-p))\).
4. Centers of a vertex-weighted tree. In the following sections, we assume that the underlying structure of the network is a tree, and we denote it by \( T(V, E) \), or simply \( T \). We describe an \( O(n \cdot \log n) \) algorithm for finding the absolute or vertex 1-center of a vertex-weighted tree,\(^6\) compared with the algorithm of Hakimi, Schmeichel, and Pierce [9] which solves the same problem in \( O(n^2) \) steps. (It is possible, however, that practically their algorithm is as good as ours.) We proceed by describing an \( O(n) \) algorithm for finding an absolute (or a vertex) dominating set of radius \( r \) on a vertex-weighted tree. No such algorithm is known, although some special cases were treated by other authors [15], [19]. This algorithm leads to an \( O(n^2 \cdot \log n) \) algorithm for finding a (vertex or absolute) \( p \)-center \((p > 1)\) on a vertex-weighted tree. No algorithm specifically for this problem has been published so far. We conclude by discussing some generalizations of the problem.

4.1. The centroid method for finding 1-center of a tree. The following discussion applies to both the absolute 1-center problem and the vertex 1-center problem. Let \( T \) be a vertex-weighted tree and \( v \) be a vertex of the tree whose degree is \( d_v \). Let \( T - v \) be the graph which results by removing \( v \) (and its \( d_v \) incident edges) from \( T \). Then \( T - v \) consists of \( d_v \) connected subtrees which will be denoted by \( T_{v,1}, T_{v,2}, \ldots, T_{v,d_v} \). We shall denote by \( T_{v,l}^+ \) the subtree which consists of \( T_{v,l} \), the vertex \( v \), and the edge which connects \( v \) to \( T_{v,l} \).

**Lemma 4.1.** Let \( v \in V \) be a fixed vertex and let \( \hat{v} \) be a vertex such that \( w(\hat{v}) \cdot d(\hat{v}, v) = \max_{v' \in V} \{w(v') \cdot d(v', v)\} \). Let \( T_{v,l}^\prime \) be the subtree of \( T - v \) to which \( \hat{v} \) belongs. Then, the 1-center of \( T \) is on \( T_{v,l}^\prime \).

**Proof.** Assume that the 1-center \( x^* \) of \( T \) is not on \( T_{v,l}^\prime \). Then \( d(\hat{v}, x^*) > d(\hat{v}, v) \) and thus if \( r_1 \) is the 1-radius of \( T \), then \( r_1 \equiv w(\hat{v}) \cdot d(\hat{v}, x^*) > w(\hat{v}) \cdot d(\hat{v}, v) = \max_{v' \in V} \{w(v') \cdot d(v', v)\} \). Therefore, the choice of \( v \) as the 1-center of \( T \) is better than the choice of \( x^* \), which is a contradiction.

**Corollary 4.1.** Let \( \hat{v} \) and \( \hat{v}^\prime \) be two vertices such that \( \hat{v} \in T_{v,l}, \hat{v} \in T_{v,k}, k \neq l, \) and \( w(\hat{v}) \cdot d(\hat{v}, v) = w(\hat{v}^\prime) \cdot d(\hat{v}^\prime, v) = \max_{v' \in V} \{w(v') \cdot d(v', v)\} \). Then \( v \) is the 1-center of \( T \).

We note that if the vertex \( v \) in Lemma 4.1 is not a leaf of \( T \) (namely, if \( d_v > 1 \)), then \( T_{v,l}^\prime \) is a proper subtree of \( T \). Thus, Lemma 4.1 suggests a straightforward algorithm for finding the 1-centered of a weighted tree: Let \( T_0 \) be the original tree \( T \). Choose some vertex \( v_0 \) of \( T_0 \) (where \( v_0 \) is not a leaf of \( T_0 \)) and find the subtree \( T_{v_0,l_0}^+ \) on which \( T_{v_0,l_0}^+ \) is a proper subtree of \( T \). Thus, Lemma 4.1 suggests a straight-forward algorithm for finding the 1-centered of a weighted tree: Let \( T_0 \) be the original tree \( T \). Choose some vertex \( v_0 \) of \( T_0 \) (where \( v_0 \) is not a leaf of \( T_0 \)) and find the subtree \( T_{v_0,l_0}^+ \) on which \( T_{v_0,l_0}^+ \) is a proper subtree of \( T \). Thus, the 1-centered of \( T_{v_0,l_0}^+ \) is on the edge which connects \( v_0 \) to \( T_{v_0,l_0}^+ \).

Except for the last step of finding the local-center (or finding which end-point is the vertex 1-center), each stage (i.e., finding the subtrees \( T_{v_i,l_i}^+ \) and \( T_{v_i,l_i+1}^+ \)) requires \( O(n) \) operations. Thus, the total complexity of this algorithm is \( O(n \cdot k + n \cdot \log n) \) for the absolute 1-center of \( O(n \cdot k) \) for the vertex 1-center (where \( k \) is the number of stages, and \( n \cdot \log n \) is the cost of the last step in the case of the absolute 1-center; in the case of the vertex 1-center, the last step requires \( O(n) \) time). The value of \( k \) depends on the way we choose the vertex \( v_i \) at each stage \( i \) of the algorithm. In order to have a good choice of \( v_i \) we introduce here the notion of a centroid of a tree.

\(^6\) Notice that a tree always has a unique absolute 1-center, and at most 2 vertex 1-centers.
Centroid of a tree. Let \( v \) be a vertex of \( T \), and let \( T_{v,1}, T_{v,2}, \ldots, T_{v,d_v} \) be the connected subtrees of \( T - v \). Let \( |T| \) denote the number of vertices in \( T \) and define \( N(v) \) by:

\[
N(v) = \max_{1 \leq i \leq d_v} \{|T_{v,i}|\}.
\]

A centroid of the tree \([13]\) is a vertex \( v_c \) for which \( N(v) \) is minimum:

\[
N(v_c) = \min_{v \in V} \{N(v)\}.
\]

We note that a tree might have either one centroid or two. In the latter case, the two centroids are connected by an edge \([13]\).

One can easily see that \( N(v_c) \leq \lfloor n/2 \rfloor \). Namely, the number of vertices in each of the subtrees \( T_{v_c,1}, T_{v_c,2}, \ldots, T_{v_c,d_{vc}} \) is not greater than \( \lfloor n/2 \rfloor + 1 \). Thus, if in the above algorithm, at each stage we choose a centroid of \( T_i \) to be the vertex \( v_i \), then \( |T_{i+1}| \leq \lfloor |T_i|/2 \rfloor + 1 \), and the number \( k \) of stages will be at most \( O(\log n) \). Therefore, in order to accomplish the construction of an \( O(n \cdot \log n) \) algorithm for finding the 1-center of a tree, we have to supply an \( O(n) \) algorithm for finding a centroid of a tree.

However, it is not difficult to see that the inequality \( N(v_c) \leq \lfloor n/2 \rfloor \) is not only a necessary condition for a vertex \( v_c \) to be a centroid of the tree, but it is also a sufficient condition. Thus, based on this property, we can use a version of Goldman's algorithm \([14]\) to find a centroid of a tree in \( O(n) \) steps. Throughout the algorithm we use a copy \( T' \) of the original tree \( T \) as an auxiliary tree on which the algorithm works. We also attach a variable \( n(v) \) to every vertex \( v \) of the tree. By the treatment of \( T' \) throughout the algorithm, it is easy to see that if \( v \) is a leaf of \( T' \), then \( T' - v \) is contained in one of the connected subtrees of \( T - v \) and \( n(v) \) then gives the number of vertices of this subtree.

Algorithm 4.1. A centroid of a tree.

1. [Initialization.] Assign: \( T' \leftarrow T \). For each vertex \( v \in T' \) assign: \( n(v) \leftarrow n - 1 \).
2. If the auxiliary tree \( T' \) consists of a single vertex \( v_0 \), then Halt [the vertex \( v_0 \) is a centroid of the original tree \( T \)].
3. Let \( v \) be a leaf of the auxiliary tree \( T' \). If \( n(v) \leq \lfloor n/2 \rfloor \) then Halt \([v \ is \ a \ centroid \ of \ the \ original \ tree \ T]\). Else, let \( u \) be the vertex adjacent to \( v \) in \( T' \). Assign: \( n(u) \leftarrow n(u) - (n - n(v)) \). Remove vertex \( v \) (and edge \( (u, v) \)) from \( T'[T' \leftarrow T' - \{v\}] \) and go to 2.

The detailed proof of the validity of Algorithm 4.1, and that it takes \( O(n) \) steps to find a centroid of a tree \( T \) is left to the reader. (Notice that in step 3, if \( n(v) \) is the number of vertices of the subtree of \( T - v \) to which \( u \) belongs, then \( n - n(v) \) is the number of vertices of the subtree of \( T - u \) which contains \( v \). Therefore, \( n(u) \) is correctly updated, i.e., when \( u \) becomes a leaf, \( n(u) \) gives the number of vertices of the subtree of \( T - u \) which contains \( T' - u \). Also, if the relation \( n(v) \leq \lfloor n/2 \rfloor \) does not hold, then \( v \) cannot be a centroid and its removal from \( T' \) still leaves the centroid(s) of \( T \) in \( T' \).)

Using Algorithm 4.1, we can now formulate Algorithm 4.2 for finding the 1-center of a tree. In the following algorithm, the variables \( T', T'', \) and \( T''' \) represent at each stage the subtrees \( T, T_{v,1}, \) and \( T_{v,1} \), respectively.

Algorithm 4.2. The 1-center of a tree (the centroid method).

1. Assign: \( T' \leftarrow T \).

Goldman's algorithm was originally written for the purpose of finding a 1-median of a tree (see also part II of this paper: "The \( p \)-Medians").
NETWORK LOCATION PROBLEMS. I

2. [For absolute 1-center:] If $T'$ has a single edge $e$, then by Algorithm 2.1, find the local-center $x^*$ of $T$ on $e$ and Halt [$x^*$ is the 1-center of $T$]. [For vertex 1-center:] If $T'$ has a single edge $e = (v_n, v_s)$, then let $d_e = \max_{v' \in V} \{w(v') \cdot d(v', v_s)\}$ and let $d_s = \max_{v' \in V} \{w(v') \cdot d(v', v_s)\}$. If $d_e < d_s$, then $v_n$ is the vertex 1-center of $T$; if $d_e > d_s$, then $v_s$ is the vertex 1-center of $T$; if $d_e = d_s$, then each of $v_n$ and $v_s$ is a vertex 1-center of $T$. Halt.

3. Using Algorithm 4.1, find a centroid $v_c$ of $T'$.

4. Let $\tilde{v}$ be a vertex of $T$ such that $w(\tilde{v}) \cdot d(\tilde{v}, v_c) = \max_{v' \in V} \{w(v') \cdot d(v', v_c)\}$. Let $T''$ be the component of $T - v_c$ which contains $\tilde{v}$. Let $T''$ be the subtree which consists of $T''$, the vertex $v_c$, and the edge which connects $v_c$ to $T''$.

5. [An optional step.] If there exists a vertex $v$ such that $w(v) \cdot d(v, v_c) < w(v_n) \cdot d(v_n, v_c)$, then [by Corollary 4.1], $v_c$ is the 1-center of $T$. Halt.

6. Assign: $T' \leftarrow T' \cap T''$ and return to 2.

By the above considerations, Algorithm 4.2 finds the 1-center of a tree in $O(n \cdot \lg n)$ steps. We notice that since the network is a tree, the computation of the distance-matrix is not required for finding the local-center on $e$, or the weighted distances $d_e$ and $d_s$, in step 2 of Algorithm 4.2.

4.2. A dominating set of radius $r$ in a vertex-weighted tree. In this section we describe an $O(n)$ algorithm to find an (absolute or a vertex) dominating set of radius $r$ in a vertex-weighted tree. Namely: Given a positive number $r$ and a vertex-weighted tree $T(V, E) = T$, find the smallest positive integer $p$ such that the (absolute or vertex) $p$-radius of $T$ is not greater than $r$. We intend to show that this task can be carried out by an $O(n)$ algorithm, and that this algorithm also finds a corresponding $p$-center.

The problem of finding a (simple) dominating set in a vertex-unweighted tree (i.e., finding a dominating vertex set of radius 1 in a tree all whose vertices have weight 1) was solved in $O(n)$ time by Cockayne, Goodman, and Hedetniemi [19]. The restricted problem of finding a dominating vertex set of an integer radius $r$ in a vertex-unweighted tree all whose vertices are of length 1 was solved in linear time by Slater [15]. Although Slater's algorithm is based on ideas similar to ours, namely searching the tree from the leaves toward its middle, no direct generalization of his algorithm to the general problem seems applicable. On the other hand, algorithm 4.3, which is presented in this section, can be simply modified to handle a generalization treated by Slater in which every vertex $v$ is covered by the dominating set within a different radius $r(v)$. In fact, all that should be modified in Algorithm 4.3 in order to handle this generalization is to replace each term $r/(w(v))$ in the algorithm by the term $(r(v))/(w(v))$. Other generalizations are discussed in §4.4.

The following algorithm finds both an absolute and a vertex dominating set of radius $r$. The algorithm is carried out through a search on the edges of $T$, starting from the leaves and moving toward the “middle.” During this search, we locate the points of the desired dominating set of radius $r$ in an “optimal fashion” until the whole tree is covered by some $p$ points within a (weighted) radius of $r$. To do this, we use a copy $T'$ of the original tree as an auxiliary tree on which the algorithm is carried out, and we attach a variable $R(v)$ to each vertex $v$ of $T'$. (The interpretation of $R(v)$ will be given later.) If $v_i$ is a leaf of the auxiliary tree $T'$ and $e = (v_n, v_s)$ is the edge incident to it, then we traverse (search) the edge $e$ from $v_i$ to $v_s$, update the variable $R(v_i)$, and remove the vertex $v_i$ and the edge $e$ from the tree $T'$. As a result, a new vertex may become a leaf of $T'$, and the process is repeated until all the vertices and edges of $T'$ are deleted.

The variable $R(v)$ has the following interpretation:
Case (a). If the vertex \( v \) is already covered by one of the points (of the dominating set) which have been located on the tree so far, then \( R(v) \) is the unweighted distance between \( v \) and the nearest located point. Clearly, in this case \( r/(w(v)) \geq R(v) \geq 0 \).

Case (b). If the vertex \( v \) is not yet covered, then let \( S(v) \) be the set of all the vertices of the original tree \( T \) which are not yet covered, and for which \( v \) is the nearest vertex on the auxiliary tree \( T' \). Notice that \( v \) is the only vertex in \( S(v) \) which belongs to \( T' \); in fact, \( S(v) \) is the set consisting of the vertex \( v \) and all those vertices which have already been removed from \( T' \) and are to be covered by the same point of the dominating set as \( v \). We define: \( -R(v) = \min_{v' \in S(v)} \{ r/(w(v')) - d(v', v) \} \). Thus, \( -R(v) \) is the maximum (unweighted) distance from \( v \) within which any point of the auxiliary tree \( T' \) would cover all vertices of the set \( S(v) \). In particular, in this case: \( r/(w(v)) \geq -R(v) > 0 \) (the fact that in Case (b) \( R(v) \neq 0 \) is a consequence of the following algorithm rather than that of the above definition of \( R(v) \)).

**Algorithm 4.3.** A dominating set of radius \( r \) in a vertex-weighted tree.

1. **[Initialization.]** Assign \( T' \leftarrow T, p \leftarrow 0 \). For each vertex \( v \) of \( T' \) assign: \( R(v) \leftarrow -r/(w(v)) \).

2. If there exists no leaf \( v_r \) of the auxiliary tree \( T' \) such that \( R(v_r) \geq 0 \) [no leaf is covered], then go to 5. If the auxiliary tree \( T' \) is one vertex \( v_r \) with \( R(v_r) \geq 0 \) then Halt [the original tree is covered within radius \( r \) by a dominating set of \( p \) points]. Else, let \( v_r \) be a leaf of \( T' \) such that \( R(v_r) >-0 \) and let \( e = (v_n, v_r) \) be the edge incident in \( T' \) to \( v_r \). Remove \( e \) and \( v_r \) from the auxiliary tree \( T' \).

3. **[The point of the dominating set which covers \( v_r \) is located at a distance \( R(v_r) + l(e) \) from \( v_r \).]** If \( R(v_r) + l(e) \leq r/(w(v_r)) \) then go to 4. If \( R(v_r) + l(e) > r/(w(v_r)) \) [the point which covers \( v_r \) does not cover \( v_r \)], then return to 2.

4. **[\( R(v_r) + l(e) \leq r/(w(v_r)) \): the point of the dominating set which covers \( v_r \) also covers \( v_n \).]** If \( 0 \leq R(v_r) \leq R(v_r) + l(e) \) [\( v_r \) is already covered by a closer point], then return to 2. If \( 0 < R(v_r) + l(e) \leq R(v_r) \) [the point of the dominating set which covers \( v_r \) is closer to \( v_r \) than the point of the dominating set which presently covers \( v_r \)], then assign: \( R(v_r) \leftarrow R(v_r) + l(e) \) and return to 2. If \( 0 < -R(v_r) < R(v_r) + l(e) \) [the point of the dominating set which covers \( v_r \) does not cover all points of \( S(v_r) \)], then return to 2.

5. **[For each leaf \( v_r \) of \( T' \), \( R(v_r) < 0 \): No leaf is already covered.]** If the auxiliary tree \( T' \) is one vertex, then assign \( p \leftarrow p + 1 \), locate a new point of the dominating set at the vertex \( v_r \), and Halt. Else, let \( v_r \) be a leaf of \( T' \) and let \( e = (v_n, v_r) \) be the edge incident in \( T' \) to \( v_r \). Remove \( e \) and \( v_r \) from the auxiliary tree \( T' \).

6. If \(-R(v_r) > l(e) \), then go to 7. If \(-R(v_r) = l(e) \), then go to 8. If \(-R(v_r) < l(e) \), then go to 9.

7. **[-R(v_r) > l(e): It is not necessary for the point of the dominating set which would cover the set \( S(v_r) \) to be located on the edge \( e \).]** If \(-R(v_r) - l(e) < -R(v_r) \), then \( [R(v_r) < 0 \) and thus \( v_r \) is not yet covered; following the removal of \( v_r \) from \( T' \), the vertices of \( S(v_r) \) are now included in the set \( S(v_n) \), and therefore the value of \( R(v_r) \) must be updated] assign: \( R(v_r) \leftarrow R(v_r) + l(e) \) and return to 2. If \( 0 < -R(v_r) \leq -R(v_r) - l(r) \), then [although the vertices of \( S(v_r) \) are now...
NETWORK LOCATION PROBLEMS

included in the set $S(v_s)$, no change of the value of $R(v_s)$ is required] return to 2. If $0 \leq R(v_s) \leq -R(v_r) - l(e)$, then $v_s$ is already covered, and the point of the dominating set which covers $v_s$ also covers the vertices of $S(v_s)$] return to 2. If $-R(v_s) - l(e) < R(v_r)$, then [the vertex $v_s$ is already covered; however, the point of the dominating set which is now covering $v_s$ cannot cover the vertices of the set $S(v_s)$]. We shall treat $v_s$ as if it is not yet covered, and will look for a point which covers both the set $S(v_s)$ and the vertex $v_r$. Thus, the new set $S(v_s)$ is now $S(v_s) \cup \{v_s\}$, and the value of $R(v_s)$ must be reassigned] assign: $R(v_s) = R(v_r) + l(e)$, and return to 2.

8. $[-R(v_r) - l(e)]$ In order to cover the set $S(v_r)$, a new point of the dominating set must be located at $v_r$ itself.] If $R(v_r) = 0$ [there is already a point of the dominating set located at $v_r$], then return to 2. Else, assign: $p \leftarrow p + 1$, locate a new point of the dominating set at $v_r$, assign: $R(v_r) = 0$ and return to 2.

9. [For absolute dominating set of radius $r$] $[-R(v_r) - l(e)]$ In order to cover the set $S(v_r)$, a new point of the dominating set must be located on edge $e$, at a distance $-R(v_r)$ from $v_r$, namely, at a distance $l(e) + R(v_r)$ from $v_r$. Assign: $p \leftarrow p + 1$, locate a new point of the dominating set on the edge $e$ at a distance $-R(v_r)$ from $v_r$ [R$(v_r)$ + l(e) from $v_r$] and go to 3. [For vertex dominating set of radius $r$] $[-R(v_r) - l(e)]$: The vertex $v_r$ does not cover the set $S(v_r)$, and therefore a new point of the dominating set should be located at $v_r$. Assign: $p \leftarrow p + 1$, locate a new point of the $p$-center at $v_r$, assign: $R(v_r) = 0$ and go to 3.

The number $p$ which is obtained when the algorithm terminates is the (absolute or vertex) domination number of radius $r$, and the set of $p$ points which was constructed during the algorithm is a dominating set of radius $r$. The validity of Algorithm 4.3 can be established by observing that, at each stage, the present number $p$ is the minimum number of points which are needed to cover those vertices of the tree which are presently covered. (That is, a point is added to the dominating set whenever it is absolutely necessary.) The complexity of this algorithm is easily seen to be $O(n)$.

4.3. A $p$-center ($p > 1$) of a vertex-weighted tree. In this section, we use Algorithm 4.3, which was designed for finding the (absolute or vertex) domination number of radius $r$ and a corresponding dominating set, in order to find the (absolute or vertex) $p$-radius of a vertex-weighted tree $T$ and a corresponding $p$-center. For this, we first show that a priori there exist $O(n^2)$ possible values of the (absolute or vertex) $p$-radius.

For the absolute $p$-radius: Let $X_p^*$ be an absolute $p$-center of a vertex-weighted tree $T$, and let $r_p$ be its absolute $p$-radius. Then (by the discussion of § 3.2), there exists at least one point $x \in X_p^*$ and a pair of vertices $v_1$ and $v_2$, such that the paths from $x$ to $v_1$ and to $v_2$ do not intersect (that is, $d(v_1, v_2) = d(v_1, x) + d(x, v_2)$) and $w(v_1) \cdot d(v_1, x) = w(v_2) \cdot d(v_2, x) = r_p$. A simple manipulation of these relations yields [9]: $r_p = ((w(v_1) \cdot w(v_2))/(w(v_1) + w(v_2))) d(v_1, v_2)$. Thus, a priori there are at least $2$ possible values for the absolute $p$-radius, namely the $(n \cdot (n - 1))/2$ possible values for the absolute $p$-radius, namely the $(n \cdot (n - 1))/2$ values given by

$$r_{ij} = \frac{w(v_i) \cdot w(v_j)}{w(v_i) + w(v_j)} d(v_i, v_j)$$

for $1 \leq i < j \leq n$.

Since the distance-matrix of a tree can be computed in $O(n^2)$ steps, so can the matrix $[r_{ij}] (1 \leq i < j \leq n)$. This matrix gives us all the possible values that $r_p$ can assume for any value of $p (1 \leq p < n)$.

For vertex $p$-radius: Let $V_p^*$ be a vertex $p$-center of a vertex-weighted tree $T$, and let $r_p^{(v)}$ be its vertex $p$-radius. Then, by the definition (1.2) and (1.3), $r_p^{(v)} = \max_{v \in V} \{w(v) \cdot d(v, V_p^*)\}$. Thus, there exist (at least) two vertices $v_1$ and $v_2$ such that
\[ r_p^{(v)} = w(v_i) \cdot d(v_i, v_j). \] Therefore, a priori there are at most \( n \cdot (n - 1) \) possible values for the vertex \( p \)-radius, namely the \( n \cdot (n - 1) \) values given by

\[
(4.4) \quad r_i = w(v_i) \cdot d(v_i, v_j) \quad \text{for } i \neq j, \ 1 \leq i, j \leq n.
\]

(Although we denote the values of (4.3) and (4.4) by the same notation \( r_i \), there should be no confusion, as (4.3) is used for absolute \( p \)-radius, while (4.4) is used for the vertex \( p \)-radius.) The matrix \( \|r_i\| \) can be computed in \( O(n^2) \) steps, and it gives us all the possible values that \( r_p^{(v)} \) can assume for any value of \( p \) (\( 1 \leq p < n \)).

We return now to the problem of finding the (absolute or vertex) \( p \)-radius of a tree. Denote the desired (absolute or vertex) \( p \)-radius by \( r \) and let \( r_i \) be one of the possible values of \( r_p \) as given by (4.3) (for absolute \( p \)-radius) or (4.4) (for vertex \( p \)-radius). Let \( p_i \) be the (absolute or vertex) domination number of radius \( r_i \) (\( p_i \) can be found by Algorithm 4.3). If \( p_i \leq p \), then \( r_p \leq r_i \). Therefore we obtain:

\[
(4.5) \quad r_p = \min_{i,j} \{ r_i | p_i \leq p \}.
\]

Thus, by using Algorithm 4.3, one can search the \( O(n^2) \) possible values \( r_i \) and find the \( p \)-radius of the tree. In order to improve the efficiency of this procedure, we can perform a binary search rather than an exhaustive search.

Algorithm 4.3, which gives the domination number \( p_i \), also constructs a dominating set of radius \( p_i \). Thus, once the \( p \)-radius \( r_p \) is known, one can construct a dominating set of radius \( r_p \). Let \( p' \) be the number of points in this set. If we add to the dominating set \( p - p' \) points (arbitrary \( p - p' \) points in the case of absolute \( p \)-center, or \( p - p' \) arbitrary vertices in the case of vertex \( p \)-center), then we obtain a desired (absolute or vertex) \( p \)-center.

We can now formulate an algorithm for finding the (absolute or vertex) \( p \)-radius of a tree and a corresponding \( p \)-center.

**Algorithm 4.4.** A \( p \)-center (\( p > 1 \)) of a vertex-weighted tree.

1. Calculate the \( O(n^2) \) values \( r_i \) [for absolute \( p \)-center, \( r_i = ((w(v_i) \cdot w(v_j))/\sum_{i,j} w(v_i)) \cdot d(v_i, v_j) \) for \( 1 \leq i < j \leq n \); for vertex \( p \)-center, \( r_i = w(v_i) \cdot d(v_i, v_j) \) for \( i \neq j, \ 1 \leq i, j \leq n \)].
2. Arrange the \( O(n^2) \) values \( r_i \) in a list \( L \) according to a nondecreasing order.
3. By performing a binary search on the list \( L \) of the values \( r_i \), and by using Algorithm 4.3, find the smallest \( r_i \) for which \( p_i \leq p \) (where \( p_i \) is the domination number of radius \( r_i \)). Denote this value of \( r_i \) by \( r_p \) [this is the desired \( p \)-radius], and denote the domination number of radius \( r_p \) by \( p' \).
4. Let \( X_p^* \) be the dominating set of radius \( r_p \) as constructed by Algorithm 4.3. Add any arbitrary \( p - p' \) points to \( X_p^* \) (in the case of the vertex \( p \)-center, these points must be vertices). The resulting set \( X_p^* \) is a \( p \)-center of the tree.

Step 1 of Algorithm 4.4 is carried out in \( O(n^2) \) operations. Step 2 requires \( O(n^2 \cdot \lg n) \) operations. The complexity of the binary search which is performed in step 3 is \( O(\lg n) \) operations, while Algorithm 4.3, which is carried out in each search, costs \( O(n) \) operations—and thus the complexity of step 3 is \( O(n \cdot \lg n) \). Therefore, the complexity of Algorithm 4.4 is determined by step 2 and it is \( O(n^2 \cdot \lg n) \).

**4.4. Some remarks and generalizations.** (a) We finally note that as the \( O(n^2) \) possible values \( r_i \) of the \( p \)-radius are the same for any possible \( p \) (\( 1 \leq p < n \)), one can find (absolute or vertex) \( p \)-centers of the tree for all values of \( p \) by the following procedure: Find all the \( O(n^2) \) values of \( r_i \) and all the \( O(n^2) \) corresponding domination numbers \( p_i \).
Now for each \( 1 \leq p < n \) find the smallest \( r_{ij} \) for which \( p_{ij} \leq p \) and add arbitrary \( p - p_{ij} \) points (or vertices) to the corresponding dominating set of radius \( r_{ij} \). For each \( p \), the resulting set would be a \( p \)-center while the corresponding \( r_{ij} \) would be the \( p \)-radius. Clearly, the complexity of this procedure is \( O(n^3) \).

(b) Algorithms 4.3 and 4.4 can be modified to handle the following generalization of the \( p \)-center problem that we refer to as the “constrained absolute (vertex) \( p \)-center problem”: Given a set \( Y^*_p \) of \( p \) points (vertices) on the tree, find a set \( Y^*_{p_2} \) of \( p_2 \) points (vertices) such that \( p_2 = p - p_1 \) and

\[
\max_{v \in V} \left\{ w(v) \cdot d(v, Y^*_{p_1} \cup Y^*_{p_2}) \right\} = \min \left[ \max_{v \in V} \left\{ w(v) \cdot d(v, Y^*_{p_1} \cup Y^*_{p_2}) \right\} \right]
\]

[where the minimum is taken over all sets \( Y^*_{p_2} \) of \( p_2 \) points (vertices) on the tree]. In other words, the set \( X^* = Y^*_{p_1} \cup Y^*_{p_2} \) is a “constrained \( p \)-center,” where \( p_1 \) of the points are fixed.

Similarly, the “constrained absolute (vertex) domination number of radius \( r \)” is defined as the smallest integer \( p \) \( (p \geq p_1) \) such that a constrained absolute (vertex) \( p \)-center, of which \( p_1 \) points (vertices) are fixed, has radius \( \leq r \).

Some special cases of a constrained vertex dominating set were treated by Cockayne, Goodman, and Hedetniemi [19], and by Slater [15]. The constrained \( p \)-center problem was treated by Matula and Kolde [16].

In order to solve the “constrained vertex dominating set of radius \( r \)” problem, one should modify step 1 of Algorithm 4.3 as follows:

1. [Initialization] Let \( Y^*_{p_1} \) be the given set of \( p_1 \) vertices which should belong to the constrained dominating set. Assign: \( T' \leftarrow T, p \leftarrow p_1 \). For each vertex \( v \) of \( Y^*_{p_1} \), assign: \( R(v) \leftarrow 0 \). For each vertex \( v \) of \( T' - Y^*_{p_1} \), assign: \( R(v) \leftarrow -r/w(v) \).

In order to solve the “constrained absolute dominating set of radius \( r \)” problem, one should modify step 1 of Algorithm 4.3 as follows:

1. [Initialization] Let \( Y^*_{p_1} \) be the given set of \( p_1 \) points which should belong to the dominating set. Assign: \( T' \leftarrow T, p \leftarrow p_1 \). For each vertex \( v \) of \( T' \), assign: \( R(v) \leftarrow -r/w(v) \).

1.5. For each point \( y \in Y^*_{p_1} \), do: If \( y \) is a vertex of \( T' \), then assign: \( R(y) \leftarrow 0 \); else, let \( e = (v_1, v_2) \) be the edge on which \( y \) is located. If \( (i = 1, 2) d(v_i, y) \leq |R(v_i)| \), then assign: \( R(v_i) \leftarrow d(v_i, y) \).

[In fact, if we want to follow the interpretation of the steps of Algorithm 4.3, it would be desirable to replace the last sentence of step 2. “Remove \( e \) and \( v_i \) from the auxiliary tree \( T' \),” by the following sentences: “If a point of \( Y^*_{p_1} \) is located on \( e \) in a distance \( R(v_i) \) from \( v_i \), then assign: \( R(v_i) \leftarrow -R(v_i) \) [the point which covers \( v_i \) is located at a distance \( R(v_i) + l(e) \) from \( v_i \]. Remove \( e \) and \( v_i \) from the auxiliary tree \( T' \).” However, the reader can observe that this modification is insignificant to the course of the algorithm.]

In order to solve the “constrained \( p \)-center” problem, we first make the following observation: For each vertex \( v \in V \), let \( y(v) \) be the point of \( Y^*_{p_1} \), which is closest to \( v \) (ties are broken arbitrarily). Thus, \( d(v, y(v)) = \min_{y \in Y^*_{p_1}} \{d(v, y)\} \). Notice that each of the weighted distances \( w(v) \cdot d(v, y(v)) \) may be the “constrained \( p \)-radius” of the tree \( T \). Of course, in the case of the constrained vertex \( p \)-center problem, these values are already included in the set of the \( n \cdot (n - 1) \) values \( r_{ij} \) as defined by (4.4) and therefore no change is required. However, in the case of the constrained absolute \( p \)-center problem, the \( n \) possible values \( w(v) \cdot d(v, y(v)) \) should be added to the previous \((n \cdot (n - 1))/2 \) possible values for the \( p \)-radius which are given by (4.3). Thus, step 1 of Algorithm 4.4 should be
changed accordingly in this case. Yet, the order of magnitude of the number of possible values for the absolute \( p \)-radius remains the same—namely, \( O(n^2) \).

We can now use Algorithm 4.4 where in steps 3 and 4 we of course use the modified Algorithm 4.3 to find the constrained domination number of radius \( r_{ij} \) (rather than the usual domination number of radius \( r_{ij} \)) and a corresponding constrained dominating set. Clearly, the complexity of the modified algorithm is the same as in the "nonconstrained" case, namely, \( O(n^2 \cdot \lg n) \).

5. Centers of a vertex-unweighted tree. It has been shown by Handler \[5\], \[7\] and others \[9\] that the 1-center and a 2-center of a vertex-unweighted tree can be found in \( O(n) \) steps. Here we concern ourselves with \( p \)-centers of vertex-unweighted trees where \( p > 2 \). Our result is based on the following theorem of Hakimi, Schmeichel, and Pierce \[9\]:

"Let point \( x^* \) be the (absolute) 1-center of \( T \), \( v_1 \) be a peripheral vertex, and \( p(v_1, x^*) \) be the path which connects \( v_1 \) and \( x^* \), and if \( x^* \) is not a vertex, \( p(v_1, x^*) \) is assumed to include the edge upon which \( x^* \) lies. There exists an edge \( e \) on \( p(v_1, x^*) \) which when removed from \( T \) leaves two components, \( T_1 \) (containing \( v_1 \)) and \( T_2 \), such that if \( X^*_1 \) is the (absolute) 1-center of \( T_1 \) and \( X_{p-1}^* \) is an absolute \( (p-1) \)-center of \( T_2 \), then \( X^*_p = \{x^*\} \cup X_{p-1}^* \) is an (absolute) \( p \)-center of \( T \)."

Based on this property, Hakimi, Schmeichel, and Pierce suggested the following \( O(n^{p-1}) \) algorithm for finding an (absolute) \( p \)-center (\( p > 2 \)):

1. Find the absolute 1-center \( x^* \) of \( T \) and a peripheral vertex \( v_1 \) [this step requires \( O(n) \) operations].
2. Let \( p(v_1, x^*) \) be the path which connects \( v_1 \) and \( x^* \), and if \( x^* \) is not a vertex, \( p(v_1, x^*) \) is assumed to include the edge upon which \( x^* \) lies. For each edge \( e \) of \( p(v_1, x^*) \), let \( T_1(e) \) be the component of \( T-e \) which contains \( v_1 \) and \( T_2(e) \) the other component of \( T-e \). Find the (absolute) 1-radius \( r_1^*(e) \) of \( T_1(e) \) [this requires \( O(n) \) operations]. Find an (absolute) \( (p-1) \)-center \( X_{p-1}^*(e) \) and the (absolute) \( (p-1) \)-radius \( r_{p-1}^*(e) \) of \( T_2(e) \). [By recursion, this requires \( O(n^{p-2}) \) operations.] Assign: \( r_p(e) = \max \{r_1^*(e), r_{p-1}^*(e)\} \). [The total complexity of performing step 2 for all the edges on \( p(v_1, x^*) \) is thus \( O(n^{p-1}) \).]
3. Let \( r_p(e_0) = \min_{e \in p(v_1, x^*)} \{r_p(e)\} \). Then \( r_p(e_0) \) is the (absolute) \( p \)-radius of \( T \) and the set \( \{ x^*_0 \cup X_{p-1}^*(e_0) \} \) is an (absolute) \( p \)-center.

However, it is clear that in step 2 of the above algorithm, one can choose the edges of \( p(v_1, x^*) \) by a binary search (rather than by an exhaustive search). In this case, the complexity of the algorithm will be \( O(n \cdot \lg^{p-2} n) \) for \( p > 2 \) (namely, less than \( O(n^2) \)).

The theorem of Hakimi, Schmeichel, and Pierce is not valid in the case of a vertex \( p \)-center of a tree. However, the following lemma holds in the case of a vertex \( p \)-center of a vertex-unweighted tree:

Lemma 5.1. Let \( T(V, E) \) be a vertex-unweighted tree, and let \( v_1 \) and \( v_2 \) be two peripheral vertices such that \( d(v_1, v_2) = \max_{v \in V} \{d(v, v')\} \). For each edge \( e \) on the path \( p(v_1, v_2) \) which connects \( v_1 \) and \( v_2 \), let \( T_1(e) \) and \( T_2(e) \) be the two connected subtrees which are obtained by removing \( e \) from \( T \) (where \( T_1(e) \) contains \( v_1 \)). Then, for each \( p \) (\( p > 1 \)), there exists an edge \( e_p \) on \( p(v_1, v_2) \) and two positive integers \( p_1 \) and \( p_2 \) where

\[ r_p(e) = \max \{r_1(e), r_{p-1}(e)\}. \]

In fact, due to the points of \( Y_{p-1}^* \), some of the previous possible \( O(n^2) \) values may now be eliminated. However, this does not change the order of magnitude of the number of possible values which still remains \( O(n^2) \).

A peripheral vertex \( v_1 \) is defined by the following relation: \( d(v_1, x^*) = \max_{v \in V} \{d(v, x^*)\} \).
\( p = p_1 + p_2, \) such that if \( V_{p_1}^* \) is a vertex \( p_1 \)-center of \( T_1(e_p) \) and \( V_{p_2}^* \) is a vertex \( p_2 \)-center of \( T_2(e_p) \), then \( V_{p}^* = V_{p_1}^* \cup V_{p_2}^* \) is a vertex \( p \)-center of \( T \).

**Proof.** Let \( V_{p}^* = \{u_1, u_2, \ldots, u_p\} \) be a vertex \( p \)-center of \( T \), and for each \( 1 \leq i \leq p \), define \( U_i \) to be the set of all vertices of \( T \) for which \( u_i \) is a closest vertex among the vertices of \( V_{p}^* \): \( U_i = \{v \in V | d(v, u_i) = d(v, V_{p}^*)\} \). If there exists an index \( i_0 \) such that all vertices of \( p(v_1, v_2) \) belong to \( U_{i_0} \), then \( U_{i_0} \) is a vertex 1-center of \( T \) and therefore the vertex \( p \)-radius of \( T \) is equal to its vertex 1-radius. In this case, let \( e_p \) be the (only) edge adjacent to \( v_2 \), and thus \( T_1(e_p) = T - \{v_2\}, T_2(e_p) = \{v_2\} \). Clearly, if \( p_1 = p - 1, p_2 = 1 \), then \( V_{p_1}^* \) is a vertex \( p_1 \)-center of \( T_1(e_p) \) and \( V_{p_2}^* = \{v_2\} \) is a vertex \( p_2 \)-center of \( T \). Hence, suppose that not all the vertices of \( p(v_1, v_2) \) belong to the same set \( U_{i_0} \) and let \( e_p = (v_1, v_2) \) be an edge on \( p(v_1, v_2) \) such that for some index \( i, v_i \in U_i \), but \( v_j \notin U_i \). Let \( p_1 \) and \( p_2 \) be the number of vertices of \( V_{p}^* \) which are on \( T_1(e_p) \) and on \( T_2(e_p) \), respectively; then clearly \( p_1 + p_2 = p \) and \( p_1, p_2 > 0 \). If \( V_{p_1}^* \) and \( V_{p_2}^* \) are the vertex \( p_1 \)-center and the vertex \( p_2 \)-center of \( T_1(e_p) \) and \( T_2(e_p) \), respectively, then \( V_{p}^* = V_{p_1}^* \cup V_{p_2}^* \) is a vertex \( p \)-center of \( T \). Q.E.D.

Now, for any edge \( e \) on \( p(v_1, v_2) \) and for any \( p_1 (1 \leq p_1 < p) \) and \( p_2 = p - p_1 \), let \( r_{p_1} \) be the vertex \( p_1 \)-radius of \( T(e) \) and \( r_{p_2} \) be the vertex \( p_2 \)-radius of \( T(e) \) Denote \( r_{p_1,p_2}(e) = \max \{r_{p_1}(e), r_{p_2}(e)\} \) and \( r_{p_2}(e) = \min_{e \in p(v_1,v_2)} \{r_{p_1,p_2}(e)\} \). Then, clearly (by Lemma 5.1) \( r_p = \min \{r_{p_1,p_2}(e) | 1 \leq p_1 < p \) and \( p_2 = p - p_1 \}) \) is the vertex \( p \)-radius of \( T \). The following recursive algorithm is based on finding a minimum of the set \( \{r_{p_1,p_2}(e) | 1 \leq p_1 < p \) and \( p_2 = p - p_1 \}) \) where each \( r_{p_1,p_2}(e) \) is found by performing a binary search on the edges of \( p(v_1, v_2) \).

**Algorithm 5.1.** A vertex \( p \)-center (\( p > 1 \)) of a vertex-unweighted tree.

1. Find two peripheral vertices \( v_1 \) and \( v_2 \) of \( T \) such that \( d(v_1, v_2) = \max_{v,v' \in V} \{d(v, v')\} \). Let \( p(v_1, v_2) \) be the path which connects \( v_1 \) and \( v_2 \) in \( T \).
2. For each \( p_1 (p_1 = 1, \ldots, p - 1) \) do steps 3-4.
3. Perform a binary search over the edges of \( p(v_1, v_2) \) to find \( \min_{e \in p(v_1,v_2)} \{r_{p_1,p_2}(e)\} \)
   where for each edge \( e \) which is searched, do the following:
   (a) Find a vertex \( p_1 \)-center \( V_{p_1}^* (e) \) and the vertex \( p_1 \)-radius \( r_{p_1}(e) \) of \( T_1(e) \)
       where \( T_1(e) \) is the connected subtree of \( T - \{e\} \) which contains \( v_1 \).
   (b) Find a vertex \( p_2 \)-center \( V_{p_2}^* (e) \) and the vertex \( p_2 \)-radius \( r_{p_2}(e) \) of \( T_2(e) \)
       where \( T_2(e) \) is the connected subtree of \( T - \{e\} \) which contains \( v_2 \).
   (c) Let \( r_{p_1,p_2}(e) = \max\{r_{p_1}(e), r_{p_2}(e)\} \).
4. Let \( e_{p_1,p_2} \) be the edge of \( p(v_1, v_2) \) such that \( r_{p_1,p_2}(e_{p_1,p_2}) = \min_{e \in p(v_1,v_2)} \{r_{p_1,p_2}(e)\} \)
   [this edge was found by the binary search of step 3]. Denote: \( r_{p_1,p_2} = r_{p_1,p_2}(e_{p_1,p_2}) \)
   and \( V_{p_1}^* = V_{p_1}^* (e_{p_1,p_2}) \cup V_{p_2}^* (e_{p_1,p_2}). \)
5. Assign: \( r_p = \min\{r_{p_1,p_2} | 1 \leq p_1 < p \) and \( p_2 = p - p_1 \}) \) and let \( V_{p}^* \) be the vertex \( p \)-radius of \( T \) and \( V_{p}^* \) is a vertex \( p \)-center of \( T \).

The complexity of this algorithm is \( O(n \cdot \log^{p-1} n) \). This can be proved by induction: For, suppose that this time bound is correct for finding an \( m \)-center where \( m < p \) (clearly this is correct for \( p = 2 \)). Then, when we want to find a \( p \)-center of \( T \), each execution of step 3(a) requires \( O(n \cdot \log^{p-1} n) \), while step 3(b) requires \( O(n \cdot \log^{p-1} n) \). Because of the binary search, for each value \( p_1 (1 \leq p_1 < p) \), step 3 is repeated \( O(\log n) \) times, and therefore, for each \( p_1 \), the total work done in steps 3-4 is \( O(n \cdot \log^{p_1} n + n \cdot \log^{p-p_1} n) \). Therefore, the complexity of the loop of steps 2-3-4 is

\[
\sum_{p_1=1}^{p-1} O(n \cdot \log^{p_1} n + n \cdot \log^{p-p_1} n) = O\left(2n \sum_{p_1=1}^{p-1} \log^{p_1} n\right) = O\left(2n \frac{\log^{p_1} n - \log n}{\log n - 1}\right) = O(n \cdot \log^{p-1} n).
\]

Thus, Algorithm 5.1 works on \( O(n \cdot \log^{p-1} n) \) time, which is less than \( O(n^2) \).
Appendix: Glossary.

An absolute $p$-center: A set $X_p^*$ of $p$ points (not necessarily vertices) on a graph $G$, such that for any other set $X_p$ of $p$ points on $G$, the following relation holds:

$$\max_{v \in V} \{w(v) \cdot d(v, X_p^*)\} = \min_{X_p \text{ on } G} \left[ \max_{v \in V} \{w(v) \cdot d(v, X_p)\} \right].$$

The absolute $p$-radius: If $X_p^*$ is an absolute $p$-center of $G$, then $r_p = \max_{v \in V} \{w(v) \cdot d(v, X_p^*)\}$ is called "the absolute (weighted) $p$-radius of $G$". [If, for all $v \in V$, $w(v) = c$, then $r_p$ is the absolute unweighted $p$-radius of $G$.]

A centroid of a tree: Let $T_{vi}$ ($1 \leq i \leq d_v$) be the connected subtrees which are obtained from a tree $T$ by removing a vertex $v$ of degree $d_v$. Then, a centroid of the tree is a vertex $c_v$ for which

$$\max_{1 \leq i \leq d_v} \{|T_{vi}|\} = \min_{v \in V} \left\{ \max_{1 \leq i \leq d_v} \{|T_{vi}|\} \right\}.$$ 

$D_e(v, t)$ Let $e = (v_r, v_v)$ be an edge of $G$ of length $l(e)$. Then each point $x$ of $e$ is given by a parameter $t(x)$ ($0 \leq t \leq l(e)$), where $t$ gives the distance on $e$ of $x$ from $v_r$. $D_e(v, t)$ is defined as the weighted distance from a vertex $v$ on $G$ to a point $t$ on $e$:

$$D_e(v, t) = w(v) \cdot \min \{(t + d(v_r, v), (l(e) - t) + d(v_v, v))\}.$$ 

$D_e(t)$

The distance matrix: An $n \times n$ matrix whose $ij$th entry is the distance between vertex $v_i$ and vertex $v_j$.

The distance between a point and a set of points:

The distance $d(y, X_p)$ between a point $y$ on a graph $G$ and a set $X_p$ of $p$ points on $G$ is defined by:

$$d(y, X_p) = \min_{x \in X_p} \{d(y, x)\}$$

where $d(y, x)$ is the distance between the points $y$ and $x$ on $G$, namely, the length of the shortest path which connects $y$ and $x$ on $G$.

(Absolute, vertex) dominating set of radius $r$:

If $p$ is the domination number of radius $r$, then every (absolute, vertex) $p$-center of $G$ is an (absolute, vertex) dominating set of radius $r$.

(Absolute, vertex) domination number of radius $r$:

This is the smallest positive integer number $p$ such that the (absolute, vertex) $p$-radius of $G$ is not greater than $r$.

Dominating set [Domination number]:

A vertex dominating set [the vertex domination number] of radius 1 of a graph $G$ all whose edges are assumed to be of length 1 and all whose vertices have weight 1.

Local-center (local-radius) on edge $e$:

A local-center on edge $e$ is any point $x^*$ on $e$ such that

$$\max_{v \in V} \{w(v) \cdot d(v, x^*)\} = \min_{x \text{ on } e} \left[ \max_{v \in V} \{w(v) \cdot d(v, x)\} \right],$$

namely, it is an absolute 1-center of $G$ which is restricted to...
A vertex p-center: A subset $V^*_p$ of $p$ vertices of $G$ such that for any subset $V_p$ of $p$ vertices of $G$ the following relation holds:

$$\max_{v \in V} \{w(v) \cdot d(v, x^*)\} = \min_{V_p \subset V} \left[ \max_{v \in V_p} \{w(v) \cdot d(v, V_p)\} \right].$$

Namely, it is a p-center whose points are restricted to be vertices.

The vertex p-radius: If $V^*_p$ is a vertex p-center of $G$, then $\max_{v \in V} \{w(v) \cdot d(v, V^*_p)\}$ is the vertex p-radius of $G$.

A vertex-weighted [vertex unweighted] graph (network):

A 2–3 tree:

It is a tree in which each vertex which is not a leaf has 2 or 3 sons, and every path from the root to a leaf is of the same length (see [12]).

REFERENCES


