Megiddo, Nimrod; Zemel, Eitan

Working Paper

An O (n log n) Randomizing Algorithm for the Weighted Euclidean l-Center Problem

Discussion Paper, No. 613

Provided in Cooperation with:
Kellogg School of Management - Center for Mathematical Studies in Economics and Management Science, Northwestern University

Suggested Citation: Megiddo, Nimrod; Zemel, Eitan (1984) : An O (n log n) Randomizing Algorithm for the Weighted Euclidean l-Center Problem, Discussion Paper, No. 613, Northwestern University, Kellogg School of Management, Center for Mathematical Studies in Economics and Management Science, Evanston, IL

This Version is available at:
https://hdl.handle.net/10419/220972

Discussion Paper No. 613

AN O(n log n) RANDOMIZING ALGORITHM FOR THE WEIGHTED EUCLIDEAN 1-CENTER PROBLEM

by

Nimrod Megiddo*
and
Eitan Zemel**

August 1984

---

*Department of Computer Sciences, Stanford, California 94305, XEROX Palo Alto Research Center, and Tel Aviv University, Tel Aviv, Israel.

**Department of Managerial Economics and Decision Sciences, J. L. Kellogg Graduate School of Management, Northwestern University, Evanston, Illinois 60201, and the Leon Recanati Graduate School of Business Administration, Tel Aviv University, Tel Aviv, Israel.

This work was supported in part by the National Science Foundation under Grant ECS-8121741.

AN $O(n \log n)$ RANDOMIZING ALGORITHM FOR THE WEIGHTED EUCLIDEAN 1-CENTER PROBLEM*

NIMROD MEGIDDO† and EITAN ZEMEL‡

A randomizing algorithm for the weighted Euclidean 1-center problem is presented. The algorithm is shown to run on any problem in $O(n \log n)$ time with high probability.

1. Introduction.

The *weighted Euclidean 1-center problem* is defined as follows. Let n points, $p_i = (x_i, y_i)$, $(i = 1, \cdots, n)$ be given together with positive weights, w_i $(i = 1, \cdots, n)$. For any point $p = (x, y)$, let

$$d(p, p_i) = \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

and

$$H(p) = H(x, y) = \max_{1 \le i \le n} w_i d(p, p_i) \quad .$$

The weighted Euclidean 1-center problem is to find a point $p^* = (x^*, y^*)$ so as to minimize H over R^2. The special case where all the w_i's equal 1 (that is, the unweighted 1-center problem) was proposed in 1857 by Sylvester and amounts to finding the smallest circle containing all the given points. Algorithms for the unweighted case were given in [CR], [CC], [EH], [M1], [NC], [RT], [SH] and [Sm]. Megiddo's algorithm [M1] runs in linear time. The Elzinga-Hearn algorithm [EH] is claimed to be very practical [HV].

The more general weighted case was introduced in [F]. It can be solved in $O(n^3)$ time (by enumerating all triples of points) under a model in which square roots are computed in constant-time (see [CT] for a treatment of the square roots issues in general). Megiddo [M3] gave an $O(n(\log n)^3(\log \log n)^2)$ algorithm for the weighted problem, and later an improvement based on the methods in [M2] was found, which runs in $O(n(\log n)^2)$. The observation of [CT] eliminates the need to compute square roots in these algorithms. In another paper, Chandrasekaran [C] pointed out that the weighted center problem in general dimension can be solved in polynomial time, with the aid of an "ellipsoid" method. The unweighted problem can be solved in linear time whenever the dimension is fixed [M5]. We note that, relative to the l_1- and the l_∞-metrics, even the *weighted* problem can also be solved in linear time whenever the dimension is fixed [M5].

In this paper we develop a *randomizing* algorithm for the weighted problem, whose running time can be made $O(n \log n)$ with any prespecified probability of success (less than 1). The algorithm

---

*Sep. 1983; rev. Feb. 1984. This work was supported in part by the National Science Foundation under Grant ECS-8121741.

† Department of Computer Science, Stanford University, Stanford, CA 94305, XEROX Palo Alto Research Center and Tel Aviv University.

‡ Department of Managerial Economics and Decision Sciences, J. L. Kellogg Graduate School of Management, Northwestern University, Evanston, IL 60201, and the Leon Recanati Graduate School of Business Administration, Tel Aviv University, Tel Aviv, Israel.

relies on the parametric method developed by Megiddo [M1, M2]. The present paper builds upon ideas developed by the present authors in [M3, M4, Z1, Z2].

In order to simplify the presentation, we assume all weights are distinct and, in general, we assume elements to be distinct whenever we need to select or rank in a set of numbers. This assumption is made only for the convenience of presentation. Similarly, whenever we deal with a number of the form αn, where n is an integer and $0 < \alpha < 1$, we regard αn as integer (think, for example, of $\lfloor \alpha n \rfloor$).

The organization of the paper is as follows. In section 2 we present an $O(n)$ algorithm for a one-dimensional version of the problem with a constraint. In Section 3 we briefly discuss a randomizing version of the algorithm developed in Section 2. In Section 4 we present a parameterized version of the algorithm developed in Section 3. We show in Section 5 that this parameterized version solves the problem in $O(n \log n)$ time with high probability.

2. A one dimensional search.

Consider a vertical line L defined by the equation $x = x'$, for some fixed x'. Suppose we add to the weighted 1-center problem a constraint that the center must lie on L. We now have a one-dimensional problem, namely, to find a value $y' = y'(x')$ which minimizes the expression

$$h(y) = \max_{1 \le i \le n} w_i d((x', y), p_i) \quad .$$

Note that $h(y)$ is convex. Let the value of y be fixed, and let i be a maximizer of the weighted distance, that is,

$$h(y) = w_i d((x', y), p_i) \quad .$$

It is easy to verify that the maximum distance can be reduced only if y gets closer to y_i, that is, $y'(x') \ge y$ if and only if $y_i \ge y$. Also note that, given y, a maximizer i can be found in $O(n)$ time. This implies that we can answer a query like "Is $y \le y'$?" in $O(n)$ time for any y. We call such (vertical) queries *tests*.

The procedure for solving the one-dimensional constrained problem is essentially the same as the two-variable linear programming algorithm [M4]. Instead of minimizing the maximum of a set of linear functions (in the linear programming problem), here we have to minimize the maximum of a set of convex parabolas. The reader may like at this point to verify that a linear-time algorithm for the latter can be developed along the lines of [M4]. We, however, describe here an algorithm using the terminology of the weighted center problem. This will help us later in describing the two-dimensional algorithm. Consider the equation

$$w_i^2[(x_i - x)^2 + (y_i - y)^2] = w_j^2[(x_j - x)^2 + (y_j - y)^2] \quad .$$

This equation describes a circle C_{ij} such that $p = (x, y)$ is closer (in the weighted sense) to p_i than to p_j if and only if p is inside the circle. Our strategy in computing y' is to identify either one or two points at which the weighted distance to the center (x', y') is maximized. To that end, we eliminate points which are "dominated" in the following sense: A point p_i is dominated by p_j if the distance can be maximized at p_i only if it is also maximized at p_j. More specifically, if p' is inside the circle C_{ij} then p_j can be eliminated, and if p' is outside of C_{ij} then p_i can be eliminated. In either case the set of points is reduced. However, we need to find a way to acquire information about domination relations by performing a small number of tests.

Consider the intersection of the circle C_{ij} with the line L. This intersection is either empty or equal to an interval $Y_{ij} = [y_{ij}^-, y_{ij}^+]$, possibly with $y_{ij}^- = y_{ij}^+$. Now, let E denote the set of pairs of points whose circles do not intersect the line, and let F denote the set of all other pairs. Let $e = |E|$ and $f = |F|$. Obviously, one member of each pair in E can be eliminated without any further work, namely, the one with the smaller weight. Also, for a pair $(i, j) \in F$ we can eliminate p_i if y is not inside C_{ij} and p_j otherwise. The query "Is y inside C_{ij}?" can be answered by two tests, namely, at y_{ij}^- and y_{ij}^-. More efficiently, let y^- denote the median of the y_{ij}^-'s. Note that if $y' \leq y^-$ then for each (k, l) such that $y_{kl}^- \geq y^-$ we have $y' \leq y_{kl}^-$ and hence y' is not inside C_{kl}, so that one member from each such pair can be eliminated; note that there are $k/2$ such pairs. If $y' > y^-$, then consider the set F_1 of pairs k, l in F with $y'_{kl} \geq y^-$. Since y^- is the median, we also have $|F_1| = f/2$. Now, let y^+ be the median of the y'_{kl}'s in F_1. A test at y^+ identifies a subset F_1 of cardinality $|F_1|/2 = f/4$ so that one element can be eliminated from each pair in this set. All in all, we have identified at most two medians y^- and y^+, and performed no more than two tests. The overall effort so far is $O(n)$. In return, we eliminate at least $e/2 + f/8 \geq n/16$ points. In fact, it is easy to achieve a larger fraction, $n/12$, of eliminated points. By repeating this step until the set is exhausted, we establish a linear time algorithm.

Recall that an optimal solution for the unconstrained problem is denoted by (x^*, y^*). Once y' has been identified, we can easily decide whether or not $x^* \geq x'$. Specifically, let p_i be such that

$$w_i d(p', p_i) = \max_{1 \leq j \leq n} w_j \, d(p', p_j) \quad .$$

Then, $x^* \geq x'$ if and only $x_i \geq x'$. We call the query "Is $x \geq x^*$?" a *(horizontal) test* at x. Note that a horizontal test yields information as to the location of the unconstrained minimum p^*, whereas a vertical test deals with the constrained minimum p'.

3. The benefit from randomization.

The procedure for the constrained problem runs in linear time so there is no room for improvement by order of magnitude. Nevertheless, we will now present a randomizing version which runs in linear time with high probability. It is this randomizing algorithm which will be parameterized later, improving the solution of the *unconstrained* problem by order of magnitude.

The randomization enters the algorithm via the choice of y^- and y^+. Rather than picking y^- as the median of the set of y_{kl}^-'s, we simply pick y^- to be any element with the same probability. This means that the number of points eliminated in each iteration is a random variate. We note that the situation in our algorithm looks like an extension of the FIND algorithm [Ho], where the k-th element a^* of an ordered (but not sorted) set $\{a_1, \cdots, a_n\}$ is selected by repeating the following step: Pick a random element a_i of the currently remaining set and eliminate all those elements which are separated by a_i from the k-th element of the original set. In this algorithm the k-th element is unknown throughout but it can easily be decided whether or not $a_i \leq a^*$, so that we can eliminate all those elements a_j such that a_i is between a_j and a^*. The expected number of comparisons that algorithm FIND makes was analyzed by Knuth [K] for any combination of n and k. Our problem seems much more complicated for such an anlysis, since it cannot be characterized by an ordered set (of fixed cardinality) of numbers. In particular, in our algorithm after points have been eliminated, new pairs are formed which add new values to the domain we are searching. Thus, we will evaluate *upper-bounds* on the expected time. In Section 5 we show that the expected time for running the one-dimensional search this way is linear and in fact for any positive ϵ there is a constant C (independent of n) such that the running time is less than Cn with probability greater

than $1 - \epsilon$. Of course, this result is not so significant for the one-dimensional problem alone. The significance is that it improves the running time of the algorithm for the weighted center problem.

4. The two-dimensional search.

As we have said before, the weighted 1-center problem can be viewed as a problem of searching for an optimal point $p^* = (x^*, y^*)$. Now, if x^* is known then we can find y^* in additional $O(n)$ time, using the one-dimensional search procedure of Section 2. The general idea of Megiddo's parametric method [M1, M2] is to perform the one-dimensional search with x^* being indeterminate. The parametric algorithm frequently pauses and requests information which helps reduce the interval of indeterminacy, to enable proceeding with the execution. The reader is advised to gain better idea of this method from the previous papers. When the parametric search ends, the algorithm has produced an interval which contains x^* along with sufficient information for computing both x^* and y^* in essentially one step. We will explain in detail how this is carried out.

We will first review the basic steps of a single iteration of the one-dimensional search. Later, we will specify how these steps can be carried out with x^* being indeterminate. While reading the following description, the reader is advised to consider x^* as an indeterminate, which is known to belong to a certain interval.

Algorithm . *One-dimensional search*

1. [*Pairing.*] Form disjoint pairs of points (i, j) and compute the equations of the circles C_{ij}.

2. [*Classify pairs.*] Identify the set E of pairs (i, j) whose circles C_{ij} do not intersect the line $x = x^*$. Denote by F the set of the remaining pairs. For each pair in E, eliminate the point with the smaller weight.

3. [*First Sample-and-Test.*] Pick a random pair $(i, j) \in F$ and perform a vertical test at $y^- = y_{ij}^-$. The test amounts to finding a point p_r at which the weighted distance is maximized, that is

$$w_r\, d(p_r, (x^*, y^-)) = \max_{1 \le k \le n} w_k\, d(p_k, (x^*, y^-)) \quad .$$

4. [*First Elimination.*] If $y_r \le y^-$ then for each pair $(k, l) \in F$ such that $y_{kl}^- \ge y^-$, eliminate the point with the smaller weight and terminate the current iteration.

5. [*Second Sample-and-Test.*] If $y_r > y^-$ then pick a random pair (i, j) from the set F_1 of pairs (i, j) such that $y_{ij}^- \le y^-$ and perform a vertical test at $y^+ = y_{ij}^+$.

6. [*Second Elimination.*] If $y_r \ge y^+$ then for each pair $(k, l) \in F_1$ (that is, $y_{kl}^+ \le y^+$), eliminate the point with the smaller weight; otherwise, for each pair not in F_1 eliminate the point with the larger weight. ∎

We now examine the role of x^* in each of the steps. Obviously, Step 1 does not depend on x^*. Also, Steps 5 and 6 are analogous to 3 and 4, respectively, and do not require a separate treatment.

Consider Step 2. Here, the sets E and F do depend on x^*. For each circle C_{ij}, let $[a_{ij}, b_{ij}]$ denote the projection of C_{ij} on the x-axis. Obviously, $(i, j) \in F$ if and only if $x^* \in [a_{ij}, b_{ij}]$. Let X be the set of all endpoints a_{ij}, b_{ij}. We can locate x^* within X by $\log|X|$ horizontal tests. Each test takes $O(n)$ time and hence this step takes $O(n \log n)$ time. At the end of Step 2 we know the sets E and F, even though the value of x^* is yet unknown.

Consider Step 3. Here, we first pick a random pair $(i,j) \in F$ (clearly, an operation which is independent of x^* after the set F has been determined), but then the value y_{ij}^- is a function of x^*. The graph of this function coincides with the lower half of the circle C_{ij}. Denote this half-circle by D_{ij}. Next, we need to find a point p_r at which the weighted distance from the (yet unknown) point (x^*, y_{ij}^-) is maximized.

We now describe a procedure for finding a point p_r at which the weighted distance from the point (x^*, y_{ij}^-) is maximized. The steps that we define in the present paragraph are of course "local" and should not be confused with the progress of the main algorithm. The present procedure is quite similar to the main one. We first form disjoint pairs (k,l) and look at their respective critical circles C_{kl}. For each circle, there are at most two intersection points with the half-circle D_{ij}. Thus, there are at most n such points. Let X denote the set of the x-components of this points. We can obviously locate x^* within X by $O(\log n)$ horizontal tests. Thus, this search takes $O(n \log n)$ time. As a result, we know for each pair (k,l) whether the point (x^*, y_{ij}^-) lies inside or outside the circle C_{kl}. So, we can eliminate (just for the sake of the present step) one member of the pair (k,l). Reiterating this idea eventually leads to one point p_r at which the distance from the point (x^*, y_{ij}^-) is maximized. The entire procedure in this paragraph still takes $O(n \log n)$ time.

Consider Step 4. We first need to decide whether $y_r \leq y_{ij}^-$, but the exact value of y_{ij}^- is yet unknown since it is a function of x^*. However, like in the previous cases, we can carry the step out without knowing x^* precisely. Consider the horizontal line $y = y_r$. This line intersects the half-circle D_{ij} at no more than two points, (a_1, y_r), (a_2, y_r). Thus, it takes at most two horizontal tests (at a_1 and a_2) to tell whether at x^* the line $y = y_r$ lies above or beneath the arc D_{ij}. This takes $O(n)$ time. Suppose $y_r \leq y_{ij}^-$ (otherwise we go to Step 5). We now need to eliminate the point of the smaller weight from each pair (k,l) such that $y_{kl}^- \leq y_{ij}^-$, where both y_{ij}^- and y_{kl}^- are functions of x^*. Once again, this can be carried out even though we do not know the exact value of x^*. We consider the half-circles D_{kl}. There are at most n intersection points of such arcs with the half-circle D_{ij}. By $O(\log n)$ horizontal tests we can locate x^* within the set of x-values of these intersection points, and thus we can eliminate one point per pair for (k,l) such that $y_{kl}^- \leq y_{ij}^-$.

We have shown that each of the steps of the one-dimensional search can be carried out in $O(n \log n)$ time even if x^* is not known, since we can always find the necessary information about x^* by performing $O(\log n)$ horizontal tests, at the cost of $O(n)$ per test. Note that a single iteration of this parametric algorithm takes no more than $C n' \log n'$ time, where n' is the number of remaining points at the start of the iteration, and C is a constant independent of the random elements. The choice of the random elements affects only the *number of iterations* and not the cost of a single one, given the number n'.

5. The probabilistic estimates.

We now obtain an upper bound on the expected time required by the algorithm. Notice that we are interested in the expected time for the *worst-case* instance and the expectation is only relative to the *internal* randomization of the algorithm.

We first find an upper-bound on the expected number of points eliminated during a single iteration. At any instant of time during the execution of the algorithm the input points can be classified as active or inactive. At the beginning all are active and during the execution each becomes inactive at some stage. Let us consider a single iteration of the algorithm and denote by n' the current number of active points. For convenience, let us rename the active points to be p_i, $(i = 1, \cdots, n')$. At the end of the current iteration certain active points become inactive. Let us

denote by S the cardinality of the subset of these points. Note that S is a random variate whose distribution depends on the past history of the execution of the algorithm. However, it is possible to bound the expectation of S (as a function of n') from below in a useful way, regardless of that past history.

Lemma . *For any problem instance, regardless of the past history of the algorithm, the expected number of points eliminated during any iteration is at least $n'/20$.*

► *Proof:* Obviously, one point from each pair $(i,j) \in E$ is always eliminated, so it suffices to consider the set F. For simplicity, let us index the pairs of points in F with $\alpha \in \{1,\cdots,f\}$, where $f = |F|$. Thus, quantities like y_α^- are well-defined whenever x is fixed. For the purpose of the discussion let us fix $x = x^*$, even though x^* is of course not known at the current stage. Without loss of generality, assume that $1 \le \alpha \le \beta \le f$ implies $y_\alpha^- \le y_\beta^-$. Now, let k denote the rank of the number y^* in the set $\{y_\alpha^- : \alpha = 1,\cdots,f\}$, that is, $y_k^- < y^* < y_{k+1}^-$. Note that k is completely determined by the past history of the execution, even though it is not known at the current stage, since y^* is not known. At this point the algorithm picks a random pair from F. The rank of the corresponding y^- value in the set $\{y_1^-,\cdots,y_f^-\}$, which we denote by K, is a random variate which is uniformly distributed over $\{1,\cdots,f\}$. Recall that if $K > k$ then, for every pair whose rank is greater than or equal to K, we eliminate one point. Thus, $f - K + 1$ points are eliminated in this case. Otherwise, when $K \le k$, we proceed to the next step with the corresponding K pairs, that is, those with $\alpha \le K$. Once again, for convenience, let us rename these K pairs and index them so as to conform with the y^+ values, that is, $1 \le \alpha \le \beta \le K$ implies $y_\alpha^+ \le y_\beta^+$. Let r denote the rank of y^* in the set $\{y_1^-,\cdots,y_K^+\}$. As was the case with k, the rank r is determined at the current stage, yet unknown to the algorithm. Now the algorithm picks a random pair whose rank R is uniformly distributed over $\{1,\cdots,K\}$. If $R \le r$ then R points are eliminated. Otherwise, $K - R$ points are eliminated.

Consider the case where $K \le k$. Given the value of K, the expected number of points eliminated during the second step is at least $K/4$. This follows from the fact that even if an adversary could change the value of y^* after R had been picked, then the most he could achieve would be that the smaller of the two sets determined by R be eliminated. The expected cardinality of the smaller set determined by a uniformly distributed R is equal to one quarter of the cardinality of the grand set. On the other hand, if $K > k$ then $f - K + 1$ points are eliminated. It follows that the expected number of points eliminated altogether is

$$\frac{1}{f}\left(\sum_{K=1}^{k}\frac{K}{4} + \sum_{K=k+1}^{f}(f-K+1)\right) \quad > \quad \frac{1}{f}\left(\frac{5}{8}k^2 - fk + \frac{1}{2}f^2\right).$$

Even if an adversary could freely change the value of k, then the expected number of points eliminated would be at least the minimum of the quadratic function of k we have just calculated. The minimum is attained at $k = 4f/5$ and is equal to $f/10$. Since $f \le n'/2$, it follows that we eliminate an expected number of at least $n'/20$ points during each iteration. ◄

Corollary . *The expected running time of the one-dimensional search algorithm is linear.*

► *Proof:* The linear upper bound is established by considering the performance of the algorithm against an adversary who is able to change the value of k between iterations. Let N_i denote the number of active points after the i-th iteration. In particular, $N_0 = n$. Consider the ratios $Y_i = N_i/N_{i-1}$, $i = 1,2,\cdots$. If the algorithm is run against the said adversary then

the random variates $Y_1, Y_2, \cdots$ are *independent* and each has expectation of at most $19/20$. The effort in performing an iteration starting with N_i points is bounded by $C'N_i$ where C' is a certain constant. Thus, the expected effort in performing the i-th iteration is bounded by $C'n\mathcal{E}[Y_1Y_2\cdots Y_{i-1}]$ (where $\mathcal{E}$ stands for expectation), which equals $C'n(19/20)^{i-1}$. It follows that the expected total effort is bounded from above by $20C'n$. ◄

As a matter of fact, it follows from the analysis here that for every $\epsilon > 0$ there exists a constant $C^* = C^*(\epsilon)$ such that for every n and every problem of n points, the probability that the running time will be greater than C^*n is less than ϵ.

Finally, the expected time required for solving the weighted center problem is $O(n \log n)$ by an analogous argument. The effort of performing an iteration starting with n' points is bounded from above by $C''n \log n$ and hence the total effort has an expected value less than $20C''n \log n$. Again, for every positive ϵ there exists C^* such that the probability that the running time will be greater than $C^*n \log n$ is less than ϵ.

References

[C] R. Chandrasekaran, "The weighted Euclidean 1-center problem", *Operations Research Letters* 1 (1982) pp. 111-112.

[CT] R. Chandrasekaran and A. Tamir, "Optimization problems with algebraic solutions: Quadratic fractional programs and ratio games", manuscript, December 1982.

[CC] R. K. Chakraborty and P. K. Chaudhuri, "Note on geometrical solutions for some minimax location problems", *Transportation Science* 15 (1981) pp. 164-166.

[CR] R. Courant and H. Robbins, *What is mathematics?*, Oxford University Press, New York, 1941.

[D] M. E. Dyer, "An $O(n)$ algorithm for the multiple-choice knapsack linear program", manuscript, 1982.

[EH] J. Elzinga and D. W. Hearn, "Geometrical solutions for some minimax location problems", *Transportation Science* 6 (1972) pp. 379-394.

[FR] R. W. Floyd and R. L. Rivest, "Expected time bounds for selection", *Comm. Assoc. Comput. Mach.* 18 (1975) pp. 165-172.

[F] R. L. Francis, "Some aspects of a minimax location problem", *Operations Research* 15 (1967) pp. 1163-1168.

[HV] D. W. Hearn and J. Vijay, "Efficient algorithms for the (weighted) minimum circle problem", *Operations Research* 30 (1982) pp. 777-795.

[Ho] C. A. R. Hoare, "Algorithm 65: Find", *Comm. Assoc. Comput. Mach.* 4 (1961) pp. 321-322.

[K] D. E. Knuth, "Mathematical analysis ofalgorithms", in *Information Processing 71*, Elsevier North-Holland, New York, 1972, pp. 19-27.

[M1] N. Megiddo, "Combinatorial optimization with rational objective functions", *Math. of Oper. Res.* 4 (1979) pp. 414-424.

[M2] N. Megiddo, "Applying parallel algorithms in the design of serial algorithms", *J. Assoc. Comput. Mach.* 30 (1983) pp. 852-865; also in *Proc. 22nd IEEE Symp. Foundations of Computer Science*, IEEE, New York, 1981, pp. 399-408.

[M3] N. Megiddo, "The weighted Euclidean 1-center problem", *Math. of Oper. Res.* 8 (1983) pp. 498-504.

[M4] N. Megiddo, "Linear time algorithms for linear programming in R^3 and related problems", *SIAM J. Comput.* 12 (1983) pp. 759-776.

[M5] N. Megiddo, "Linear programming in linear time when the dimension is fixed", *J. Assoc. Comput. Mach.* 31 (1984) pp. 114-127.

[NC] K. P. K. Nair and R. Chandrasekaran, "Optimal location of a single service center of certain types", *Naval Res. Log. Quart.* 18 (1971) pp. 503-510.

[RT] H. Rademacher and O. Toeplitz, *The enjoyment of mathematics*, Princeton University Press, Princeton, N.J., 1970.

[SH] M. I. Shamos and D. Hoey, "Closest-point problems", *Proc. 16th IEEE Symp. Foundations of Computer Science* IEEE, New York, 1975, pp. 151-162.

[Sm] R. D. Smallwood, "Minimax detection station", *Operations Research* 13 (1965) pp. 151-162.

[Z1] E. Zemel, "A linear time randomized algorithm for finding roots and optima of ranked functions", manuscript, 1983.

[Z2] E. Zemel, "An $O(n)$ algorithm for multiple choice knapsack and related problems", *Information Processing Letters* to appear.