SIAM J. COMPUT.
Vol. 12, No. 4, November 1983

0097-5397/83/1204-0012 $01.25/0

LINEAR-TIME ALGORITHMS FOR LINEAR PROGRAMMING IN R^3 AND RELATED PROBLEMS*

NIMROD MEGIDDO†

Abstract. Linear-time algorithms for linear programming in R^2 and R^3 are presented. The methods used are applicable for other graphic and geometric problems as well as quadratic programming. For example, a linear-time algorithm is given for the classical problem of finding the smallest circle enclosing n given points in the plane; this disproves a conjecture by Shamos and Hoey [Proc. 16th IEEE Symposium on Foundations of Computer Science, 1975] that this problem requires $\Omega(n \log n)$ time. An immediate consequence of the main result is that the problem of linear separability is solvable in linear time. This corrects an error in Shamos and Hoey's paper, namely, that their $O(n \log n)$ algorithm for this problem in the plane was optimal. Also, a linear-time algorithm is given for the problem of finding the weighted center of a tree, and algorithms for other common location-theoretic problems are indicated. The results apply also to the problem of convex quadratic programming in three dimensions.

The results have already been extended to higher dimensions, and we know that *linear programming can be solved in linear time when the dimension is fixed.* This will be reported elsewhere; a preliminary version is available from the author.

Key words. linear programming, 1-center, weighted center, smallest circle, linear time, median, separability, quadratic programming

1. Introduction.

The problem of finding the convex hull of n points in the plane has been studied by many authors, and its complexity is known to be $O(n \log n)$ not only in the plane but also in R^3 (Graham [G], Preparata and Hong [PH] and Yao [Y]). Several known problems in computational geometry, such as farthest points, smallest circle, extreme point, etc., are closely related to the problem of finding the convex hull of n points in the plane (Shamos [Sh], Shamos and Hoey [ShH] and Dobkin and Reiss [DR]). We have not found in these references an explicit statement about the complexity of linear programming in two and three dimensions. A closely related problem is the "separability" problem for which a statement of complexity was made. The separability problem is to separate two sets of n points in R^d by means of a hyperplane. Dobkin and Reiss [DR] report that this problem is solvable in $O(n \log n)$ time when $d \leqq 3$, referring to Preparata and Hong's work [PH]. Moreover, Shamos and Hoey solve the separability problem in R^2 in $O(n \log n)$ time and claim (erroneously) [ShH, p. 224] their algorithm to be optimal. The truth is that the separability problem in R^d is obviously solvable by linear programming in d variables. In particular, it follows from the results of the present paper that it can be solved in $O(n)$ time when $d \leqq 3$.

We may learn about the state-of-art of the complexity of linear programming in R^2 by considering the "extreme point" problem, i.e., the problem of determining whether a given point P_0 in R^2 is a convex combination of n given points $P_1, \cdots, P_n$ in R^2. Dobkin and Reiss [DR, p. 17] state without proof or reference that this problem (in R^2) is solvable in linear time. This statement is rather obvious since the extreme point problem in the plane can be modeled as a problem of finding a straight line which crosses through P_0 and has all the points $P_1, \cdots, P_n$ lying on one side of it. The latter, however, amounts to linear programming in R^1 which is trivial. The same

* Received by the editors February 9, 1982, and in revised form November 15, 1982. This research was partially supported by the National Science Foundation under grants ECS-8121741 and ECS-8218181, at Northwestern University.

† Department of Statistics, Tel Aviv University, Tel Aviv, Israel. Currently visiting Department of Computer Science, Stanford University, Stanford, California 94305.

observation implies that the separability problem in R^d $(d \geqq 2)$ can be solved by linear programming in $d-1$ variables so that, in view of the present paper, it is solvable in linear time in R^4.

Another problem, related to linear programming in three variables, which we solve in $O(n)$ time, is that of finding the smallest circle enclosing n given points in the plane. Shamos and Hoey [ShH] solve this problem in $O(n \log n)$ time, improving the previously known bound of $O(n^3)$ very significantly. A seemingly related problem, namely, that of finding the largest empty circle, was shown to require $\Omega(n \log n)$ time, and that led Shamos and Hoey to the (wrong) conjecture that $\Omega(n \log n)$ was also a lower bound for the smallest enclosing circle problem. They were convinced that the so-called Voronoi diagram would always provide optimal algorithms, so they stated [ShH, p. 231]: "... the proper attack on a geometry problem is to construct those geometric entities that delineate the problem ...". Our results prove that this is not always the case, since the construction of the Voronoi diagram does require $\Omega(n \log n)$ time, while the smallest enclosing circle can be found in $O(n)$ time.

The problems discussed in this paper are presented in order of increasing difficulty. We start with linear programming in R^2 which is a subroutine for the three-dimensional problem. The two-dimensional case is discussed in § 2. In § 3 we consider the problem of the weighted center of a tree. The latter is more complicated than linear programming in two variables but yet does not involve the difficulties which arise in the three-dimensional case. The best known bound for it was $O(n \log n)$ [KH]. The problem of the smallest circle enclosing n points in the plane, which is discussed in § 4, is more complicated than linear programming in the plane. It is in fact a three-dimensional problem in a certain sense, and the algorithm which we present for it leads to the design of a linear-time algorithm for linear programming in R^3. In § 4 we also point out how our results apply to other location-theoretic problems in the plane. The problem of linear programming in three variables is discussed in § 5. Our linear programming algorithm for R^3 can easily be extended to solve convex quadratic programming problems in R^3 in $O(n)$ time. The latter is also discussed in § 5. In the Appendix we include an efficient algorithm for the extreme-point problem in the plane (discussed earlier in this Introduction) which is a routine for solving the smallest circle problem.

2. Linear programming in the plane.

2.1. Preliminaries. The linear programming problem in the plane can be stated as follows:

$$\begin{aligned} &\underset{x_1,x_2}{\text{minimize}} && c_1x_1+c_2x_2 \\ &\text{s.t.} && a_{i1}x_1+a_{i2}x_2 \geqq \beta_i \qquad (i=1,\cdots,n). \end{aligned}$$

It will be convenient for us to deal with the problem in an equivalent form, which can be obtained from the original one in $O(n)$ time:

$$\begin{aligned} &\underset{x,y}{\text{minimize}} && y \\ &\text{s.t.} && y \geqq a_ix+b_i \qquad (i \in I_1), \\ &&& y \leqq a_ix+b_i \qquad (i \in I_2), \\ &&& a \leqq x \leqq b \end{aligned}$$

where $|I_1|+|I_2| \leqq n$ and $-\infty \leqq a, b \leqq \infty$. We also define the following functions:

$$g(x) = \max\{a_i x + b_i : i \in I_1\},$$
$$h(x) = \min\{a_i x + b_i : i \in I_2\}.$$

Obviously, both of these functions are piecewise linear, and g is convex while h is concave. A number x, $a \leqq x \leqq b$, is said to be feasible if $g(x) \leqq h(x)$. We can pose our problem also in a one-dimensional form:

$$\begin{aligned} &\text{minimize} && g(x) \\ &\text{s.t.} && g(x) \leqq h(x), \\ & && a \leqq x \leqq b. \end{aligned}$$

Our algorithm works as follows. We test values of x in a fashion resembling binary search. Each test runs in linear time and enables us to drop at least a quarter of the constraints of the problem. We first have to describe our test in detail.

2.2. Testing a value of x. Given any value x' of x $(a \leqq x' \leqq b)$, we test the following: (i) Is x' feasible? (ii) If x' is not feasible then if there are any feasible values of x then they must lie on one side of x'; our test either determines that side or concludes that no feasible values exist; (iii) If x' is feasible then our test will recognize whether x' is also optimal and if not then it will tell us on what side of x' the minimum lies.

We start with the case of infeasible x'. In other words, $g(x') > h(x')$. Consider the function $f(x) = g(x) - h(x)$. This function is convex so the values of x such that $f(x) \leqq 0$ (if there are any) all lie on one side of x'. In order to tell the correct side we look at the one-sided derivatives of f at x'. This is done as follows (see Fig. 1). Define

$$s_g = \min\{a_i : i \in I_1,\ a_i x' + b_i = g(x')\},$$
$$S_g = \max\{a_i : i \in I_1,\ a_i x' + b_i = g(x')\},$$
$$s_h = \min\{a_i : i \in I_2,\ a_i x' + b_i = h(x')\},$$
$$S_h = \max\{a_i : i \in I_2,\ a_i x' + b_i = h(x')\}.$$

If $s_g > S_h$ then $f(x)$ is ascending at x' so that a feasible x can only be smaller than x'. Analogously, if $S_g < s_h$, then $f(x)$ is descending at x' so that a feasible x can only be larger than x'. The remaining case is when $s_g - S_h \leqq 0 \leqq S_g - s_h$. In this case f attains its minimum at x', i.e., there are no feasible values of x.

Consider now the case when x' is found to be feasible. We are interested in finding out on what side of x' the optimal solution lines. Assume, first, that $g(x') < h(x')$. Here the analysis is quite simple. We need to look only at the numbers s_g and S_g. If $s_g > 0$ then an optimal solution (denote it by x^*) must satisfy $x^* < x'$. Analogously, if $S_g < 0$ then $x^* < x'$. Otherwise, $s_g \leqq 0 \leqq S_g$ and x' itself is a minimum of g. If $g(x') = h(x')$ then the situation could be one of the following: (i) If $s_g > 0$ and $s_g \geqq S_h$ then $x^* < x'$. (ii) If $S_g < 0$ and $S_g \leqq s_h$ then $x^* > x'$. Otherwise x' itself is a minimum of $g(x)$ under the constraint $g(x) \leqq h(x)$.

In summary, if x' is any value in $[a, b]$ then in linear time we can either find that the problem is infeasible, recognize that x' itself is an optimal solution, or decide that the rest of the computation may be confined to one of the subintervals $[a, x']$, $[x', b]$.

2.3. The algorithm. We start the procedure by arranging the elements of I_1 in disjoint pairs and, similarly, those of I_2 in disjoint pairs (a single element from either

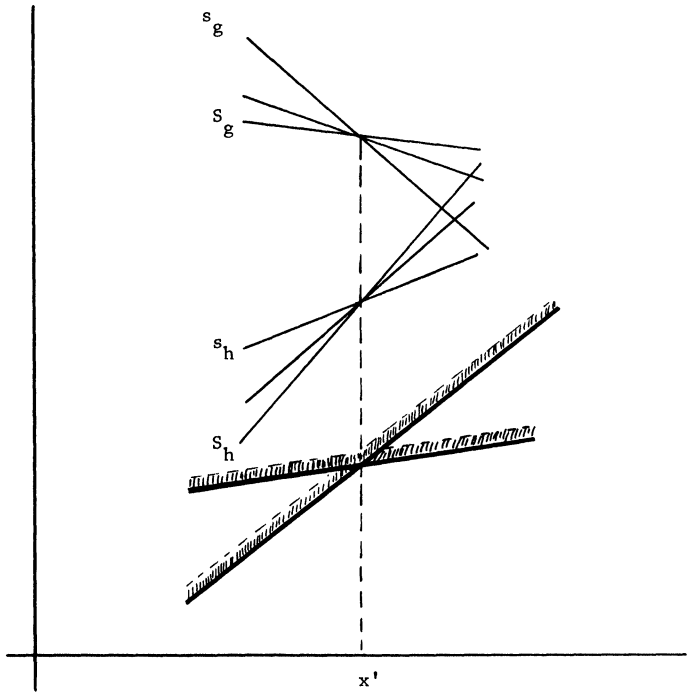

FIG. 1

set may be left unmatched). Consider, for example, a pair $i, j \in I_1$. If $a_i = a_j$ then, obviously, one of the constraints $y \geqq a_i x + b_i$, $y \geqq a_j x + b_j$ may be dropped without affecting the optimal solution of the problem. Otherwise, $a_i \neq a_j$ and the number $x_{ij} = (b_i - b_j)/(a_j - a_i)$ (i.e., the solution of the equation $a_i x + b_i = a_j x + b_j$) has the property that if x is confined to an interval which does not contain x_{ij} in its interior then, again, one of the two constraints is redundant and may be dropped. A similar observation is of course valid for a pair of constraints of the form $y \leqq a_i x + b_i$.

Consider all the pairs that have been formed. First, drop every constraint which is redundant according to the previous observation, i.e., either because of an inequality $a_i = a_j$ or because of a relation $x_{ij} \notin (a, b)$. We now consider only those pairs i, j (that have been formed) for which $a_i \neq a_j$ and $a < x_{ij} < b$. The next step is to find the median x_m of the set of x_{ij}'s. This can be done in linear time (see [AHU]). We then test the value x_m along the lines described in § 2.2, i.e., we either recognize that our problem is infeasible, find out that x_m is an optimal solution for our problem, or deduce that the interval $[a, b]$ may be redefined (the new interval being either $[a, x_m]$ or $[x_m, b]$). In the first two cases our task is finished. In the latter case we do the following. At least half of the critical values x_{ij} (as defined by the original pairs) will not be in the interior of the new interval. We will thus be able to drop one constraint per each such pair which is at least a quarter of the set of constraints (including those that have been dropped prior to the evaluation of x_m). We are thus left with a linear programming problem in the plane with at most $[3n/4]$ constraints. This implies that the runtime time (n) of our algorithm on an n-constraint problem satisfies time $(n) \leqq C \cdot n +$ time $(3n/4)$ and hence time $(n) = O(n)$. Of course, when n is small (e.g., $n \leqq 4$) the problem will be solved directly.

3. The weighted center of a tree.

3.1. Introduction. The weighted center of a tree is defined as follows. Given is a tree $T=(V, E)$ with n vertices. A nonnegative length d_{ij} is associated with every edge (i, j) and a nonnegative weight w_i is associated with every vertex i. An edge (i, j) is identified with a line segment of length d_{ij} so that we can talk about any "point" on the edge (i, j); formally, a point $x=(i, j; t)$ is characterized by being located at a distance of t from i and $d_{ij}-t$ from j. Thus the distance $d(x, y)$ between any two points x, y on the tree is well defined, namely, it is the length of the unique path from x to y. The weighted center of T is a point x which minimizes the function $r(x)=\max\{w_i d(x, i): i \in V\}$. The center is unique unless all the weights equal zero. A related problem is to find a vertex j which minimizes the function $r(x)$, i.e., x is restricted to be a vertex of the tree.

The best known algorithm for the weighted center problem is an $O(n \log n)$ procedure by Kariv and Hakimi [KH]. Other algorithms which run in $O(n^2)$ time have been given in Dearing and Francis [DF], Levin [L] and Hakimi, Schmeichel and Pierce [HSP].

The unweighted case, namely, when all the weights w_i are equal, is much easier and is solvable in $O(n)$ time (see Handler and Mirchandani [HM]). We will present here a linear-time algorithm for the general weighted case.

3.2. Preliminaries. The function $r(x)$ is convex on every simple path of the tree. Specifically, if P is a simple path and i is any vertex, then consider the vertex j which is on the path P and is nearest to i. The vertex j partitions the path into two pieces over each of which the function $g_i(x)=d(x, i)$ is linear and increasing as we move away from j. Thus, $g_i(x)$ is piecewise linear on P (with at most two pieces) and convex. The function $r(x)$ is hence piecewise linear and convex, being the maximum of convex functions.

Let x be any point on the tree. A vertex j $(j \neq x)$ is said to be adjacent to x if x lies on an edge which is incident upon j (x may itself be a vertex but then it is not considered adjacent to itself). Let $V_j(x)$ denote the set of vertices i such that j lies on the simple path from x to i (see Fig. 2). Let $T_j(x)$ denote the subtree which is spanned by the set $V_j(x) \cup \{x\}$; in particular, this subtree contains an edge (j, x) which is just a subsegment of (j, k) for some $k \in V$. Consider the function $r_j(x)=\max\{w_i d(x, i): i \in V_j(x)\}$. Clearly, $r_j(x)$ decreases as we move from x in the direction of j. Let $j_1, \cdots, j_l$ be all the vertices adjacent to x ($l=2$ if x is interior to some edge). Obviously, $r(x)=\max\{r_{j_1}(x), \cdots, r_{j_l}(x)\}$. Moreover, if the maximum is attained at more than one index then x is a local minimum of the function r, and hence it must be the center since r is convex. On the other hand, if the maximum is attained at a unique index, then the center must lie in the corresponding subtree. Formally, if

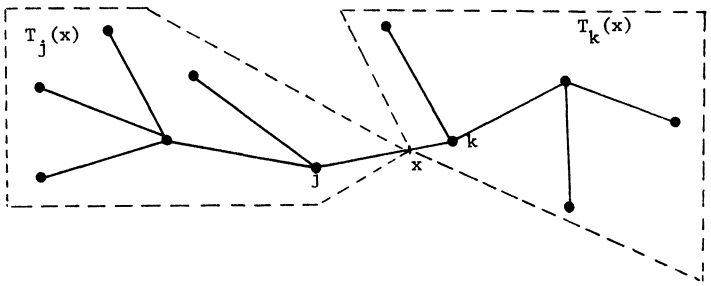

FIG. 2

$r_j(x) > r_k(x)$ for every vertex k $(k \neq j)$ adjacent to x then the center lies in $T_j(x)$. In summary, given any point x, we can easily tell (in $O(n)$ time) in which of the subtrees $T_j(x)$ the center lies.

Kariv and Hakimi [KH] based their procedure on the above observations. Given that the center lies in a subtree T' they "test" the centroid x of T' to find out which of the subtrees of T', rooted at x, contains the center. Since the centroid defines subtrees whose sizes are at most half the size of T', the process terminates within $O(\log n)$ tests and hence runs in $O(n \log n)$ time. The improvement we suggest here is in reducing the cost of a test. We will also perform $O(\log n)$ tests; however, the cost of each test will be no more than three quarters the cost of the preceding one.

3.3. The linear-time algorithm. Let c denote the centroid of T, i.e., c is a vertex such that for every adjacent vertex j, $|V_j(c)| \leqq n/2$ (where n is the number of vertices of T). We note that c can be found in $O(n)$ time [HM]. This is accomplished by a walk over the vertices, always moving in the direction in which the number of vertices, in the subtree entered into, is being maximized. Now assume that c is known.

First, evaluate $r_j(c)$ for each adjacent vertex j. This amounts to finding all the distances $d(c, i)$ and hence can also be carried out in $O(n)$ time. If there are two vertices j_1, j_2 $(j_1 \neq j_2)$ adjacent to c, such that $r_{j_1}(c) = r_{j_2}(c) = r(c)$, then c itself is the center and we terminate. Thus, let us now assume that j is adjacent to c and $r_j(c) > r_k(c)$ for every other adjacent vertex k. We now know that the center lies in $T_j(c)$.

If u is a vertex not in $V_j(c)$ and if x is in $T_j(c)$ at a distance of t from c, then $d(u, x) = d(u, c) + t$. If u and v are vertices not in $V_j(c)$ then by solving (for t) the equation $w_u(d(u, c) + t) = w_v(d(v, c) + t)$ we can tell the following: Assume, without loss of generality, that $w_u d(u, c) \geqq w_v d(v, c)$. There is a value t_{uv} $(0 \leqq t_{uv} \leqq \infty)$ such that, for every x in $T_j(c)$ at a distance of t from c, $w_u d(u, x) \geqq w_v d(v, x)$ if and only if $0 \leqq t \leqq t_{uv}$. Thus, if we knew that the center lay at a distance smaller than t_{uv} from c then we could disregard the vertex v from that point and on in the process of finding the center. Similarly, the vertex u could be eliminated if we knew that the center lay at a distance greater than t_{uv} from c. We will show below how to efficiently exploit this observation. However, we first need to show how to recognize whether or not the center lies within a distance of t from x, where x is any leaf vertex and t is any positive real number; our discussion applies to the tree $T_j(c)$ where $x = c$ is a leaf and t is some value of the type t_{uv}, derived from data which are external relative to the tree $T_j(c)$.

Given a leaf x and a positive real number t, we can (in linear time) find all the points $y_1, \cdots, y_l$ such that $d(x, y_\nu) = t$, $\nu = 1, \cdots, l$. This is done as follows. First, evaluate all the distances $d(x, i)$. Now note that every edge (i, j), such that $d(x, i) \leqq t \leqq d(x, j)$, contains a unique point y_ν at a distance of $t - d(x, i)$ from i, and hence $d(x, y_\nu) = t$. The set of all points z such that $d(x, z) \geqq t$ can be represented as a union of subtrees rooted at the points $y_1, \cdots, y_l$ (each y_ν may contribute several such subtrees). Let these subtrees be simply denoted by $T_1, \cdots, T_m$ and let their roots be denoted $u_1, \cdots, u_m$ $(\{u_1, \cdots, u_m\} \subset \{y_1, \cdots, y_l\})$. Let V_i denote the set of vertices of T_i except for u_i (see Fig. 3). Define $R_i(x) = \max\{w_k d(x, k): k \in V_i\}$. Since the sets V_i are pairwise disjoint, it follows that we can evaluate all the quantities $R_i(u_i)$, $i = 1, \cdots, m$, in $O(n)$ time. Let $R = \max\{R_i(u_i): i = 1, \cdots, m\}$. First, if $R_i(u_i) < R$ then the center is certainly not in T_i. Similarly, if $R_{i_1}(u_{i_1}) = R_{i_2}(u_{i_2}) = R$ for some $i_1 \neq i_2$, then the center cannot lie inside any T_i, $i = 1, \cdots, m$. The remaining case is when there is a unique i $(1 \leqq i \leqq m)$ such that $R_i(u_i) = R$. In this case the center may lie in T_i. However, this can be recognized by evaluating the functions $r_j(y)$, where j is any

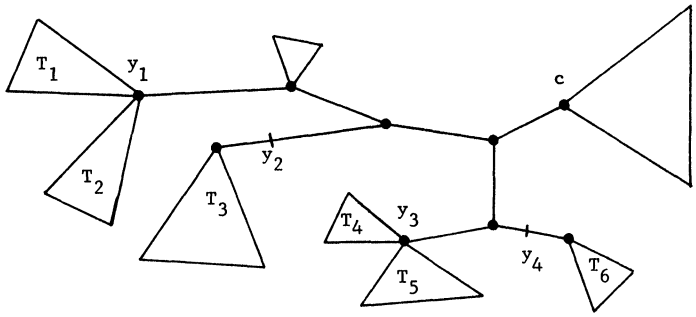

FIG. 3

vertex adjacent to u_i and $y = u_i$. All this again requires only $O(n)$ time. In summary, we can decide whether or not the center is within a distance of t from x (equivalently, whether or not it lies in one of the T_i's) in $O(n)$ time.

Returning to the original tree T, its centroid c and the subtree $T_j(c)$ which is known to contain the center, we now do the following. Arrange the vertices outside $T_j(c)$ in disjoint pairs $(u_1, v_1), (u_2, v_2), \cdots, (u_s, v_s)$ (leaving one out if there is an odd number of them). Note that there will be at least $[n/4]-1$ such pairs since at most $n/2$ vertices are in $T_j(c)$. For every such pair (u, v) consider the equation $w_u(d(u, c)+t) = w_v(d(v, c)+t)$, assuming, without loss of generality, that $w_u d(u, c) \geqq w_v d(v, c)$. If $w_u \geqq w_v$ then the vertex v is "discarded"; otherwise, let $t_{uv} = (w_u d(u, c) - w_v d(v, c))/(w_v - w_u)$.

Having calculated the values $t_{u_i v_i}$ (for all pairs from which no vertex was discarded), we now find the median of these values. This is done in $O(n)$ time (see [AHU]). Let the median be denoted by t_m. We now check whether the center lies (in $T_j(c)$) within a distance of t_m from c. This can be carried out in $O(n)$ time as we have already seen. Suppose, for example, we find that the center indeed lies within a distance of t_m from c. Consider a pair (u, v) such that $t_{uv} \geqq t_m$. It follows that wherever the center x^* lies (provided it is in $T_j(c)$ at a distance of no more than t_m from c) it must be true that $w_u d(u, x^*) \geqq w_v d(v, x^*)$. Thus, the vertex v is "dominated" by u in the sense that if the maximum weighted distance from the center is determined by v then it is also determined by u. Hence, we can safely discard the vertex v in this case. Similarly, if x^* is known to be at a distance greater than t_m from c, then from pairs (u, v), such that $t_{uv} \leqq t_m$, we can discard the vertex u.

It follows that one vertex is discarded from approximately half the pairs. In other words, we will discard approximately $\frac{1}{8}$ of the vertices of the tree. At this point we have reduced our problem to the weighted center problem on a tree T' which is defined as follows. For each vertex u not in $T_j(c)$, which has not been discarded, form an edge (c, u) and let its length be precisely $d(c, u)$. Also, let w_u be the same as in T. Adjoin all these edges to the tree $t_j(c)$ and call the new tree T'. Since $n/8$ vertices have been discarded, it follows that the run time, time(n), for a tree of n vertices, satisfies time $(n) \leqq$ time $(7n/8) + Cn$, which implies time $(n) = O(n)$.

The discrete problem of finding a vertex which minimizes the function $r(x)$ can now be solved. It follows from the convexity of the function $r(x)$ that the vertex minimizing $r(x)$ is either identical with or adjacent to the point at which $r(x)$ has its global minimum. Thus, by finding this global minimum we obtain at most two vertices (endpoints of an edge), one of which is minimizing $r(x)$ relative to the set of vertices.

4. Smallest circle enclosing n points.

4.1. Introduction. In the present section we shall deal with the classic problem of finding the smallest circle enclosing n given points (a_i, b_i), $i = 1, \cdots, n$, in the Euclidean plane. In the language of location theory this is the (unweighted) Euclidean 1-center problem in the plane. Formally, we are looking for a point (x, y) so as to minimize $\text{Max}\{((x-a_i)^2+(y-b_i)^2)^{1/2}: 1 \leqq i \leqq n\}$. Thus, the point (x, y) is an optimal location for a facility if we wish to minimize the largest distance that a customer would have to travel from his residence (in one of the given points (a_i, b_i)) to the facility.

The smallest enclosing circle problem has a long history. It was posed by Sylvester [Sy1] in 1857 and different solutions have been suggested in Sylvester [Sy2], Rademacher and Toeplitz [RT], Courant and Robbins [CR], Francis [F], Smallwood [Sm], Francis and White [FW], Nair and Chandrasekaran [NC], Elzinga and Hearn [EH], and finally, Shamos and Hoey [ShH]. Shamos and Hoey's algorithm runs in $O(n \log n)$ time and is the only one which has been proved to run in $o(n^3)$ time. It is based on constructing the so-called "farthest point Voronoi diagram" which we review below. This powerful structure is very useful for solving a number of computational geometric problems and its construction requires $\Omega(n \log n)$ time. This led Shamos and Hoey to the (wrong) conjecture that the smallest enclosing circle problem also had a lower bound of $\Omega(n \log n)$ [ShH, p. 154]. We shall present here a linear-time algorithm for this problem.

The diagram is a partition of the plane into regions V_j where a point (x, y) is in V_j if and only if the point (a_j, b_j) is farthest from (x, y) among the points (a_i, b_i), $i = 1, \cdots, n$. These regions are either empty or unbounded polyhedral sets. The construction of the diagram also yields the vertices of the polytope $\pi = \text{convex hull}\{(a_1, b_1), \cdots, (a_n, b_n)\}$ in their cyclic ordering on the boundary of π. Once the boundary is known, it takes $O(n)$ time to find the two farthest points. These two points define a circle whose diameter equals the distance between them. If the entire π is contained in this circle then this is the smallest possible circle. Otherwise, the smallest enclosing circle is centered at a point where some three regions V_i, V_j, V_k meet, i.e., the circle is defined by the points (a_i, b_i), (a_j, b_j), (a_k, b_k). It can be shown that there are at most $n-2$ such points in the diagram (relying on the fact that, as a graph, the farthest-point Voronoi diagram has no circuits) and the distances from such points to their respective defining points (a_i, b_i) can be produced during the construction of the diagram. It thus takes $O(n)$ time to find the center of the smallest enclosing circle once the diagram has been constructed.

4.2. A constrained version of the smallest circle problem. We will first develop an algorithm for a constrained problem, namely, where the center of the enclosing circle is forced to lie on a given straight line. For simplicity of presentation assume this line is the x-axis. Furthermore, at the end of the computation in this constrained problem we will be able to tell on which side of the straight line the unconstrained center lies. This will play an important role in the solution of the unconstrained problem.

Consider the problem of minimizing $g(x) = \max\{(x-a_i)^2+b_i^2: 1 \leqq i \leqq n\}$. Let i, j be any two distinct indices and consider the equation $(x-a_i)^2+b_i^2 = (x-a_j)^2+b_j^2$. This is in fact a linear equation: $-2a_ix+a_i^2+b_i^2 = -2a_jx+a_j^2+b_j^2$. If $a_i = a_j$ then (assuming, for example, $b_i^2 \leqq b_j^2$) we may drop the function $(x-a_i)^2+b_i^2$ from the definition of g. If $a_i \neq a_j$ then there is a critical value $x_{ij} = (a_j^2-a_i^2+b_j^2-b_i^2)/2(a_j-a_i)$ such that (assuming $a_j > a_i$) $(x-a_i)^2+b_i^2 \geqq (x-a_j)^2+b_j^2$ if and only if $x \geqq x_{ij}$.

The algorithm for the constrained center problem works as follows: Consider the pairs $(1, 2)$, $(3, 4)$, $\cdots$. For each pair $(i, i+1)$ (i odd), such that $a_i = a_{i+1}$, drop one of

the functions as explained above. Compute the critical values $x_{i,i+1}$. Find the median x_m of the values $x_{i,i+1}$ by any linear-time median-finding algorithm (see [AHU]). Let x^* denote the minimizer of g. Compute $g(x_m)$. We shall now discuss the question of recognizing whether $x_m < x^*$, $x_m = x^*$ or $x_m > x^*$. Let $I = \{i: (x_m - a_i)^2 + b_i^2 = g(x_m)\}$. If $x_m < a_i$ for every $i \in I$ then $x_m < x^*$. If $x_m > a_i$ for every $i \in I$ then $x_m > x^*$; otherwise, $x_m = x^*$. Knowing either that $x^* < x_m$ or that $x^* > x_m$, we can discard a quarter of our functions as follows. Assume for example $x^* < x_m$. We have at least half of our critical values $x_{i,i+1}$, greater than x^*. If $x_{i,i+1} > x_m$ then, since $(x - a_i)^2 + b_i^2 \geqq (x - a_{i+1})^2 + b_{i+1}^2$ if and only if $x \geqq x_{i,i+1}$, we may discard the function $(x - a_i)^2 + b_i^2$. This is because $(x^* - a_i)^2 + b_i^2 \leqq (x^* - a_{i+1})^2 + b_{i+1}^2$. Thus for at least half the pairs we can discard one function per pair. This implies that at least one quarter of the functions are dropped at the end of this stage. The linearity of the run time follows.

We now address the question of recognizing on which side of the straight line the unconstrained center lies. First, note that the function $f(x, y) = \max\{(x - a_i)^2 + (y - b_i)^2 : i \leqq i \leqq n\}$ is convex. It is essential to note that f is convex, not only in each variable, but also as a function of two variables. This also implies that the function $h(y) = \min_x f(x, y)$ is convex. By minimizing $g(x)$ we in fact evaluate $h(0)$. The y-coordinate of the unconstrained center is precisely where $h(y)$ attains its minimum. Denote this value by y^c. Since $h(y)$ is convex we can tell the sign of y^c simply by looking in the neighborhood of $y = 0$. Thus, let $(x^*, 0)$ be the constrained center we have found. Let $I = \{i: (x^* - a_i)^2 + b_i^2 = g(x^*)\}$. Obviously, if $I = \{i\}$ then $x^* = a_i$ and y^c has the sign of b_i. If $I = \{i, j\}$ then $(x^*, 0)$ lies on the perpendicular bisector of the line segment $[(a_i, b_i), (a_j, b_j)]$. Obviously, y^c has the sign of the y-coordinate of the midpoint of this segment, i.e., $\frac{1}{2}(b_i + b_j)$. In general, all the points (a_i, b_i), $i \in I$ lie on a circle centered at $(x^*, 0)$. If $(x^*, 0)$ is in the convex hull of these points then $y^c = 0$. Otherwise, there exist two points (a_i, b_i), (a_j, b_j), $(i, j \in I)$ such that $f(x, y)$ decreases as we move from $(x^*, 0)$ in the direction of the midpoint of the line segment $[(a_i, b_i), (a_j, b_j)]$ (i.e., along the perpendicular bisector of that segment; see Fig. 4). In this case y^c has the sign of $\frac{1}{2}(b_i + b_j)$. It should be noticed that the determination of these two points or the recognition that $(x^*, 0)$ is in the convex hull can be carried out in linear time with the aid of linear programming in the plane (see § 2); a more straightforward method is given in the Appendix.

In summary, given any straight line in the plane, we can in $O(n)$ time determine on which side of the line the center of the smallest enclosing circle lies. Moreover, if

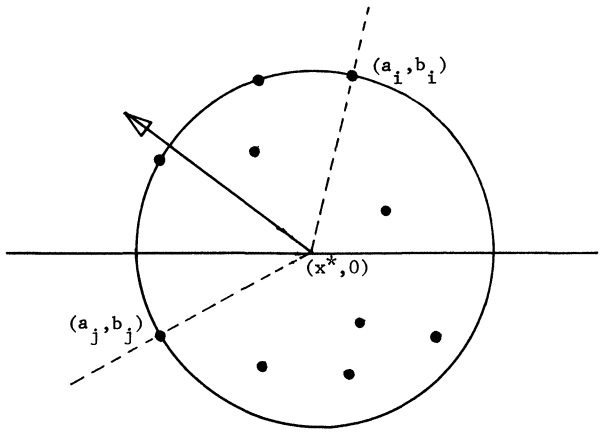

FIG. 4

this center happens to lie on the line then we discover its exact location during the procedure.

4.3. The $O(n)$ algorithm for the smallest circle. We shall now utilize the result of the preceding section for finding the center of the unconstrained problem. We start by producing the perpendicular bisectors of the line segments $[(a_{2i-1}, b_{2i-1}), (a_{2i}, b_{2i})]$, $i = 1, \cdots, [n/2]$. Denote them by L_i. Consider the angles α $(-\pi/2 \leqq \alpha < \pi/2)$ which these lines form with the positive direction of the x-axis. Let α_m denote the median of these angles. Consider the linear transformation that takes the x-axis to the line $y = \alpha_m x$ and leaves the y-axis fixed. By applying this transformation we can have at least half of our lines with nonnegative angles and at least half with nonpositive angles. It is obvious that this can be achieved in linear time.

The next step is to form pairs of lines (L_i, L_j) so that each pair has one line with nonnegative angle and one line with nonpositive angle; the pairs are disjoint. Thus, there will be $[n/4]$ such pairs. For each pair (L_i, L_j) define a value y_{ij} as follows. If L_i and L_j are parallel to the x-axis then let y_{ij} be the mean of their constant y-coordinates. Otherwise, they must intersect; let (x_{ij}, y_{ij}) denote their point of intersection. Now, let y_m denote the median of the $[n/4]$ values y_{ij}. The value y_m can be found in linear time. At this point we test on what side of the straight line $y = y_m$ the center must lie. This test runs in $O(n)$ time as we have shown in § 4.2. If the center lies on the line $y = y_m$ then we are done. Thus, suppose it does not lie on the line and assume, without loss of generality, that it lies underneath this line. Consider a pair L_i, L_j of parallel lines such that $y_{ij} \geqq y_m$. At least one of these lines lies above the line $y = y_m$. Suppose it is the line L_i (which is the perpendicular bisector of the line segment $[(a_{2i-1}, b_{2i-1}), (a_{2i}, b_{2i})]$). We can now drop one of the two defining points, namely the one which lies underneath the line L_i, since the other point is farther from the center. However, in general the lines L_i, L_j are not expected to be parallel.

Consider now the set of all pairs (L_i, L_j) of nonparallel lines for which $y_{ij} \geqq y_m$. We find the median x_m of the x_{ij}'s corresponding to such pairs. Like in the case of the y-coordinate, we now test on what side of the line $x = x_m$ the center of the smallest enclosing circle must lie. Suppose, for example, it lies to the left of this line. Consider pairs (i, j) such that $x_{ij} \geqq x_m$ and $y_{ij} \geqq y_m$. One of the lines, say L_i, forms a nonpositive angle with the positive direction of the x-axis. It follows (see Fig. 5) that one of the

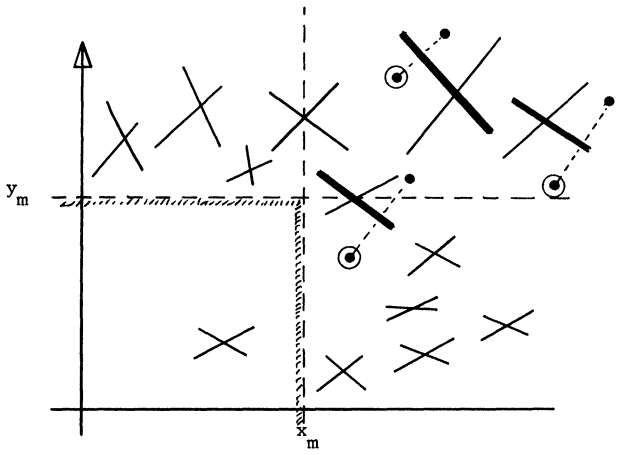

FIG. 5

points defining L_i, namely, the one which lies "southwest" of it, can be dropped since the other point will be at least as far from the center.

It follows that during this process we drop one point per pair for at least a quarter of our pairs of lines. In other words, at least $[n/16]$ points will be dropped with an $O(n)$ effort. It thus follows that the entire process runs in linear time.

4.4. A remark on other planar location problems. An analogous problem is the rectilinear 1-center problem in the plane for which linear-time algorithms are known [FW]. However, the weighted rectilinear problem, i.e.,

$$\underset{x,y}{\text{minimize}} \max \{w_i(|x-a_i|+|y-b_i|): i=1,\cdots,n\}$$

(where (a_i, b_i) are given points and w_i are given positive weights) can now be solved in $O(n)$ time by our methods in the present paper. The previously known bound was $O(n \log n)$ and followed from separating the planar problem into two one-dimensional problems. The one-dimensional problem is a special case of the weighted center problem of a tree, provided the numbers are sorted.

A much more complicated problem is the weighted Euclidean 1-center problem. The best known bound for this problem used to be $O(n^3)$ [EH], [DW], [CP]. In a recent paper the author [M2] presented an algorithm which required $O(n(\log n)^3(\log \log n)^2)$ time. The methods presented in the present paper combined with those of [M1], [M2] can yield an $O(n(\log n)^2)$ algorithm for the weighted Euclidean 1-center problem. The details will be given elsewhere.

5. Linear programming in R^3.

5.1. Preliminaries. In this section we will be dealing with the following problem:

$$\begin{aligned} &\underset{x_1,x_2,x_3}{\text{minimize}} && \gamma_1 x_1+\gamma_2 x_2+\gamma_3 x_3 \\ &\text{s.t.} && a_{i1}x_1+a_{i2}x_2+a_{i3}x_3 \geqq \beta_i \qquad (i=1,\cdots,n). \end{aligned}$$

We first transform the problem into the following form (in $O(n)$ time):

$$\begin{aligned} &\underset{x,y,z}{\text{minimize}} && z \\ &\text{s.t.} && z \geqq a_i x+b_i y+c_i \quad (i\in I_1), \\ &&& z \leqq a_i x+b_i y+c_i \quad (i\in I_2), \\ &&& a_i x+b_i y+c_i \leqq 0 \quad (i\in I_3) \end{aligned}$$

where $|I_1|+|I_2|+|I_3|=n$. We define the following functions:

$$\begin{aligned} g(x,y) &= \max\{a_i x+b_i y+c_i: i\in I_1\}, \\ h(x,y) &= \min\{a_i x+b_i y+c_i: i\in I_2\}, \\ e(x,y) &= \max\{a_i x+b_i y+c_i: i\in I_3\} \end{aligned}$$

which are all piecewise linear, g and e being convex while h is concave. Let also $f(x,y)=\max\{g(x,y)-h(x,y), e(x,y)\}$. Note that our problem is equivalent to the following:

$$\begin{aligned} &\underset{x,y}{\text{minimize}} && g(x,y) \\ &\text{s.t.} && f(x,y)\leqq 0, \end{aligned}$$

where f is convex. We call a point (x, y) feasible if $f(x,y)\leqq 0$.

The algorithm in R^3 generalizes that of the plane along the following lines. We will first develop a routine which tests straight lines in the plane. The goal of the test is to select one of the half planes determined by the line, such that the rest of the computation may be confined to that half plane. Then a procedure is developed during which applications of the test enable us to drop relatively large families of inequalities, so that the cost of the tests which follow becomes smaller and smaller.

5.2. Testing a line. Given any straight line in the x, y-plane, we wish to recognize which of the two half planes, determined by the line, is relevant for our problem. For simplicity, assume that the line coincides with the x-axis (otherwise transform the coordinate system accordingly). So, the goal of the test is to find whether we should look for a point (x, y) with positive y or negative y. The conclusion of the test may be one of the following: (i) The problem is infeasible. (ii) There is a global minimum with $y = 0$. (iii) The problem is unbounded. (iv) The rest of the computation should be in the half plane $\{(x, y): y > 0\}$. (v) The rest of the computation should be in the half plane $\{(x, y): y < 0\}$.

We may reach conclusions (iv) or (v) in one of the following circumstances. We may find that there are no feasible solutions on the x-axis and realize that if there are any feasible points then they all must lie in the half plane we have found. On the other hand, we may find a feasible point on the x-axis but realize that, in order to decrease the value of the function g, we must proceed into the half plane we have identified.

The test amounts to finding the minimum of $g(x, 0)$ subject to $f(x, 0) \leqq 0$ and then analyzing the picture in the neighborhood of the solution of this optimization problem. Our planar linear programming algorithm in § 2 is capable of reaching one of the following results: (i) It may find out that the problem is unbounded (even when restricted to the x-axis). (ii) It may find out that the problem (on the x-axis) is infeasible, in which case it will produce a point $(x^*, 0)$ which minimizes $f(x, 0)$ (in this case $f(x, 0) > 0$ for every x). (iii) It may find a point $(x^*, 0)$ which minimizes $g(x, 0)$ subject to the constraint $f(x, 0) \leqq 0$. If the problem is unbounded on the x-axis then, of course, we are done. So, assume a point $(x^*, 0)$ has been produced. Without loss of generality assume $x^* = 0$ (otherwise, translate the x-coordinate accordingly). The test relies only on the inequalities which are tight at $(0, 0)$. Formally, we define subsets $I_j^* \subset I_j$ $(j = 1, 2, 3)$ as follows. An index $i \in I_1$ belongs to I_1^* if $c_i = g(0, 0)$ (i.e., $c_i = \max\{c_j: j \in I_1\}$). An index $i \in I_2$ belongs to I_2^* if $c_i = h(0, 0)$ provided $f(0, 0) = g(0, 0) - h(0, 0) \geqq 0$. Note that $I_2^* = \varnothing$ if either $f(0, 0) < 0$ or $f(0, 0) > g(0, 0) - h(0, 0)$. Finally, an index $i \in I_3$ belongs to I_3^* if $c_i = e(0, 0)$ provided $f(0, 0) = e(0, 0) \geqq 0$.

Distinguish two cases according to the circumstances of producing the point $(x^*, 0)$ (which is now assumed to be equal to $(0, 0)$):

Case 1. $f(0, 0) \leqq 0$. In this case we have found a feasible point and are interested in decreasing the value of the function g (subject to $f(x, y) \leqq 0$). The test is based on the following:

PROPOSITION 1. *The existence of a point* (x, y) *such that* $y > 0$, $g(x, y) < g(0, 0)$ *and* $f(x, y) \leqq 0$ *is equivalent to the existence of a* λ *such that*

$$\text{(i)} \qquad \max\{a_i\lambda + b_i: i \in I_1^*\} < 0,$$

$$\text{(ii)} \qquad \max\{a_i\lambda + b_i: i \in I_1^*\} \leqq \min\{a_i\lambda + b_i: i \in I_2^*\}, \quad \textit{and}$$

$$\text{(iii)} \qquad \max\{a_i\lambda + b_i: i \in I_3^*\} \leqq 0.$$

Proof. Suppose there is such a point (x, y). Let $\lambda = x/y$. It follows that

$$\max\{a_i\lambda y + b_i y + c_i : i \in I_1^*\} \leqq g(\lambda y, y) < g(0,0) = \max\{c_i : i \in I_1^*\}$$

so that (i) holds. If $I_2^* = \varnothing$ then (ii) is trivial. Thus, assume $I_2^* \neq 0$. This implies $f(0,0) = g(0,0) - h(0,0) = 0$. Here for every $i \in I_1^*$ and $j \in I_2^*$, $c_i = c_j$. Since

$$\begin{aligned}\max\{a_i\lambda y + b_i y + c_i : i \in I_1^*\} &\leqq g(\lambda y, y) \leqq h(\lambda y, y)\\ &\leqq \min\{a_i\lambda y + b_i y + c_i : i \in I_2^*\},\end{aligned}$$

it follows that (ii) holds. The validity of (iii) is proved analogously. Conversely, suppose that there is λ which satisfies (i), (ii), and (iii). If $y > 0$ is sufficiently small, then obviously $g(\lambda y, y) < g(0,0)$ and $f(\lambda y, y) \leqq 0$. This completes the proof.

PROPOSITION 2. *The existence of a point* (x, y) *such that* $y < 0$, $g(x, y) < g(0, 0)$ *and* $f(x, y) \leqq 0$ *is equivalent to the existence of a number* λ *such that*

(i) $$\min\{a_i\lambda + b_i : i \in I_1^*\} > 0,$$

(ii) $$\min\{a_i\lambda + b_i : i \in I_1^*\} \geqq \max\{a_i\lambda + b_i : i \in I_2^*\}, \quad \textit{and}$$

(iii) $$\min\{a_i\lambda + b_i : i \in I_3^*\} \geqq 0.$$

The proof is analogous to that of Proposition 1.

We can now describe the rest of the test in the case where $f(0,0) \leqq 0$. Given the sets I_j^* $(j = 1, 2, 3)$, we solve the following problem:

$$\begin{aligned}\underset{\lambda,\eta}{\text{minimize}} \quad & \eta\\ \text{s.t.} \quad & \eta \geqq a_i\lambda + b_i \qquad (i \in I_1^*),\\ & \eta \leqq a_i\lambda + b_i \qquad (i \in I_2^*),\\ & a_i\lambda + b_i \leqq 0 \qquad (i \in I_3^*).\end{aligned}$$

If a negative η is obtained then the half plane $\{(x, y) : y > 0\}$ is the proper one. Otherwise, we need to solve the following problem:

$$\begin{aligned}\underset{\lambda,\eta}{\text{minimize}} \quad & \eta\\ \text{s.t.} \quad & \eta \leqq a_i\lambda + b_i \qquad (i \in I_1^*),\\ & \eta \geqq a_i\lambda + b_i \qquad (i \in I_2^*),\\ & a_i\lambda + b_i \geqq 0 \qquad (i \in I_3^*).\end{aligned}$$

If a positive η is obtained then the half plane $\{(x, y) : y < 0\}$ is the proper one. In the remaining case the constraint $y = 0$ does not affect the global minimum and hence the point $(0, 0)$ (i.e., $(x^*, 0)$) is an optimal solution.

Case 2. $f(0,0) > 0$. We are then interested in decreasing the value of f by entering one of the half planes. The conclusions in this case are based on the following propositions whose proofs are similar to that of Proposition 1.

PROPOSITION 3. *The existence of a point* (x, y) *such that* $y > 0$ *and* $f(x, y) < f(0, 0)$ *is equivalent to the existence of a number* λ *such that*

(i) $$\max\{a_i\lambda + b_i : i \in I_1^*\} < \min\{a_i\lambda + b_i : i \in I_2^*\} \quad \textit{and}$$

(ii) $$\max\{a_i\lambda + b_i : i \in I_3^*\} < 0.$$

PROPOSITION 4. *The existence of a point* (x, y) *such that* $y<0$ *and* $f(x, y)<f(0, 0)$ *is equivalent to the existence of a number* λ *such that*

(i) $$\min\{a_i\lambda+b_i: i\in I_1^*\}>\max\{a_i\lambda+b_i: i\in I_2^*\} \quad \textit{and}$$

(ii) $$\min\{a_i\lambda+b_i: i\in I_3^*\}>0.$$

Thus, in the case where $f(0, 0)>0$ the test proceeds as follows. Consider the function

$$\varphi(\lambda)=\max(\max\{a_i\lambda+b_i: i\in I_1^*\}-\min\{a_i\lambda+b_i: i\in I_2^*\}, \max\{a_i\lambda+b_i: i\in I_3^*\}).$$

This is a convex piecewise linear function, and our methods in § 2 are applicable for finding its minimum in $O(n)$ time. In minimizing $\varphi(\lambda)$ we form pairs (i, j) of indices only when i and j belong to the same set I_k^* $(1\leqq k\leqq 3)$. If φ attains a negative value then the half plane $\{(x, y): y>0\}$ is the correct domain wherein to look for feasible points (see Proposition 3). Otherwise, we need to consider the function

$$\psi(\lambda)=\min(\min\{a_i\lambda+b_i: i\in I_1^*\}-\max\{a_i\lambda+b_i: i\in I_2^*\}, \min\{a_i\lambda+b_i: i\in I_3^*\}).$$

Analogously, if ψ attains a positive value then the half plane $\{(x, y): y<0\}$ is the correct one. In the remaining case f attains its global minimum on the x-axis, and that implies that our original problem is infeasible.

This completes the statement of our test of a given straight line.

5.3. The algorithm. The algorithm is based on the following principle. Consider two inequalities of the form $z\geqq a_ix+b_iy+c_i$, $z\geqq a_jx+b_jy+c_j$, i.e., $i, j\in I_1$. If $(a_i, b_i)=(a_j, b_j)$ then one of these constraints is redundant. Otherwise, let $L_{ij}=\{(x, y): a_ix+b_iy+c_i=a_jx+b_jy+c_j\}$. If $(a_i, b_i)\neq(a_j, b_j)$ then L_{ij} is a straight line which divides the plane into two halves. If we know that an optimal solution to our problem (if there is any) must lie in a certain half plane determined by L_{ij} then we may discard one of the two inequalities. A similar observation is true for pairs of inequalities $z\leqq a_ix+b_iy+c_i$, $z\leqq a_jx+b_jy+c_j$, i.e., when $i, j\in I_2$ as well as for pairs i, j such that $i, j\in I_3$.

We start the procedure by arranging the inequalities (except for at most three of them) in disjoint pairs so that the two members of each pair belong to the same set I_k $(1\leqq k\leqq 3)$. For each pair, either we can drop one of the participating inequalities right away, or we have a dividing line L_{ij}. Consider the set of lines that are generated in this way. At this point our procedure is essentially the same as in the problem of the smallest circle enclosing n points (see § 4.3). We review the basic ideas here in short. Given a set of straight lines, we will in $O(n)$ time find a subset of at least a quarter of the lines, such that for each line in that subset, it is known which of the two corresponding half planes may contain the solution. This is done as follows: First, we transform the coordinate system so that half the lines will have nonnegative slope and half the lines will have nonpositive slope. We then form pairs of lines where in every pair we will have one line with nonnegative slope and one line with nonpositive slope. Let the lines be redenoted $L_1, \cdots, L_k$. If L_i and L_j are members of one of our pairs then let (x_{ij}, y_{ij}) denote their point of intersection if there is a unique point. Otherwise, the two lines must be parallel to the x-axis and we define y_{ij} to be the mean of their y-coordinates. By testing the line $y=y_m$ (where y_m is the median of the y_{ij}'s) we identify a set of at least half the pairs whose y_{ij} values are either all greater than the y-coordinate of an optimal solution or all smaller than that. We then test the line $x=x_m$ (where x_m is the median of the x_{ij}'s of those pairs of nonparallel lines, that have been identified after the test at $y=y_m$). For at least a quarter of the pairs

we will be able to find a line and a corresponding half plane in which the optimal solution cannot lie. This enables us to drop one inequality per pair for at least a quarter of the pairs, i.e., at least $\frac{1}{16}$ of the inequalities are dropped. This establishes the linearity of the run time of our algorithm. Again, the details are given in § 4.3.

5.4. Quadratic programming in R^3. Our results can be extended to solve the problem of minimizing a convex quadratic function, subject to linear constraints, in R^3 in linear time. Formally, the problem is

$$\begin{aligned} &\underset{v\in R^3}{\text{minimize}} && v^T\cdot A\cdot v + b^T\cdot v \\ &\text{s.t.} && a_i^T\cdot v \leqq \beta_i \qquad (i=1,\cdots,n) \end{aligned}$$

where A is a 3×3 positive semidefinite matrix and $b, a_1, \cdots, a_n \in R^3$. If A is not identically zero then by an appropriate affine transformation of R^3 (which can be found in constant time) we can transform our problem (in linear time) to the following:

$$\begin{aligned} &\underset{x,y,z}{\text{minimize}} && \alpha x^2+\beta y^2+\gamma xy+z^2 \\ &\text{s.t.} && z \geqq a_i x+b_i y+c_i \qquad (i\in I_1), \\ & && z \leqq a_i x+b_i y+c_i \qquad (i\in I_2), \\ & && a_i x+b_i y+c_i \leqq 0 \qquad (i\in I_3) \end{aligned}$$

where $\gamma^2 \leqq 4\alpha\beta$, $\alpha, \beta > 0$. The algorithm for linear programming can be extended to solve a problem of this kind along the same lines. That includes a routine for solving quadratic programming problems in the plane. The latter can be modeled as

$$\begin{aligned} &\underset{x,y}{\text{minimize}} && \alpha x^2+\beta x+y^2 \\ &\text{s.t.} && y \geqq a_i x+b_i \qquad (i\in I_1), \\ & && y \leqq a_i x+b_i \qquad (i\in I_2), \\ & && a \leqq x \leqq b \end{aligned}$$

where $\alpha, \beta \geqq 0$. Let $g(x)$ and $h(x)$ be defined as in § 2.1. Also, let $g^+(x) = \text{Max}\,(0, g(x))$ and $h^-(x) = \text{Min}\,(0, h(x))$. Consider the following problems:

(1)
$$\begin{aligned} &\text{minimize} && \alpha x^2+\beta x+(g^+(x))^2 \\ &\text{s.t.} && g^+(x) \leqq h(x), \\ & && a \leqq x \leqq b; \end{aligned}$$

(2)
$$\begin{aligned} &\text{minimize} && \alpha x^2+\beta x+(h^-(x))^2 \\ &\text{s.t.} && g(x) \leqq h^-(x), \\ & && a \leqq x \leqq b. \end{aligned}$$

It can be verified that our problem reduces to solving both these problems. A solution for our problem is then produced as follows: Select the problem whose minimum is smaller and let x be its minimizer. We then let y be equal to either $g^+(x)$ or $h^-(x)$, according to the case. The solution of either problem is analogous to linear programming in R^2. A similar observation holds for the variable z in the three-dimensional case.

6. Conclusion. We have demonstrated a powerful computational method for solving different problems which is based on successive reductions of the input of the given problem. The method yields linear-time algorithms. The natural question now is whether the general linear programming problem is solvable in linear time in any fixed number of variables. We already know that the answer is in the affirmative [M3]; however, this requires a nontrivial extension of the present paper, as we argue below.

Consider any pair of constraints

$$y \geqq \sum_{j=1}^{d-1} a_{1j}x_j + b_1, \qquad y \geqq \sum_{j=1}^{d-1} a_{2j}x_j + b_2$$

in a linear programming problem where we seek to minimize y. In the space of the x_j's we have the hyperplane $H_{12} \equiv \sum a_{1j}x_j + b_1 = \sum a_{2j}x_j + b_2$ which defines two half spaces; in each of these half spaces we have one of the two constraints dominated by the other one. Thus, it may be useful to know on which side of H_{12} the solution lies. Attempting to generalize what we know from the case where $d = 3$, we do the following: Let H_{12} be represented by an equation of the form $\sum a_j x_j = b$, and for any $S \subset \{1, \cdots, d-1\}$, denote

$$R^S = \{x \in R^{d-1}: x_i \geqq 0 \text{ if } j \in S \text{ and } x_i \leqq 0 \text{ if } j \notin S\}.$$

Also let $T = \{j: a_j \geqq 0\}$ and $\bar{T} = \{1, \cdots, d-1\} \backslash T$. It can be easily verified that at least one of the two orthants, R^T and $R^{\bar{T}}$, lies entirely on one side of H_{12} (depending on the sign of b). For example, if $b \leqq 0$ then R^T lies entirely on one side of H_{12}. In this case, if we knew that our solution lay in R^T then we could drop one of our two constraints mentioned above.

For the case where $d = 3$ we found a way to exploit this useful observation. We will now review that method from a somewhat different point of view.

First, the lines are grouped in disjoint pairs in a manner which takes their slopes into account. Then a single point (x_m, y_m) is found with the following property: If the origin is translated into (x_m, y_m), and if a certain linear transformation is applied, then, relative to the new coordinate system, the solution lies in an orthant which lies entirely on one side of each line, for $\frac{1}{8}$ of the set of all lines. This is based on the property that if the solution lies in the orthant R^T, then at least a quarter of the pairs of lines intersect in $R^{\bar{T}}$, and each pair is guaranteed to contain a line whose coefficients suit the sign type of R^T. This idea works in the case where $d = 3$, since then there are only two pairs of "opposing" orthants (recall that the space of x_j's is of dimension $d-1$): $(R^{\{1,2\}}, R^{\varnothing})$ and $(R^{\{1\}}, R^{\{2\}})$. Thus if, for example, the solution is known to belong to $R^{\{1\}}$, then we also have a quarter of our pairs intersecting in $R^{\{2\}}$, and at least one line per such pair has the orthant $R^{\{1\}}$ lying on one side of it.

In higher dimensions we face the following difficulty. Suppose that in the d-dimensional case (the dimension of the x-space is $d-1$) we group the hyperplanes in disjoint sets of cardinality $d-1$. Suppose that each such set is linearly independent so that it determines a single point (the other case is even simpler). Now, it is obvious that we can find a point $(x_1, \cdots, x_{d-1})$ and an orthant R^T (defined relative to this point), such that $R^{\bar{T}}$ contains the solution, and at least $1/2^{d-1}$ of the sets intersect in $R^{\bar{T}}$. However, here is the critical point. The number of pairs of opposing orthants is 2^{d-2}. We need each set of $d-1$ hyperplanes to contain (for each pair of orthants $(R^T, R^{\bar{T}})$) at least one member whose coefficients match the signs pattern of $(R^T, R^{\bar{T}})$. It is thus required that $d-1 \geqq 2^{d-2}$, which holds only if $d = 3$. Thus, a different approach is needed here. It is also interesting to mention here that previous work on

the convex hull [PH] applied only for $d \leqq 3$. However, we already know that the methods of the present paper do extend to higher dimensions [M3].

With regard to practical implications of the algorithms in the present paper we can say the following: First, we do not expect the linear-time median-finding algorithm to be useful. Thus, it is preferable to use a good practical algorithm like Floyd and Rivest's [FR]. Moreover, we do not have to find the exact median. Another practical consideration is that information may be saved when starting a new iteration of the process. A practical version of our algorithm would be very efficient for solving problems arising in computer graphics such as hidden-line elimination by means of linear programming (see [BS]).

Appendix. The extreme-point problem in the plane. In this Appendix we solve the following problem: Given n points (a_i, b_i), $i = 1, \cdots, n$, in the plane and another point (a, b), find out whether or not (a, b) is a convex combination of the points $(a_1, b_1), \cdots, (a_n, b_n)$. If so, then represent (a, b) as a convex combination of three points; otherwise, find two points (a_i, b_i), (a_j, b_j) such that all the points belong to the cone whose vertex is at (a, b) and whose extreme rays are determined by (a_i, b_i) and (a_j, b_j). This problem can be solved by our linear programming algorithm in the plane. However, a more straightforward method can be developed as follows:

Without loss of generality assume $(a, b) = (0, 0)$ and $(a_1, b_1) = (0, 1)$ (otherwise apply an affine transformation accordingly). If $(0, 0)$ is not in the convex hull of the n given points, then there exists an α such that $b_i > \alpha a_i$ for every i. That is, there is a separating line (as mentioned in the introduction). Thus, α must satisfy

$$\max \{b_i/a_i : a_i < 0\} < \alpha < \min \{b_i/a_i : a_i > 0\},$$

and also $b_i > 0$, for every i such that $a_i = 0$. If this is indeed the case, then the extreme rays are determined by a point at which the maximum on the left-hand side is attained (or the point $(0, 1)$ if $a_i \geqq 0$ for every i), and by a point at which the minimum on the right-hand side is attained (or, again, the point $(0, 1)$). Otherwise, the point $(0, 0)$ is either a convex combination of two such points together with the point $(a_1, b_1) = (0, 1)$, or a convex combination of two points on the y-axis.

REFERENCES

[AHU] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[BS] R. P. BURTON AND D. R. SMITH, *A hidden-line algorithm for hyperspace*, this Journal, 11 (1982), pp. 71–80.

[CP] R. CHANDRASEKARAN AND M. J. A. P. PACCA, *Weighted min-max and max-min location problems: Finite and polynomially bounded algorithms*, Oper. Res., 17 (1980), pp. 172–180.

[CR] R. COURANT AND H. ROBBINS, *What is Mathematics?*, Oxford Univ. Press, New York, 1941.

[DF] P. M. DEARING AND R. L. FRANCIS, *A minimax location problem on a network*, Transportation Sci., 8 (1974), pp. 333–343.

[DR] D. P. DOBKIN AND S. P. REISS, *The complexity of linear programming*, Theoret. Comput. Sci., 11 (1980), pp. 1–18.

[DW] Z. DREZNER AND G. O. WESOLOWSKY, *Single Facility l_p-distance minimax location*, SIAM J. Alg. Disc. Meth., 1 (1980), pp. 315–321.

[EH] J. ELZINGA AND D. E. HEARN, *Geometrical solutions for some minimax location problems*, Transportation Sci., 6 (1972), pp. 379–394.

[FR] R. W. FLOYD AND R. L. RIVEST, *Expected time bounds for selection*, Comm. ACM, 8 (1975), pp. 165–172.

[F] R. R. L. FRANCIS, *Some aspects of a minimax location problem*, Operat. Res., 15 (1967), pp. 1163–1168.

[FW] R. L. FRANCIS AND J. A. WHITE, *Facility Layout and Location*, Prentice-Hall, Englewood Cliffs, NJ, 1974.

[G] R. L. GRAHAM, *An efficient algorithm for determining the convex hull of a finite planar set*, Inform. Process. Lett., 1 (1972), pp. 132–133.

[HSP] S. L. HAKIMI, E. F. SCHMEICHEL AND J. G. PIERCE, *On p-centers in networks*, Transportation Sci., 12 (1978), pp. 1–15.

[HM] G. Y. HANDLER AND P. B. MIRCHANDANI, *Location on Networks Theory and Algorithms*, MIT Press, Cambridge, MA, 1979.

[KH] O. KARIV AND S. L. HAKIMI, *An algorithmic approach to network location problems, Part* I. *The* 1*-centers*, SIAM J. Appl. Math., 37 (1979), pp. 513–538.

[L] N. LEVIN, *Location problems on weighted graphs*, unpublished thesis, Tel Aviv Univ., 1977.

[M1] N. MEGIDDO, *Applying parallel computation algorithms in the design of serial algorithms*, in Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Angeles, 1981, pp. 399–408; J. Assoc. Comput. Math., to appear.

[M2] ———, *The weighted Euclidean* 1*-center problem*, Math. Oper. Res., to appear.

[M3] ———, *Linear programming in linear time when the dimension is fixed*, J. Assoc. Comput. Mach., to appear.

[NC] K. P. K. NAIR AND R. CHANDRASEKARAN, *Optimal location of a single service center of certain types*, Naval Res. Logist. Quart., 18 (1971), pp. 503–510.

[PH] F. P. PREPARATA AND S. J. HONG, *Convex hulls of finite sets of points in two and three dimensions*, Comm. ACM, 20 (1977), pp. 87–93.

[RT] H. RADEMACHER AND O. TOEPLITZ, *The Enjoyment of Mathematics*, Princeton Univ. Press, Princeton, NJ, 1957.

[Sh] M. I. SHAMOS, *Geometric complexity*, in Proc. 7th Annual ACM Symposium on Theory of Computing, 1975, ACM, New York, 1975, pp. 224–233.

[ShH] M. I. SHAMOS AND D. HOEY, *Closest-point problems*, in Proc. 16th Annual IEEE Symposium on Foundations of Computer Science, 1975, IEEE Computer Society Press, Los Angeles, 1975, pp. 151–162.

[Sm] R. D. SMALLWOOD, *Minimax detection station placement*, Oper. Res., 13 (1965), pp. 636–646.

[Sy1] J. J. SYLVESTER, *A question in the geometry of situation*, Quart. J. Math., 1 (1857), p. 79.

[Sy2] ———, *On Poncelet's approximate valuation of Surd forms*, Philosophical Mag., XX, 4th Series (1860), pp. 203–222.

[Y] A. C. YAO, *A lower bound for finding convex hulls*, J. Assoc. Comput. Mach., 28 (1981), pp. 780–787.