Algorithmica (1994) 11: 185–195

Algorithmica

Planar Geometric Location Problems[1]

Pankaj K. Agarwal[2] and Micha Sharir[3]

Abstract. We present an $O(n^2 \log^3 n)$ algorithm for the two-center problem, in which we are given a set S of n points in the plane and wish to find two closed disks whose union contains S so that the larger of the two radii is as small as possible. We also give an $O(n^2 \log^5 n)$ algorithm for solving the two-line-center problem, where we want to find two strips that cover S whose maximum width is as small as possible. The best previous solutions of both problems require $O(n^3)$ time.

Key Words. Facility location, Parametric searching, Duality, Planar arrangements.

1. Introduction. In this paper we consider the following two problems involving a set S of n points in $\mathbb{R}^2$:

TWO-CENTER PROBLEM. Find two closed disks whose union contains S so that the larger radius of the two disks is as small as possible (see Figure 1).

TWO-LINE-CENTER PROBLEM. Find two strips whose union contains S so that the larger width of the two strips is as small as possible (see Figure 1).

The two-center problem is a special case of a basic problem in geometric location theory. It is an intermediate version between the simpler one-center problem (seeking the smallest disk enclosing S), which can be solved in linear time [M2], [LW], and the general p-center problem (seeking p disks of smallest maximum radius whose union covers S), which can be solved in time $n^{O(p)}$, and is NP-complete when p is part of the input [FG], [MS], [Me]. Similarly, the two-line-center problem is an intermediate version between the one-line-center problem (seeking a strip of minimum width covering S), which can be solved in $O(n \log n)$ time by computing the width of S, and the general p-line-center problem (seeking p strips

[1] Pankaj Agarwal has been supported by DIMACS (Center for Discrete Mathematics and Theoretical Computer Science), an NSF Science and Technology Center, under Grant STC-88-09648. Micha Sharir has been supported by the Office of Naval Research under Grants N00014-89-J-3042 and N00014-90-J-1284, by the National Science Foundation under Grant CCR-89-01484, by DIMACS, and by grants from the US–Israeli Binational Science Foundation, the Fund for Basic Research administered by the Israeli Academy of Sciences, and the G.I.F., the German–Israeli Foundation for Scientific Research and Development. A preliminary version of this paper has appeared in *Proceedings of the Second Annual ACM–SIAM Symposium on Discrete Algorithms*, 1991, pp. 449–458.
[2] Computer Science Department, Duke University, Durham, NC 27706, USA.
[3] School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel, and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA.

Received August 31, 1990; revised: September 6, 1991. Communicated by Bernard Chazelle.

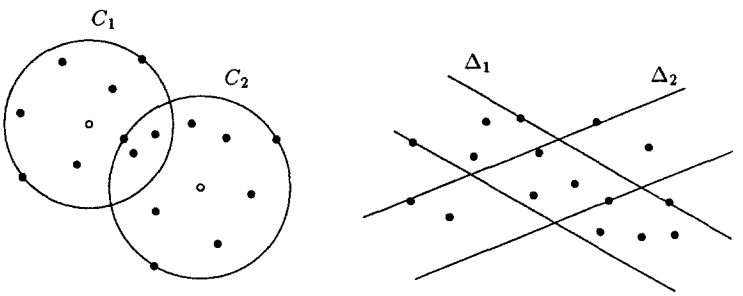

Fig. 1. The two-center and two-line-center problems.

of smallest maximum width whose union covers S), which is again known to be NP-complete [MS]. The running time of the best previously known algorithms for both of these problems is $O(n^3)$ [D]. In this paper we present an $O(n^2 \log^3 n)$ algorithm for the two-center problem and an $O(n^2 \log^5 n)$ algorithm for the two-line-center problem. Both algorithms are based on Megiddo's parametric search technique [M1], which we apply in a rather unusual manner, to be detailed below.

Our two-center algorithm requires a procedure for the *fixed-radius problem*, in which we are given a radius r and wish to determine whether S can be covered by two disks of radius r each. Such a procedure, running in $O(n^2 \log n)$ time, has recently been given by Hershberger and Suri [HS]. Similarly, our two-line-center algorithm requires a procedure for the *fixed-width problem*, in which, given a width W, we wish to determine whether S can be covered by two strips of width W each. This problem is more involved, and we derive an $O(n^2 \log^3 n)$ algorithm for its solution. The algorithm requires a subprocedure for efficient off-line dynamic maintenance of the width of a set of points in the plane, which we have recently obtained in a companion paper [AS].

The paper is organized as follows. Section 2 presents the solution to the two-center problem, and Section 3 solves the two-line-center problem, including the solution of the fixed-width problem mentioned above. The paper is concluded in Section 4 with a discussion of our results and open problems.

2. The Two-Center Problem. In this section we describe our solution to the two-center problem, which proceeds roughly as follows. Let r^* denote the smallest radius for which two (closed) disks with that radius whose union covers S exist. We want to find r^* by binary search over a sequence Σ of admissible values for that radius. The sequence Σ consists of the radii of the circumcircles of all triangles spanned by triples of points of S, and of the halves of all distances between pairs of points in S. This is indeed the case because any disk that covers a (nonsingleton) subset of S can be shrunk and translated, while still covering the same subset, until its bounding circle touches three points or two diametrically opposite points of that subset.

Unfortunately, there are $\Theta(n^3)$ such candidate values for r^*, so we cannot afford

to construct Σ explicitly. Instead we apply an implicit searching technique for locating r^* by using a somewhat unusual form of Megiddo's ingenious parametric search technique [M1]. We first give a review of Megiddo's technique, and then show how to fit it (in a somewhat nonstandard manner) into our problem.

2.1. Parametric Search Technique. The basic idea behind Megiddo's technique is as follows: Suppose we have a problem $\mathscr{P}(r)$ that receives as input n data items and a real parameter r. We want to find a value r^* of r at which the output of $\mathscr{P}(r)$ satisfies certain properties. Suppose we have an efficient (sequential) algorithm A_s for solving $\mathscr{P}(r)$ at any given r, and that, as a by-product, algorithm A_s can also determine whether the given r is equal to, smaller than, or larger than the desired value r^*. Assume moreover that the flow of execution of A_s depends on comparisons, each of which is resolved by testing the sign of a low-degree polynomial in r and in the input items.

Megiddo's technique then runs algorithm A_s "generically," without specifying the value of r, with the intension of simulating its execution at the unknown r^*. Each time a comparison is made, the few roots $r_1, r_2, \ldots$ of the associated polynomial are computed, and we run A_s at each of them, thereby determining the location of r^* among these roots, and thus also the sign of the polynomial at r^*, which determines the outcome of the comparison at r^*. If one of the r_i is equal to r^*, we stop since we have found the value of r^*, otherwise the execution of the generic A_s is resumed. As we proceed through this execution, each comparison that we resolve further constrains the range where r^* can lie, and we thus obtain a sequence of progressively smaller intervals, each known to contain r^*, until we either reach the end of A_s with a final interval I, or hit r^* at one of the comparisons of A_s. In our case, the generic algorithm will always make a comparison whose associated polynomial vanishes at r^*, which will then cause the overall algorithm to terminate with the desired value of r^*.

The cost of this implicit searching is usually dominated by $C_s T_s$, where C_s is the maximum number of comparisons executed by A_s, and T_s is the maximum running time of A_s. Since this bound is generally too high, Megiddo suggests replacing the generic A_s by a parallel algorithm, A_p. If A_p uses P processors and runs in T_p parallel steps, then each such step involves at most P *independent* comparisons (that is, each can be carried out without having to know the outcome of the others). We can then compute the roots of all P polynomials associated with these comparisons, and run a binary search to find the location of r^* among them using the serial algorithm A_s at each binary search step. This requires $O(P + T_s \log P)$ time per parallel step, for a total of $O(PT_p + T_s T_p \log P)$ time, which often results in a saving of nearly an order of magnitude in the running time. A further improvement by Cole [Co] can in certain cases reduce the running time by another logarithmic factor.

2.2. Fixed-Radius Problem. To apply Megiddo's technique to the two-center problem, we need an efficient procedure that, given an $r > 0$, can determine whether r is greater than, equal to, or less than r^*, the minimum radius for which two disks of that radius whose union covers S exists.

Hershberger and Suri [HS] have presented an algorithm that, given a set S of n points and a real parameter r, determines whether S can be covered by the union of two disks of radius r. The running time of the algorithm is $O(n^2 \log n)$. Their algorithm thus determines whether $r < r^*$ or $r \geq r^*$. Moreover, with a few obvious modifications, we can also disambiguate between the cases $r > r^*$ and $r = r^*$. We leave the easy details to the reader. To recap, we can determine, in $O(n^2 \log n)$ time, whether r is greater than, equal to, or less than r^*.

2.3. Generic Algorithm. Next, we need a parallel algorithm for the fixed-radius problem to simulate it as the generic algorithm. Unfortunately, we could not come up with an efficient parallel procedure for this problem, since it seems to require an inherently sequential search over an arrangement of circles in the plane.

To overcome this difficulty, we use the following, quite different algorithm as the one to be simulated generically. It receives as input a radius r and computes a certain measure $\varphi(r)$ that is defined as follows. For each pair of points, $p, q \in S$, at distance at most $2r$, draw the two *open* disks of radius r whose bounding circles pass through p and q; if $pq = 2r$, then only one such disk exists, but in our application this case does not arise—see below. Note that multiple copies of the same disk can arise in this manner. We thus obtain a multiset $\mathscr{D}$ of $O(n^2)$ open disks. We wish to count how many pairs of the form (D, p) are there, where $D \in \mathscr{D}$ and $p \in D \cap S$. Let us denote the resulting quantity by $\varphi(r)$.

A parallel procedure A_p for computing $\varphi(r)$ can be easily derived by a slight modification of the procedure given in [AASS]; it uses $O(n^2)$ processors and takes $O(\log n)$ parallel time in the PRAM–CRCW model. For the sake of completeness here is a brief sketch of the procedure. It first dualizes each point of S to a disk of radius r centered at the point, and dualizes each disk in $\mathscr{D}$ to its center. It next forms the arrangement of the n dual disks, preprocesses the arrangement for fast point location, and also computes how many disks cover each face of the arrangement. Then each of the dual points is located in the arrangement, and the number of disks covering the face in which the point lies is added to an overall count to yield $\varphi(r)$. All these steps can be efficiently parallelized, as explained in [AASS].

2.4. Overall Algorithm. Let us now simulate A_p at the unknown r^*, as described above. However, each time we need to compare r^* with some concrete r, we run the modified Hershberger–Suri algorithm, which is capable of making such a decision.

This approach is somewhat unusual, because there appears to be little connection between the simulated algorithm and the one that guides the binary search—only the second one appears to be relevant to our problem. However, as we see shortly, this still works correctly.

To simplify the algorithm and its analysis, we augment it with a preliminary stage that disposes of the case in which r^* is half the distance between a pair of points in S. Specifically, we list all $O(n^2)$ such half-distances, sort them, and run a standard binary search through them, using the above procedure to guide the search. This produces, in time $O(n^2 \log^2 n)$, an initial (open) interval I_0 that

contains r^* and which does not contain any value that is half the distance between any pair of points in S. This allows us, when running the generic A_p, to resolve comparisons of some r with r^* in constant time when r is not in I_0.

The subsequent generic simulation thus maintains an interval I, starting with I_0, and shrinks it as comparisons are resolved. I is always guaranteed to contain r^*. We claim that the algorithm always makes a comparison at $r = r^*$, and therefore we know the value of r^* by the time A_p stops. Before proving this claim, we need to prove an auxiliary lemma.

LEMMA 2.1. *If r^* is not half the distance between a pair of points in S, then $\varphi(r)$ strictly increases as r increases from values slightly smaller than r^* to values slightly larger than r^*.*

PROOF. Fix two values of r,

$$r^- = r^* - \varepsilon, \qquad r^+ = r^* + \varepsilon,$$

where ε is a sufficiently small positive number, so that, for all $r^- \le r \le r^+$, other than r^* itself, neither any two points of S are at distance $2r$ apart, nor is there a circle of radius r passing through three or more points of S.

We claim that $\varphi(r^-) < \varphi(r^+)$. It is clear that r^* is the only value in the interval (r^-, r^+) at which φ can change. Since we have assumed that r^* is not equal to half the distance between any pair of points in S, it follows that the *only* disk-point pairs (D, p) that may be counted in $\varphi(r^-)$ but not in $\varphi(r^+)$ or vice versa are induced by triples (p, q, t) of points in S such that the circumcircle of the triangle pqt has radius r^* and D is one of the disks whose bounding circle passes through q and t.

Let pqt be such a triangle. We consider its contribution to $\varphi(r^-)$ and to $\varphi(r^+)$. If pqt is an acute triangle, then it is easily checked that its contribution to $\varphi(r^-)$ is 0 (no circle of radius r^- that passes through two of these points can contain the third point), and that its contribution to $\varphi(r^+)$ is 3 (namely, there is exactly one open disk of radius r^+ whose bounding circle passes through q and t and contains p, one disk through p and t contains q, and one disk through p and q contains t). If pqt is an obtuse triangle, then it contributes 2 to $\varphi(r^-)$ and 3 to $\varphi(r^+)$, as is easily checked. (This is illustrated in Figure 2: the left portion shows two disk-point containments induced by this triangle at r^-; one of these pairs is eliminated at r^+, as shown in the right portion of the figure, but two new pairs arise, corresponding to the disks whose bounding circles pass through p and q and through p and t.) Our assumptions on r^* rule out the case where pqt is a right-angle triangle.

Since the contribution of each triangle whose circumcircle has radius r^* strictly increases, and there is at least one such triangle, it follows that $\varphi(r^-) < \varphi(r^+)$. □

LEMMA 2.2. *The generic algorithm A_p always makes a comparison at $r = r^*$.*

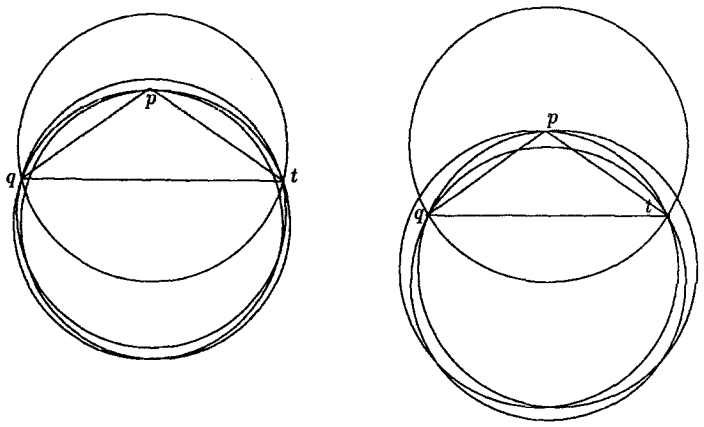

Fig. 2. The effect of an obtuse triangle pqt.

PROOF. Suppose to the contrary that A_p never makes a comparison at $r = r^*$. Then the generic algorithm stops with a final (open) interval I_f that contains r^*. It is easy to see that A_p will return the same value of $\varphi(r)$ for all $r \in I_f$. However, by Lemma 2.1, $\varphi(r)$ increases from values slightly smaller than r^* to values slightly larger than r^*, a contradiction. Hence, A_p must have made a comparison at $r = r^*$. □

We can now conclude

THEOREM 2.3. *The planar two-center problem can be solved in time* $O(n^2 \log^3 n)$.

REMARK 2.4. Our algorithm shows that the generic algorithm has very little to do with the original problem—it just simulates the solution of a completely different problem. Suppose the generic algorithm is computing a function $\zeta(r)$, then Lemma 2.2 continues to hold as long as $\zeta(r)$ changes at r^*. The important message of our variant technique is that there is hope in applying Megiddo's technique efficiently even when there is no known efficient parallel algorithm for the problem $\mathscr{P}(r)$.

3. The Two-Line-Center Problem. We next present an $O(n^2 \log^5 n)$ algorithm for the two-line-center problem. An equivalent formulation of this problem is to partition S into two subsets such that their maximum *width* is as small as possible. Recall that the width of a set is the smallest distance between a pair of parallel supporting lines.

As in the previous section, we first need an "oracle" that, given S and a width w, determines whether S can be covered by two strips of width w. Hershberger and Suri do not consider this particular problem, which indeed appears to be considerably more difficult than the fixed-radius problem that they solve. We describe an $O(n^2 \log^3 n)$ algorithm for the fixed-width problem, based on the

following off-line version of the problem of dynamic maintenance of the width of a set of points: given a real parameter $w > 0$ and a sequence $\Sigma = (\sigma_1, \ldots, \sigma_n)$ of insert/delete operations on a set S of points in $\mathbb{R}^2$, determine whether there is an i such that the width of S after the ith operation is at most w. An $O(n \log^3 n)$ algorithm for this problem is given in a companion paper [AS].

3.1. Fixed-Width Problem. In this subsection we describe the overall structure of our fixed-width algorithm. Following the general approach of [HS], we proceed as follows. Dualize the points of S to obtain a set $\mathcal{L} = \{l_1, \ldots, l_n\}$ of n lines. Suppose that one of the strips that we seek lies between the lines $y = ax + b_1$ and $y = ax + b_2$. We can assume that one of these lines, say $y = ax + b_1$ passes through a point of S, so its dual point lies on a line of $\mathcal{L}$. If the width of the strip is w, then we must have

$$|b_2 - b_1| = w\sqrt{1 + a^2}.$$

Thus the line $y = ax + b_2$ dualizes to one of the points $(a, b_1 \pm w\sqrt{1 + a^2})$. We only consider the top point, since the bottom point can be handled in a symmetric fashion.

Fix one of the dual lines, say l, and consider a point $v = (a, b) \in l$. We need to compute the subset of $\mathcal{L}$ that cross the vertical segment vv', where $v' = (a, b + w\sqrt{1 + a^2})$. All these lines are dual to points of S that lie in the strip (of width w) bounded between the lines dual to v and v' (see Figure 3). Let S_v denote this subset of S. It remains to check whether $S - S_v$ has width at most w.

Let us now move v continuously along l, and "drag" v' above it so that they keep obeying the relationship stated above. Thus v' traces a branch of a hyperbola. The set S_v changes during this motion at abscissae of either intersection points of l and the lines of $\mathcal{L}$, or points at which the hyperbola meets one of the lines of $\mathcal{L}$, and then the corresponding point of S is added or removed from S_v as appropriate. It follows that there are only $O(n)$ such local changes in S_v. Let $\sigma_1, \ldots, \sigma_k$, $k = O(n)$, be the sequence of the changes, each corresponding to inserting or deleting a point from S_v. This sequence can be easily computed in time $O(n \log n)$.

We thus need a mechanism that, given a real parameter w and a sequence $\sigma_1, \ldots, \sigma_k$ of insertion/deletion operations on S, enables us to determine whether

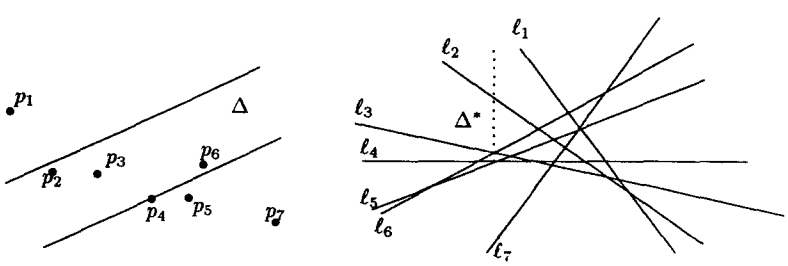

Fig. 3. Dualizing S and a vertical strip.

there is an $i \leq k$ such that the width of S after σ_i is at most w. The maintenance of the width of a set of points under insertions and deletions, even in the restrictive form just stated, turns out to be a rather difficult problem. As mentioned above, we present an $O(k \log^3 k)$ solution for this problem in a companion paper [AS]. It now follows that we can process l in time $O(n \log^3 n)$. Repeating this for all lines of $\mathscr{L}$, we obtain

THEOREM 3.1. *Given a set S of n points in $\mathbb{R}^2$ and a real parameter w, we can determine, in time $O(n^2 \log^3 n)$, whether two strips, each of width w, whose union covers S, exist.*

REMARK 3.2. The above algorithm thus determines whether $w \geq w^*$ or $w < w^*$, where w^* is the minimum width for which two strips of that width whose union covers S exist. However, with some obvious modifications, we can also disambiguate between the cases $w > w^*$ and $w = w^*$. We leave the easy details to the reader. Hence, we can determine in $O(n^2 \log^3 n)$ time, whether w is greater than, equal to, or less than w^*.

3.2. Overall Algorithm. Having this oracle at hand, the two-line-center algorithm proceeds in much the same way as the one given in the previous section. As above, we use a completely different problem to simulate generically (in parallel) at the unknown width w^*. The problem is: given the set S and a parameter w, draw for each pair of points $x, y \in S$ an *open* strip of width w whose bottom line passes through x and y, and a similar open strip whose top line passes through x and y. We get a collection of $O(n^2)$ strips (perhaps appearing with multiplicities), and our goal is to count $\psi(w)$, the number of pairs (p, Δ), where Δ is one of the strips and $p \in S \cap \Delta$ (see Figure 4).

This problem is relatively easy to solve. We dualize the points of S to lines $l_1, \ldots, l_n$ as above, and dualize each of the strips into an open vertical line segment. The problem becomes that of computing the total number of line-segment intersections. To do so we compute the entire arrangement $\mathscr{A}$ of the lines l_j, and store in each of its faces the number of lines passing below it, including those that appear on its lower boundary. We process $\mathscr{A}$ for fast point location, and then, for each vertical segment e, we locate the endpoints of e in $\mathscr{A}$ and subtract the

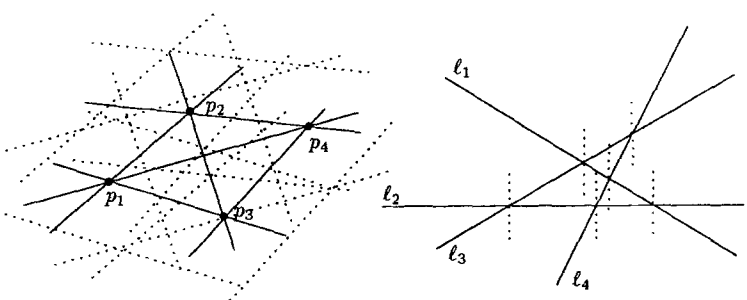

Fig. 4. Illustration of the generic algorithm.

count of the bottom face from that of the top face, to compute how many lines cut e. Summing over all vertical segments, we obtain $\psi(w)$.

To obtain a parallel version of this procedure, we can proceed in much the same way as in [AASS]; we refer the reader to this paper for more details. The resulting parallel algorithm requires $O(n^2)$ processors and $O(\log n)$ time on a PRAM–CRCW machine.

Finally, using the same argument as in Lemma 2.1 it can be proved that

LEMMA 3.3. *The value of $\psi(w)$ strictly increase as w increases from values slightly smaller than w^* to values slightly larger than w^*.*

Hence we can apply Megiddo's technique in the same manner as in the previous section. We obtain an algorithm that runs in $O(n^2 \log^5 n)$ time, leading to

THEOREM 3.4. *The planar two-line-center problem can be solved in time $O(n^2 \log^5 n)$.*

4. Discussion and Open Problems. In this paper we presented efficient (near quadratic) solutions to two "two-center" problems, one seeking to cover a set S of n points in the plane by two congruent disks of smallest possible radius, and the other seeking to cover S by two congruent strips of smallest possible width. In both cases our solution applies a rather unusual variant of Megiddo's parametric searching technique. This technique relies on the existence of an efficient solution to the "fixed-radius" (or "fixed-width") version of the problem, in which the radius of the disks is given and we need to determine whether S can be covered by two disks with that radius (and similarly for the fixed-width version). Such solutions in turn rely on efficient techniques to determine whether the smallest enclosing circle (in the first problem) or the width (in the second one) of a set of points in the plane ever becomes smaller than some specified threshold value, as the set undergoes a sequence of insertions and deletions. The solution for the latter width problem is more involved and is presented in a companion paper [AS], while the case of the smallest enclosing circle is simpler, and is discussed in [HS].

The solutions for the fixed-radius and fixed-width problems are fairly "modular" and can therefore be generalized in many ways. For example, we can solve variants in which we want to cover S by a disk and a strip, with, say some prespecified ratio between the radius of the disk and the width of the strip, or cover S by two disks with such a prespecified ratio between their radii, etc. We can also replace the disks in the two-center problem by any reasonable shape, provided we only allow translations (and of course also scaling) of that shape, but no rotation. Generally speaking, it seems that our method should apply to any situation in which each of the shapes with which we want to cover S is allowed only two degrees of freedom (plus a third degree of scaling) in its possible placements over S.

There are many open problems that this paper raises. One is to extend our techniques so as to allow the covering objects three degrees of freedom (plus scaling) in their placements. For example, how fast can we find the smallest a so

that S can be covered by two congruent copies of, say, a square of side a, or an equilateral triangle of side a, etc. Even the case of covering S by just a single such shape is not simple (see [Ch] and [CK]).

A second problem is to extend our solution to solving three-center problems of various types, say, that in which we want to cover S by three congruent disks of smallest possible radius. Can this be done in time close to $O(n^3)$? Can the general p-center problem be solved in time close to $O(n^p)$? Another interesting generalization is to three dimensions: how fast can we find the smallest radius a so that a given set of n points in 3-space can be covered by two balls of radius a? Yet another problem is the *two-median problem* [D], in which we seek two disks covering a set of n points in the plane so that the sum of the radii of the disks is minimized. A corresponding two-line median problem can be formulated analogously. Can these problems also be solved in close to quadratic time?

Finally, there is the issue of finding other problems for which one can apply the parametric searching technique of Megiddo but the underlying algorithm is not easy to parallelize, so that our variant becomes applicable. We mention one problem that seems to be amenable to this technique: given a set S of n points in the plane, and a sequence of n insertions and deletions of points into/from S, compute the smallest width that S assumes during these updates.

Acknowledgments. The authors would like to thank Michal Ben-Eli for introducing the two-center problem to us and for helpful discussions. Thanks are also extended to Arie Tamir for helpful discussions and information concerning the problem, and to Subhash Suri and John Hershberger for information concerning their procedure and related work.

References

[AASS] P. Agarwal, B. Aronov, M. Sharir, and S. Suri, Selecting distances in the plane, *Algorithmica* **9** (1993), 495–514.

[AS] P. Agarwal and M. Sharir, Off-line dynamic maintenance of the width of a planar point set, *Comput. Geom. Theory Appl.* **1** (1991), 65–78.

[Ch] B. Chazelle, The polygon containment problem, in *Advances in Computer Science*, Vol. 1 (F. P. Preparata, ed.), JAI Press, Greenwich, CT, 1983, pp. 1–33.

[CK] P. Chew and K. Kedem, Placing the largest similar copy of a convex polygon among polygonal obstacles, *Comput. Geom. Theory Appl.* **3** (1993), 59–89.

[Co] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. Assoc. Comput. Mach.* **31** (1984), 200–208.

[D] Z. Drezner, The planar two-center and two-median problems, *Transportation Sci.* **18** (1984), 351–361.

[FG] T. Feder and D. H. Greene, Optimal algorithms for approximate clustering, *Proc. 20th ACM Symp. on Theory of Computing*, 1988, pp. 434–444.

[HS] J. Hershberger and S. Suri, Finding tailored partitions, *J. Algorithms* **12** (1991), 431–463.

[LW] D. T. Lee and Y. Wu, Geometric complexity of some location problems, *Algorithmica* **1** (1986), 193–211.

[M1] N. Megiddo, Applying parallel computation algorithms in the design of serial algorithms, *J. Assoc. Comput. Mach.* **30** (1983), 852–865.

[M2] N. Megiddo, Linear-time algorithms for linear programming in $\mathbb{R}^3$ and related problems, *SIAM J. Comput.* **12** (1983), 759–776.

[Me] S. G. Mentzer, Lower bounds on metric k-center problems, Manuscript, 1988.

[MS] N. Megiddo and K. Supowit, On the complexity of some common geometric location problems, *SIAM J. Comput.* **13** (1984), 182–196.