The obnoxious center problem on a tree *

Rainer E. Burkard † Helidon Dollani † Yixun Lin ‡ Günter Rote §

June 15, 1998, revised July 26, 2001

Abstract. The obnoxious center problem in a graph G asks for a location on an edge of the graph such that the minimum weighted distance from this point to a vertex of the graph is as large as possible. We derive algorithms with linear running time for the cases when G is a path or a star, thus improving previous results of Tamir. For subdivided stars we present an algorithm of running time $O(n \log n)$. For general trees, we improve an algorithm of Tamir by a factor of $\log n$. Moreover, a linear algorithm for the unweighted center problem on an arbitrary tree with neutral and obnoxious vertices is described.

Keywords: Location problems, center problem, obnoxious facilities, linear-time algorithm

AMS-classification: 90B80, 90B85, 90C35, 90C27

1 Introduction

In the center problem, a set of clients on certain locations (sites) is given. The center problem asks for finding a location for a new facility from which the farthest client (site) can be reached in minimum time. It occurs if a fast service in the case of an emergency is needed. This problem has received strong interest since Hakimi (1964) published the first paper on this topic, see also Kariv and Hakimi (1979). For a survey on center problems see Handler (1990), for a study of algorithms with respect to a good complexity see Megiddo and Tamir (1983). In particular the case has been considered that the clients are modeled as vertices of a tree. In this case it has been shown that the objective function is convex on each path, which implies that a local optimum solution is also a global optimum. Exploiting this, Megiddo (1983) gave a linear algorithm for the center problem in trees.

Recently, *obnoxious* location problems find an increasing interest. In contrast to the usual problem, the center should be *as far away as possible* from the given sites. Thus we have to maximize the minimum (weighted) distance from the center to the sites. A formal definition of the problem is given in Section 2. Such a problem occurs for example if the center is a facility which produces toxic agents which should be as far away as possible from the given locations of cities.

*This research has been supported by the Spezialforschungsbereich F 003 "Optimierung und Kontrolle", Projektbereich Diskrete Optimierung.

†Technische Universität Graz, Institut für Mathematik, Steyrergasse 30, A-8010 Graz, Austria. e-mail addresses: burkard@opt.math.tu-graz.ac.at, dollani@opt.math.tu-graz.ac.at

‡Department of Mathematics, Zhengzhou University, Zhengzhou 450052, P.R.China. e-mail address: linyixun@mail.zzu.edu.cn

§Freie Universität Berlin, Institut für Informatik, Takustraße 9, D-14195 Berlin, Germany. e-mail address: rote@inf.fu-berlin.de

Another case where this model is applicable is a facility which is to be located as far away as possible from *obnoxious sites*. The center in this case might be some sensitive facility like an observatory or a radio station, which is affected by moisture from lakes, pollution from cities, traffic from airports, or the like. In such models, it is unnatural to use the metaphor of "clients" for the given locations; thus we prefer the neutral term *sites*.

Complexity issues regarding the placement of several facilities in an obnoxious setting were considered by Tamir (1991). If we again consider the model where the sites are vertices of a tree, the objective function is no longer convex or concave. Drezner and Wesolowsky (1985) solve the obnoxious center problem on a tree with n vertices in $O(n^3)$ time. Tamir (1991, 1988) gives two algorithms of $O(n \log^2 n)$ and $O(kn \log^2 n)$ time, respectively, where k is a parameter that depends on the structure of the tree. In Section 6 we will make a simple observation which reduces the time bound of the second algorithm to $O(kn \log n)$ time by the use of different data structures. Tamir shows that the obnoxious center problem on a path or a star with n vertices can be solved in $O(n \log n)$ time. In this paper we show that the center problem on a path or a star can even be solved in linear time (in Sections 3 and 4, respectively). In Section 5, we treat extended stars, which can be obtained from stars by subdividing edges and introducing additional vertices on them. We show that an obnoxious center in an extended star graph with b branches can be found in $O(n + b \log n)$ time.

In Section 6 we design as well a linear algorithm for the obnoxious center problem on a tree where all sites have the same weight. Even in this case the objective function does not have any useful convexity properties. Obnoxious center problems for locating a center in the *plane* have also been considered. Melachrinoudis and MacGregor Smith (1995) used weighted Voronoi diagrams to find a weighted obnoxious center inside a convex m-gon with n sites in $O(mn^2)$ time.

The counterpart to the center problem is the median problem, where a location should be found such that the *sum* of weighted distances from the median to the sites is minimized. A version including obnoxious sites has recently been treated in Burkard and Krarup (1998). They showed that the median problem where the friendly and/or obnoxious sites correspond to the vertices of a cactus, can be solved in linear time. (A cactus is a graph where any two cycles have at most one vertex in common.) Some further questions for future research will be mentioned in the concluding Section 7.

2 Obnoxious center problems

Let $G = (V, E)$ be a simple graph with a set of n vertices $V = \{v_1, \ldots, v_n\}$ and a set E of m edges. Each edge $(v_i, v_j) \in E$ has a positive length c_{ij}. Thus we can interpret each edge as the image of a closed real interval $[0, c_{ij}]$ of length c_{ij}. For any point z in this interval we have a corresponding point P, and we define its distance to v_i as $|z|$ and its distance to v_j as $|c_{ij} - z|$. This enables us to define the shortest distance between any point P on an edge of G and a vertex v of G. The distance between P and v, denoted by $d(P, v)$, is thus the length of a shortest path from P to v. Moreover, we consider each vertex v_i of G as a site (client) and attach a positive weight w_i to it. The *center problem* on a graph G is to minimize

$$f(P) = \max_{v_i \in V} \; w_i d(P, v_i)$$

over all points P on the edges of G. This objective function reflects the goal to locate a facility (center) P as close to the clients v_i as possible, so that the clients can quickly get services from the center in case of an emergency.

In the case of an *obnoxious* facility one wants to maximize

$$g(P) = \min_{v_i \in V} \; w_i d(P, v_i).$$

This objective function places the new location as far away as possible from the sites (vertices) v_i of G. Note that the significance of weights is contrary to the usual case: Important and highly sensitive sites (or highly obnoxious sites) receive small weights. Vertices which contain no sites should get weight ∞.

3 Obnoxious center problems on paths

If the input graph is a path, we may put the n vertices on the real line and identify them with real numbers such that

$$0 = x_1 < x_2 < \cdots < x_n$$

and $d(x_i, x_j) = |x_i - x_j|$. Then the objective function is

$$g(z) = \min\{\, w_i |z - x_i| : i = 1, \ldots, n \,\} = \min\{g^+(z), g^-(z)\},$$

with

$$\begin{aligned} g^+(z) &= \min\{\, w_i(z - x_i) : x_i \le z \,\}, \\ g^-(z) &= \min\{\, w_i(x_i - z) : x_i \ge z \,\}. \end{aligned}$$

In this section we give a linear time algorithm which finds the maximum of $g(z)$ along a path. We first describe how to compute $g^+(z)$ from left to right. We can compute $g^-(z)$ in an analogous ways from right to left, and then it is easy to compute $g(z)$ and to find the maximum.

We incrementally compute the functions g_1^+, g_2^+, ..., g_{n-1}^+, which are defined as follows:

$$g_j^+ \colon [x_j, x_n] \to \mathbb{R}_{\ge 0}\,, \text{ with } g_j^+(z) := \min\{\, w_i(z - x_i) : i = 1, \ldots, j \,\}$$

The following properties are straightforward consequences of the definition, except for the statement about the number of breakpoints in part (e), which we shall prove later.

Lemma 1 (a) *For $x_j \le z < x_{j+1}$, we have $g^+(z) = g_j^+(z)$.*

(b) *g_j^+ is a piecewise linear concave and increasing function.*

(c) $g_1^+(z) = w_1(z - x_1)$.

(d) *For $j = 2, \ldots, n-1$, $g_j^+(z) = \min\{w_j(z - x_j), g_{j-1}^+(z)\}$ in the domain of g_j^+ (i. e., for $z \ge x_j$).*

(e) *The function $g^+(z)$ is piecewise linear and has at most $2n - 3$ linear pieces.*

Lemma 1(b) and (e) suggests a way to represent the functions g_j^+ and g^+: as a list of adjacent *intervals*, together with the coefficients a, b of a linear function $az + b$ for each interval. Actually, the list for g_j^+ can be conveniently organized as a stack, because we will only modify the list at its left end. In the sequel, when we speak of an interval, we will include the coefficient of a linear function defined on that interval without mentioning it.

Lemma 1(c–d) opens the way for an incremental construction of these lists, see Figure 1.

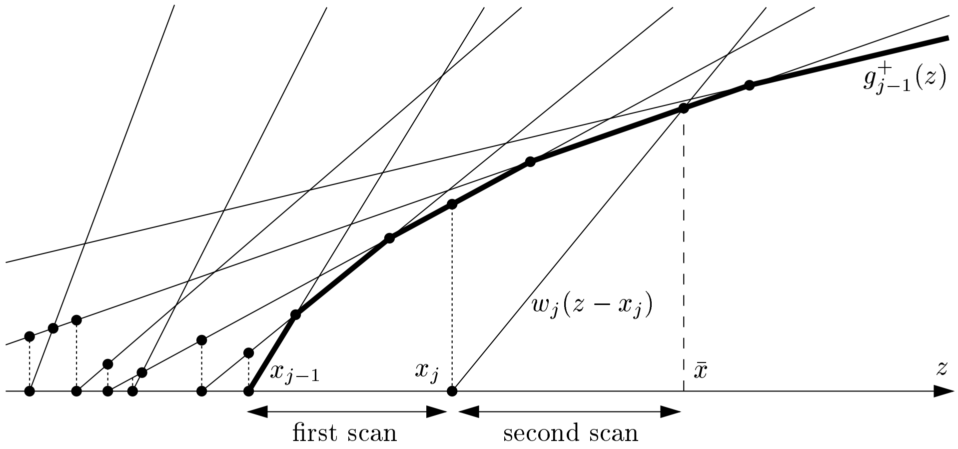

Figure 1: The construction of g^+_j from g^+_{j-1}.

Incremental Step: Construction of g^+_j from g^+_{j-1}. **First Scan.** We scan the list of intervals of g^+_{j-1} from x_{j-1} until we reach x_j. This list of intervals is removed and copied to the list of intervals that define the function g^+, according to Lemma 1(a). In general, the point x_j lies in the middle of an interval. That interval is then split into a left part, which is contributed to g^+, and the right part, which remains as part of the definition of g^+_{j-1}.

Second Scan. After this transformation, the domain of the function has been reduced to $[x_j, x_n]$, which is the domain of the function g^+_j that we wish to compute. We now apply Lemma 1(d). The function g^+_j will start with an interval where $g^+_j(z) = w_j(z - x_j)$. This interval extends until this linear function intersects the graph of $g^+_{j-1}(z)$; from then on, $g^+_j(z)$ coincides with $g^+_{j-1}(z)$. More precisely, this is done as follows.

We scan the list of (remaining) intervals of g^+_{j-1} from the left and compare each interval to the function $w_j(z - x_j)$; as long as this function is smaller than g^+_{j-1} in the whole range of the interval, we remove that interval from the list. When the two functions intersect, we split the interval at the intersection point $\bar{x}$, and we replace the left half by an interval $[x_j, \bar{x}]$ where the function $w_j(z - x_j)$ is used.

This concludes the construction of g^+_j.

We can now prove the bound of $2n-3$ on the total number of linear pieces of g^+. Initially, g^+_1 has one piece. In the first scan, each piece that is contributed to the final function g^+ is removed from g^+_{j-1}, except for one additional piece that results from splitting one interval. In the second scan, the function g^+_j gets one additional piece (and it may lose other pieces). This gives a total of $1 + (n-2)(1+1) = 2n-3$ pieces altogether, taking into account that the last iteration terminates by copying everything from g^+_{n-1} to g^+ in the first scan.

It is easy to see that the number of $2n-3$ pieces can actually be attained.

The computation of g^- proceeds in the same way from right to left. Finally, we vary z from x_1 to x_n and compute $g(z) = \max\{g^+(z), g^-(z)\}$, simultaneously scanning the two lists of intervals for g^+ and g^-, and we return the solution z attaining the maximum of this function.

Theorem 1 *The above algorithm solves the weighted obnoxious center problem on a path in linear time.*

Proof. The computation of g^+ (and g^-) takes linear time: each interval that is looked at in the first scan, contributes one piece to the function g^+. So the total time for the first scan is $O(n)$, by Lemma 1(e). The time for the second scan is $O(1 + k)$, where k is the number of intervals that are removed from the list of intervals. Since the total number of removed intervals cannot be bigger than the total number of intervals that were ever added to the list, the total time for the second scan is also $O(n)$. Note that the comparison of two linear functions and determining the intersection point in each interval can be executed in constant time.

The final scan of the algorithm is easily done in $O(n)$ time. ■

We remark that the essence of the above algorithm for computing g^+ is the same as an incremental algorithm for computing the intersection of half-spaces $H_1 \cap \cdots \cap H_i$, for $i = 1, \ldots, n$, if the half-spaces are inserted *in the order of their intersection with a fixed line* (the x-axis in our case). In our case, we have the half-spaces $H_j := \{\, (z, y) : y \le w_j(z - x_j) \,\}$, and we are actually only interested in the part lying above the x-axis. The algorithm is geometrically dual to (and algebraically identical to) an incremental algorithm for computing the convex hull for points in the plane which are *sorted by x-coordinate*, cf. for example Preparata and Shamos (1985) for a description of this duality and for the linear-time convex hull algorithm for sorted points.

The unweighted version of this problem is essentially the MAX-GAP problem of finding the longest edge (or the *maximum gap*) between two successive numbers which can be solved in linear time even if the numbers x_i are not given in sorted order, see Gonzalez (1975).

4 Obnoxious centers in star graphs

A *star* is a complete bipartite graph $K_{1,n}$. It is a tree $T = (V, E)$ consisting of a central vertex v_0 which has edges to n other vertices $\{v_1, \ldots, v_n\}$. Denote $x_i := c_{0i}$, for $i = 1, \ldots, n$, and $x_0 = 0$. The subproblem of determining a locally optimum solution on the edge (v_0, v_i) will be denoted by S_i. It is equivalent to the problem on a path as follows: we put all vertices

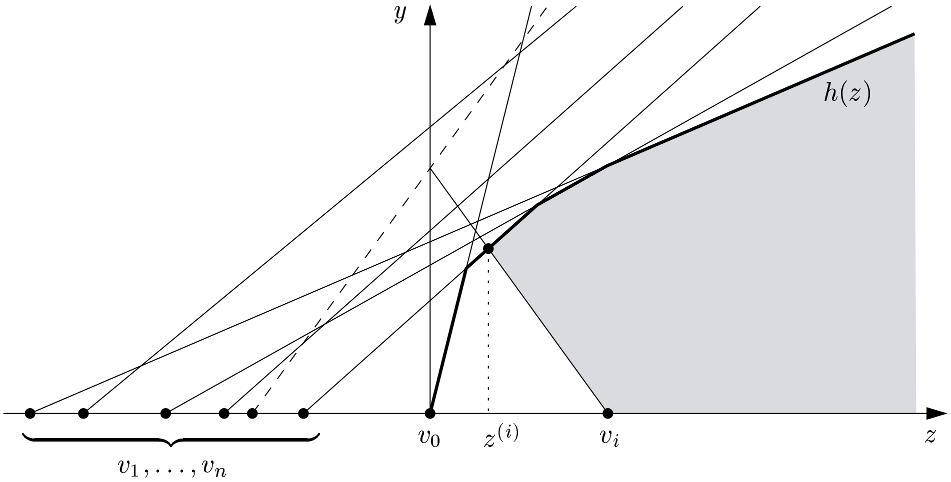

Figure 2: Graphical representation of subproblem S_i. The shaded region is the feasible region of (3). Adding the constraint indicated by the dotted line does not change the problem.

on the real line such that $v_0 = 0$, v_i is to the right of v_0 with distance $|v_i - v_0| = x_i$, and all other vertices v_j $(j \ne i)$ are to the left of v_0 with distance $|v_j - v_0| = x_j$. In problem S_i we have to maximize the function

$$g_i(z) = \min\{\min_{j \ne i} w_j(x_j + z),\ w_i(x_i - z)\} \tag{1}$$

for $0 \leq z \leq x_i$ (see Fig. 2). We can omit the condition $j \neq i$ from (1) without changing

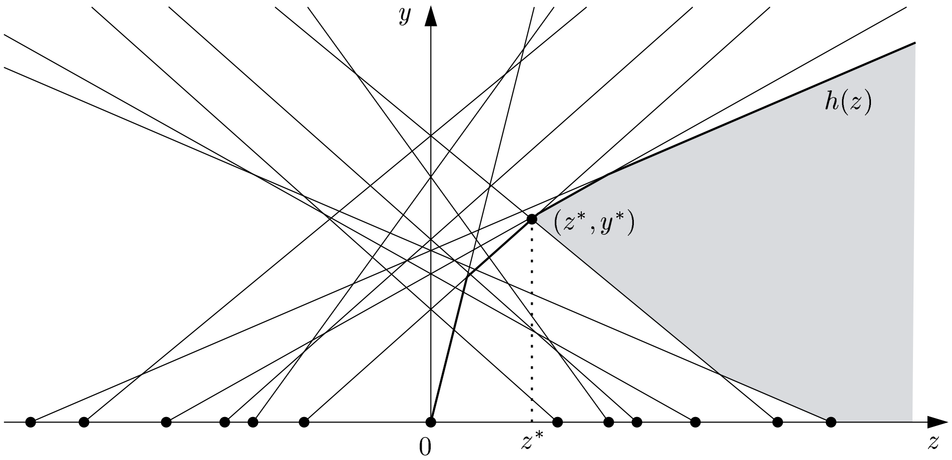

Figure 3: The feasible region of the linear program (4)

the problem because, for $z \geq 0$, $w_i(x_i + z)$ is always larger than the second term of the expression, $w_i(x_i - z)$:

$$g_i(z) = \min\{\min_{j=0,\ldots,n} w_j(x_j + z),\ w_i(x_i - z)\} \tag{2}$$

Let the maximum be attained in $z^{(i)}$. Obviously, the point $z^{(i)}$ is a solution of the following linear program in two variables y and z (see Fig. 2):

$$\min\{\, z : y \leq w_j(x_j + z) \text{ for } j = 0, \ldots, n,\ \ y \geq w_i(x_i - z)\,\} \tag{3}$$

The constraints which are common to all problems S_i can be written as $y \leq h(z)$, where

$$h(z) := \min_{j=0,\ldots,n} w_j(x_j + z).$$

Obviously, $h(z)$ is a piecewise linear and increasing function. The point $z^{(i)}$ is the intersection point of $h(z)$ with $w_i(x_i - z)$. Due to the monotonicity of $h(z)$, the obnoxious center problem asks for $z^* := \max z^{(i)}$. But this z^* can now be obtained as optimal solution of the linear program (see Fig. 3)

$$\min\{\, z : \ y \leq w_j(x_j + z) \text{ for } j = 0, \ldots, n;\ \ y \geq w_j(x_j - z) \text{ for } j = 1, \ldots, n\,\}. \tag{4}$$

Theorem 2 *Consider an optimal solution (z^*, y^*) of the linear program (4). Then the optimal objective function value of the obnoxious center problem is y^*, and the optimal locations problem are the points at distance z^* from the central vertex v_0 on all edges (v_0, v_i) for which $y^* = w_i(x_i - z^*)$ holds.*

Proof. Note first that $h(z^*) = y^*$, and $y^* = w_i(x_i - z^*)$ must hold for at least one index i, because otherwise there would be a solution of (4) with $z < z^*$. It is easy to check that the locations which are constructed in the theorem have the claimed objective function value.

We still have to show that there is no other solution. Consider a point P on edge (v_0, v_i) at distance z from v_0. If $z < z^*$, then $h(z) < h(z^*) = y^*$, and hence there is a site v_j whose distance $d(P, v_j)$ from P is $h(z)/w_j$, which means that $w_j d(P, v_j) < y^*$. If $z > z^*$, or if $z = z^*$

and i is not one of the indices j for which $y^* = w_j(x_j - z^*)$ holds, then $y^* > w_i(x_i - z^*)$ and $d(P, v_i) = x_i - z < y^*/w_i$, and thus $w_i d(P, v_i) < y^*$. ■

The linear program (4) has $2n$ constraints and 2 variables. By the algorithm of Megiddo (1983) it can be solved in linear time. Thus the obnoxious center in a star graph can be found in linear time.

5 Obnoxious centers in extended star graphs

An *extended* star graph is a tree which has a single vertex v_0 with degree greater than 2. The remaining vertices form paths from v_0 to the leaves of the graph. We call these paths the *branches* of the tree. This class of graphs is a mixture between paths and stars, which were considered in the previous two sections. We will show that an obnoxious center in an extended star with n vertices and b branches can be found in $O(n + b \log n)$ time. When b is relatively small, the algorithm runs in linear time. However, when for example all branches contain two edges and $b \approx n/2$, the time complexity is $O(n \log n)$. We do not see how to solve the problem in linear time even in this special case.

As in Section 4, we denote $x_i := c_{0i}$. First we consider a local subproblem for each branch separately: We move the center P on the i-th branch using $z := d(P, v_0)$ as a parameter. We construct the "local" objective function $g_i(z)$, considering only sites on the i-th branch (starting at v_0) and ignoring all vertices on other branches. This function is defined on some interval $0 \le z \le Z_i$, where Z_i is the length of the branch. The vertex v_0 corresponds to $z = 0$. By the methods of Section 3, all piecewise linear functions g_i can be constructed in linear time.

We also consider the function

$$h(z) := \min_{j=0,\ldots,n} w_j(z + x_j).$$

As in the previous section, the optimum value is now given by

$$y^* = \max_{i=1,\ldots,b} \max_{0 \le z \le Z_i} \min\{h(z), g_i(z)\}. \tag{5}$$

It is clear that this optimum is located either at a local maximum of some function g_i or at an intersection point of the graph of g_i with the graph of h.

The overall approach for solving this problem can roughly be described as follows. The function h is a piecewise linear concave and increasing function. We perform a binary search among its breakpoints $(\hat{z}, \hat{y})$ to find the range of y values in which the optimum value lies. To do this, we need to test whether $y^* \ge \hat{y}$, for a given point $\hat{y} = h(\hat{z})$ on the function h.

This test is carried out as follows. We successively look at each function g_i, decreasing z from the maximum permitted value Z_i down to $\hat{z}$. We stop this scan as soon as some value $g_i(z) \ge \hat{y}$ with $z \ge \hat{z}$ is found. We have found a feasible solution with value $\hat{y}$ and hence $y^* \ge \hat{y}$. On the other hand, if we have scanned all domains $z \ge \hat{z}$ for all branches i without finding a value $g_i(z) \ge \hat{y}$, we know that $y^* < \hat{y}$.

As we scan the functions g_i, we remember the highest value $\tilde{y}$ that we have encountered. If we later get another query with a different point $(\hat{z}, \hat{y})$, we may be able to answer immediately because $\tilde{y} \ge \hat{y}$. Otherwise, we continue the right-to-left scan of each function g_i at the value z where we left off during the last previous scan of this function.

Note that we scan only intervals where we know that $g_i(z) \le h(z)$. This means that linear pieces of g_i which were examined need not be examined again, because the feasible value $\tilde{y}$

is an upper bound of $g_i(z)$ over all intervals that were examined. So the time complexity for answering a sequence of tests of the condition $y^* \geq \hat{y}$ is bounded by the total number of pieces of all functions g_i, which is $O(n)$, plus an overhead of $O(b)$ for each test. The overhead comes from the fact that we may have to spend constant time for each branch i to just "look at" this branch: For example, if we stopped the previous scan of g_i because the point $\hat{z}$ was reached, we may have to repeatedly examine the single linear piece to which this point belongs.

We will now describe more precisely how the binary search among the breakpoints of h is carried out. The function h is the lower envelope of $n+1$ increasing linear functions, whose slopes are given by the weights w_i. We start by finding the median $\hat{w}$ of the $n+1$ slopes and identifying the leftmost point $\hat{y} = h(\hat{z})$ on the graph where the slope becomes $\leq \hat{w}$. This point can be identified by solving the linear program

$$\max\{\, y - \hat{w}z : z \geq 0,\ y \leq w_j(z + x_j) \text{ for } j = 0, \ldots, n \,\} \tag{6}$$

in the two variables y and z. Actually, this linear program may yield *any* point with slope $\hat{w}$, not necessarily the *leftmost* point with slope $\hat{w}$ or smaller. The leftmost point can be found by perturbing the objective function or by solving the following auxiliary linear program:

$$\min\{\, z : y - \hat{w}z = K^*,\ z \geq 0,\ y \leq w_j(z + x_j) \text{ for } j = 0, \ldots, n \,\},$$

where K^* is the optimum value of (6). (This is a linear program in only one variable, after using the equation to eliminate y.)

Now we test the condition $y^* \geq \hat{y}$ as described above. If we find that $y^* \geq \hat{y}$, we can discard the first half of the linear functions, with slopes $> \hat{w}$, from consideration in $h(z)$, because we know that the optimum cannot lie in the range $z < \hat{z}$. Otherwise, if $y^* < \hat{y}$, we can discard the other half of the linear functions from the definition of $h(z)$, because they play no role for restricting the optimum of (5). (In fact, in this case, we have actually scanned the part with $z \geq \hat{z}$ of all branches, and we can restrict the remaining search to the range $z < \hat{z}$.)

We continue this process by finding the median of the remaining pieces of h, and so on, until only one linear piece of h is left. It is then easy to find the optimum value of (5) directly in linear time.

Since the median of n numbers can be found in linear time by the algorithm of Blum, Floyd, Pratt, Rivest, and Tarjan (1973), (see also Aho, Hopcroft, and Ullman 1983), the overall effort for the binary search is

$$O(n) + O(n/2) + O(n/4) + \cdots = O(n).$$

To this we must add the effort for the $O(\log n)$ queries, which is

$$O(\log n) \cdot O(b) + O(n),$$

as discussed above. Summarizing, we have

Theorem 3 *The weighted obnoxious center problem on an extended star tree with n vertices and b branches can be found in $O(n + b\log n)$ time and $O(n)$ space.*

6 The obnoxious center problem in general trees

In this section we consider the obnoxious center problem in weighted and unweighted trees.

6.1 Finding an obnoxious center in weighted trees

Tamir (1991, 1988) gives two algorithms of $O(n \log^2 n)$ and $O(kn \log^2 n)$ time complexity, respectively, for solving the obnoxious center problem on an arbitrary tree with n vertices. The parameter k depends on the structure of the tree. For paths and stars $k = O(1)$, for balanced trees $k = O(\log n)$, but there exist trees such that $k = \Theta(n)$. By an easy observation Tamir's algorithm of 1988 can be improved by a factor of $\log n$. Tamir notes that, if the center is restricted to a single edge, the objective function is a lower envelope of n linear functions. When one goes from an edge to an adjacent edge, not all of these linear functions have to be changed. We can obtain the lower envelope of the functions for the adjacent edge by removing some linear functions and adding new ones. Tamir showed that one can successively obtain the objective function for all edges with a total of $O(kn)$ insertions and deletions of linear functions.

Tamir used the data structure of Overmars and van Leeuwen (1981) for maintaining a lower envelope of n linear functions under deletions and insertions. This data structure takes linear space and $O(\log^2 n)$ time for a deletion or insertion. The maximum of the current lower envelope over some given interval can be found in $O(\log n)$ time.

However, the sequence of $O(kn)$ insertions and deletions of linear functions can be computed beforehand, and thus we can use the algorithm of Hershberger and Suri (1996) for an *off-line* maintenance of the lower envelope of linear functions. This data structure needs only $O(n \log n)$ time to process a sequence of n insertions, deletions, and queries for the maximum, i.e., only $O(\log n)$ time per operation on the average. In total, this reduces the complexity to $O(kn \log n)$. Note that for small k ($k = O(\log n)$), this time complexity bound is lower than the $O(n \log^2 n)$ bound of Tamir (1991).

6.2 A linear algorithm for finding an obnoxious center in unweighted trees

Let us determine an obnoxious center in the tree $T = (V, E)$ with edge lengths c_{xy}. The center can be placed in any vertex or on any edge of the tree, but should be as far away as possible from the vertices of some given set $V_0 \subseteq V$ of obnoxious sites. Let us call the vertices in V_0 *black vertices* and the vertices outside of V_0 *white vertices*. Thus our problem is to maximize

$$g(z) = \min_{v \in V_0} d(z, v).$$

The objective function $g(z)$ is not necessarily concave along a path. For example, in the tree shown in Figure 4 (V_0 consists of the 4 black vertices), $g(z)$ is neither convex nor concave along the path (a, b, c, d).

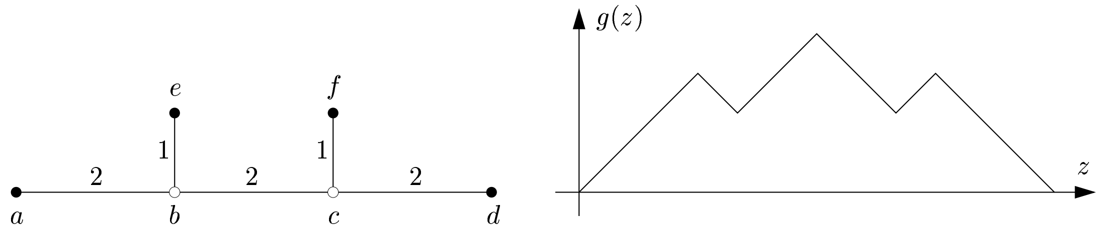

Figure 4: Nonconvexity of objective function $g(z)$

Since $g(z)$ is concave on each edge, an $O(n^2)$ algorithm is easy to realize by examining every edge. In the following we describe a linear algorithm.

We first select an arbitrary vertex r as the root of the tree. Then we perform a sweep from the leaves to the root, and for each vertex u, we compute the minimum distance $g^+(u)$ from u to a black vertex in the subtree below u (including u itself). Finally, during a root-to-leaf sweep, for each vertex u, we compute the minimum distance $g(u)$ from u to any black vertex, and we also locate the optimal point on each edge.

Phase I. We denote the set of children of the vertex u by $S(u)$. We can set $g^+(u) := 0$ for all black vertices. For white vertices we have

$$g^+(u) = \min_{v \in S(u)} (c_{uv} + g^+(v)).$$

This includes the case of white leaves ($S(u) = \emptyset$), for which we can initialize $g^+(u) = \infty$. We then inductively compute $g^+(u)$ for all other white vertices, proceeding from the leaves towards the root.

Phase II. We proceed from the root to the leaves. At the root r, we have $g(r) := g^+(r)$.

Now consider an edge (u, v) from a vertex u to its child v. We assume that $g(u)$ has already been determined correctly. For a point P at distance z from u on this edge, we claim that the minimum distance to a black vertex is

$$\min\{g(u) + z,\ g^+(v) + (c_{uv} - z)\}, \text{ for } 0 \leq z \leq c_{uv}. \tag{7}$$

The second expression is clearly equal to the minimum distance from P to the nearest black vertex in the subtree of v. On the other hand, if the path from P to the nearest black vertex goes through u, its length is represented by the first expression in the above formula. It follows that the expression (7) is certainly not bigger than the minimum distance from P to a black vertex.

It is possible that the first expression $g(u) + z$ does not correspond to a path from P to a black vertex: It may represent a walk that starts by going from P to u, returns to v, and continues into the subtree of v. This happens precisely when the closest black vertex of u lies in this subtree. But then we must have $g(u) = c_{uv} + g^+(v)$, and the second expression is smaller than the first. We conclude that there is a black vertex whose distance from P equals (7), and hence the claim is true.

We can now determine the optimal location for the center on the edge (u, v) in constant time by maximizing (7) over all z. For $z = c_{uv}$, we get $P = v$, and hence

$$g(v) := \min\{g(u) + c_{uv}, g^+(v)\}.$$

This formula allows us to determine $g(v)$ for every vertex v from the value $g(u)$ of its parent u, and we can inductively find $g(v)$ for all vertices.

It is obvious that the above procedure takes linear time. Thus we have shown the following theorem.

Theorem 4 *The unweighted obnoxious center problem on a tree can be solved in linear time.*

7 Concluding Remarks

In the study of obnoxious center problems, Tamir (1988, 1991) presented $O(n \log n)$ algorithms for path trees and star trees, and an $O(n \log^2 n)$ algorithm for general trees. For the extremal cases, i.e., for paths (the trees with largest diameter) and for stars (the trees with smallest diameter), as well as for unweighted trees, we have obtained $O(n)$ algorithms in this note. The question whether one can get linear time algorithms for general trees remains open.

For the multifacility obnoxious center problem on a path, an approach based on the $O(n \log n)$ algorithm of Tamir (1988) significantly improved the $O(n^3)$ bound of Drezner and Wesolowsky (1985). Now, by using our linear algorithm of Section 3, the bound can be further improved to $O(n)$.

A natural generalization of the center problem and the obnoxious center problem is to combine the two objective functions $f(x)$ and $g(x)$ for locating a center x. A similar approach was proposed for the generalized median problem by Burkard and Krarup (1998). On one hand, we may view

$$f(z) = \max_{v_i \in V_+} w_i d(z, v_i)$$

as service cost for friendly sites in $V_+ \subseteq V$ in case of emergency, where $w_i > 0$ for $v_i \in V_+$. On the other hand, we may view

$$g(z) = M + \max_{v_i \in V_-} w_i d(z, v_i)$$

as the damage cost of the obnoxious sites in $V_- \subseteq V$ in case of an emergency, where $w_i < 0$ for $v_i \in V_-$ and $M > 0$ is a constant. Let p, q be the probabilities of these two kinds of emergency events. Then the expected cost will be

$$E(z) = p \cdot f(z) + q \cdot g(z).$$

The model of minimizing $E(z)$ would be an analogue of the median problem with positive and negative weights (see Burkard and Krarup 1998), and could be an interesting problem for further study. First results in this respect concerning paths, stars and trees can be found in the recent report by Burkard and Dollani (2001).

Acknowledgment. We thank Gerhard Woeginger for useful discussions.

References

[1] Aho, A. V., J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading (MA) 1983.

[2] Blum, M., R. W. Floyd, V. Pratt, R. L. Rivest, and R. E. Tarjan, Time bounds for selection. *J. Comput. Syst. Sci.* **7**, 1973, 448–461.

[3] Burkard, R. E. and H. Dollani, Center problems with pos/neg weights on trees. SFB-Report 215, Institute of Mathematics, Graz University of Technology, Graz (Austria), February 2001.

[4] Burkard, R. E. and J. Krarup, A linear algorithm for the pos/neg-weighted 1-median problem on a cactus. *Computing* **60**, 1998, 193–215.

[5] Drezner, Z. and G. O. Wesolowsky, Location of multiple obnoxious facilities. *Transportation Sci.* **19**, 1985, 193–202.

[6] Gonzalez, T., Algorithms on sets and related problems. Technical report, Department of Computer Science, University of Oklahoma, 1975.

[7] Hakimi, S. L., Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research* **12**, 1964, 450–459.

[8] Handler, G. Y., p-center problems. In *Discrete Location Theory* (ed. by P. B. Mirchandani and R. L. Francis), New York: J. Wiley, 1990, pp. 305–347.

[9] Hershberger, J. and S. Suri, Off-line maintenance of planar configurations. *J. Algorithms* **21**, 1996, 453–475.

[10] Kariv, O. and S. L. Hakimi, An algorithmic approach to network location problems I: The p-centers, II: The p-medians. *SIAM Journal on Applied Mathematics* **37**, 1979, 513–538 and 539–560.

[11] Megiddo, N., Linear-time algorithms for linear programming in R^3 and related problems. *SIAM J. Comput.* **12**, 1983, 759–776.

[12] Megiddo, N. and A. Tamir, New results on the complexity of p-center problems. *SIAM J. Comput.* **12**, 1983, 751–758.

[13] Melachrinoudis, E. and J. MacGregor Smith, An $O(mn^2)$ algorithm for the maximin problem in E^2. *Oper. Res. Letters* **18**, 1995, 25–30.

[14] Overmars, M. H. and J. van Leeuwen, Maintenance of configurations in the plane. *J. Comput. Syst. Sci.* **23**, 1981, 166–204.

[15] Preparata, F. and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, 1985.

[16] Tamir, A., Improved complexity bounds for center location problems on networks by using dynamic data structures. *SIAM J. Discrete Math.* **1**, 1988, 377–396.

[17] Tamir, A., Obnoxious facility location on graphs. *SIAM J. Discr. Math.* **4**, 1991, 550–567.