Geometric Applications of a Randomized Optimization Technique*

Timothy M. Chan

Department of Mathematics and Computer Science
University of Miami, Coral Gables, FL 33124–4250, USA
tchan@cs.miami.edu

March 19, 1999

Abstract

We propose a simple, general, randomized technique to reduce certain geometric optimization problems to their corresponding decision problems. These reductions increase the expected time complexity by only a constant factor and eliminate extra logarithmic factors in previous, often more complicated, deterministic approaches (such as parametric searching). Faster algorithms are thus obtained for a variety of problems in computational geometry: finding minimal k-point subsets, matching point sets under translation, computing rectilinear p-centers and discrete 1-centers, and solving linear programs with k violations.

1 Introduction

Consider the classic randomized algorithm for finding the minimum of r numbers $\min\{A[1], \ldots, A[r]\}$:

Algorithm RAND-MIN
1. randomly pick a permutation $\langle i_1, \ldots, i_r \rangle$ of $\langle 1, \ldots, r \rangle$
2. $t \leftarrow \infty$
3. for $k = 1, \ldots, r$ do
4. if $A[i_k] < t$ then
5. $t \leftarrow A[i_k]$
6. return t

By a well-known fact [27, 44], the expected number of times that step 5 is executed is given by the harmonic number $1 + 1/2 + \cdots + 1/r \leq \ln r + 1$. Imagine that the numbers $A[1], \ldots, A[r]$ are not pre-computed but are evaluated only "on demand." If a decision $A[i] < t$ can be made in D time for a given t but an evaluation of $A[i]$ takes E time, then algorithm RAND-MIN runs in $O(Dr + E \log r)$ expected time, which is an improvement over the obvious $O(Er)$ worst-case bound when $D \ll E$.

This simple observation suggests a general strategy to solve a given optimization problem: express the solution as a minimum of the solutions of several subproblems and apply the above algorithm to find the minimum. If the decision versions of the subproblems are easier to solve than the

*A preliminary version of this work appeared in *Proc. 14th ACM Sympos. Comput. Geom.*, 1998.

subproblems themselves, then a faster algorithm for the optimization problem may be obtained with randomization.

In the next section, we formulate the idea more precisely and propose a recursive version of the algorithm in which we use a constant number r of subproblems at each node of the recursion. This recursive algorithm may be regarded as a generalization of prune-and-search. Assuming that the size of the subproblems is reduced by a constant factor at each level, we show that the optimization problem can be solved within the same asymptotic expected time bound as the decision problem. Previous reduction of the optimization problem to the decision problem is usually obtained by some kind of binary or parametric search, which increases the running time by a polylogarithmic factor.

1.1 New Results

This simple technique is applicable to a surprisingly diverse range of geometric optimization problems. Besides easily re-deriving some old results on closest-pair-type and ray-shooting problems (Section 3), we are able to prove a number of new ones (Sections 4 and 5), including the following:

- The minimum L_∞-diameter k-point subset of a planar n-point set can be found in $O(n \log n)$ expected time for any $2 \le k \le n$. The best previous algorithm for general k was by Eppstein and Erickson [35] and took $O(n \log^2 n)$ time.
- The minimum L_∞-Hausdorff distance between two planar n-point sets under translations can be found in $O(n^2 \log n)$ expected time. This problem was studied by Chew and Kedem [20], who gave an $O(n^2 \log^2 n)$-time algorithm.
- The two-dimensional rectilinear 5-center problem can be solved in $O(n \log n)$ expected time. Recent works [46, 62, 66] gave an $O(n \log^2 n)$ time bound.
- The two-dimensional linear programming problem with k violated constraints can be solved in $O(n \log n)$ expected time in the feasible case for any $0 \le k \le n$. (If k is not too large, the time bound is actually linear.) This type of problem was considered by Matoušek [51] and Roos and Widmayer [60]; the previous time bound for this version for arbitrary k was $O(n \log^2 n)$.
- Given an infeasible two-dimensional linear program with n constraints, the smallest number k of violations that make it feasible can be found in $O(nk)$ expected time. This speeds up an algorithm of Everett *et al.* [36] by a $\log k$ factor.
- The three-dimensional Euclidean discrete 1-center problem can be solved in $O(n \log n)$ expected time. Deterministic techniques yield an $O(n \operatorname{polylog} n)$ running time only.

The above list of problems is not meant to be exhaustive. Rather, it is used to illustrate the various ways in which the technique can be applied. We expect that more applications will follow.

When is the technique applicable? The first prerequisite is an efficient algorithm for the decision version of the problem at hand (is the optimal value smaller/larger than a given number?). Secondly, we should be able to "decompose" the problem into a constant number of subproblems of the same type but with a fraction of the size. This usually means that the number of variables (degrees of freedom) in the optimization must be a constant. It also implies that the problem is somehow "self-reducible." In many cases, we are thus forced to consider a generalized form of the original problem with added special constraints. (Rectilinear applications are often more amenable, but certain non-rectilinear ones currently pose technical difficulties.) Note that unlike in traditional prune-and-search methods (e.g., [31, 55, 56]), we do not need an oracle to identify which subproblem contains the actual solution and requires recursion; our randomized technique will guide us to the right subproblem quickly, with the aid of the decision algorithm.

1.2 Previous Approaches

One of the most general approaches for reducing geometric optimization problems to their decision problems is *parametric search*, invented by Megiddo [54] (see [1, 4, 7, 12, 15, 17, 26, 34, 53, 59, 61, 64] for just a partial list of examples). The basic idea is to simulate the decision algorithm—compare the optimum with t—with the parameter t being the unknown optimum itself. In most instances, the branching points of the simulation require testing the signs of low-degree polynomials in t, which reduces to comparing t with the roots of these polynomials. These comparisons can be resolved by making ordinary calls to the decision algorithm. In order to lower the number of such calls, we need an efficient parallelization of the simulated decision algorithm, so that comparisons can be "batched." Running time typically increases by logarithmic factors, even when an improvement by Cole [25] is applicable. As many researchers have commented, the resulting algorithms tend to be complicated and impractical; see the survey by Agarwal and Sharir [3]. In contrast, our randomized reductions use the decision algorithms purely as "black boxes," avoid the extra logarithmic factors in the running time, and are easier to implement.

A number of alternatives to parametric search have been proposed in the geometry literature [3]. First, if the search space has linear size, then an ordinary binary search is sufficient. For many rectilinear problems, the search space forms a *matrix with sorted rows/columns*, and one can use Frederickson and Johnson's selection algorithm [37] to carry out the binary search [20, 39, 66]; that algorithm relies heavily on repeated weighted-median computations. In other instances, one can employ nontrivial explicit constructions of *expander graphs* (e.g., [45]) following a technique of Katz and Sharir [42, 43]. Without additional ideas, all of these techniques increase the running time by at least a logarithmic factor.

Randomized techniques have also been suggested as an alternative in several isolated cases [2, 10, 13, 15, 29, 34, 47, 50]. Unfortunately, since these techniques are not unified and as straightforward to apply as parametric search, potential applications are sometimes missed. (One important exception though is the class of *LP-type* optimization problems [65], where general randomized linear-time solutions have been developed; most of the problems considered in this paper do not fit into this class.) We hope that our randomized technique will partially rectify this shortcoming.

One other alternative to parametric searching that has been noted in the literature [11] involves the application of *geometric cutting* tools. Some of these tools (in a very elementary form) will be of use in several of our own randomized reductions for non-rectilinear problems.

In Section 6, we discuss limitation of our technique and some issues regarding possible derandomization.

2 The Technique

We describe our technique in a general setting. Let Π represent the *problem space*. Given a problem $P \in \Pi$, let $w(P) \in \mathbb{R}$ be its *solution*. Denote the *size* of P by $|P|$ (a positive integer). We assume that the solution of a problem of constant size can be computed in constant time. The simple lemma below states in exact terms the technique in its entirety.

Lemma 2.1 *Let $\alpha < 1$, $\varepsilon > 0$, and r be constants, and let $D(\cdot)$ be a function such that $D(n)/n^{\varepsilon}$ is monotone increasing in n. Given any problem $P \in \Pi$, suppose that within $D(|P|)$ time,*

(i) *we can decide whether $w(P) < t$ for any given $t \in \mathbb{R}$, and*

(ii) *we can construct r subproblems $P_1, \ldots, P_r \in \Pi$, each of size at most $\lceil \alpha |P| \rceil$, so that*

$$w(P) = \min\{w(P_1), \ldots, w(P_r)\}.$$

Then for any problem $P \in \Pi$, we can compute the solution $w(P)$ in $O(D(|P|))$ time.

Proof: We compute $w(P)$ by applying Algorithm RAND-MIN to the (unknown) numbers $w(P_1), \ldots, w(P_r)$. Deciding $w(P_i) < t$ takes $D(|P_i|)$ time. Evaluating $w(P_i)$ is done recursively, unless $|P_i|$ drops below a certain constant. Note that this procedure not only computes $w(P)$ but identifies a constant-size subproblem attaining the minimum.

For the analysis, let $T(P)$ be the random variable corresponding to the time needed to compute $w(P)$ by this procedure. Let $N(P_i)$ be a 0-1 random variable, having value 1 if and only if $w(P_i)$ is evaluated. We have

$$T(P) \;=\; \sum_{i=1}^{r} N(P_i) T(P_i) \;+\; O(rD(|P|)).$$

As noted earlier, the expected number of evaluations by Algorithm RAND-MIN is $E[\sum_{i=1}^{r} N(P_i)] \le \ln r + 1$.

Define $T(n) = \max_{|P| \le n} E[T(P)]$. Since $N(P_i)$ and $T(P_i)$ are independent, we have

$$\begin{aligned} E[T(P)] \;&=\; \sum_{i=1}^{r} E[N(P_i)]\, E[T(P_i)] \;+\; O(rD(|P|)) \\ &\le\; (\ln r + 1)\, T(\lceil \alpha |P| \rceil) \;+\; O(rD(|P|)), \end{aligned}$$

implying a "textbook" recurrence [27]:

$$T(n) \;=\; (\ln r + 1)\, T(\lceil \alpha n \rceil) \;+\; O(D(n)).$$

If we assume

$$(\ln r + 1)\, \alpha^{\varepsilon} \;<\; 1, \tag{1}$$

then by induction one can easily show that $T(n) \le C \cdot D(n)$ for an appropriate constant C (depending on α, r, and ε).

To enforce condition (1), we compress ℓ levels of the recursion into one before applying Algorithm RAND-MIN, where ℓ is a sufficiently large constant. Then r increases to r^{ℓ} and α decreases to α^{ℓ}. To finish the proof, just note that $\lim_{\ell \to \infty} (\ln r^{\ell} + 1)\, \alpha^{\ell\varepsilon} \;=\; 0$. $\square$

Note: The above lemma still holds if (i) and (ii) require $D(|P|)$ expected time (rather than worst-case time). In all of our applications, the cost of (ii) is subsumed by the cost of (i), so $D(\cdot)$ really stands for the complexity of the decision problem.

For $r = 1$, randomization is not required: our recursive algorithm reduces to the standard prune-and-search algorithm. For a larger constant r, the worst-case running time is at least $\Omega(n^{\log r/\log(1/\alpha)})$. So the efficiency of a deterministic algorithm depends crucially on the values of the constants α and r. In contrast, by Lemma 2.1, these constants are unimportant in bounding the randomized complexity; we can thus afford a crude scheme to divide a problem into subproblems. Such a scheme is easily obtainable for certain classes of problems, as we point out in the next section; however, it may be less apparent for others.

3 Easy Applications

We now illustrate our technique in its simplest form on some abstract closest-pair and ray-shooting problems. Although no specific new results are obtained, the applications are instructive.

3.1 Closest Pairs

Let U be a collection of objects. Given a *distance function* $d : U \times U \to \mathbb{R}$, the *closest-pair problem* is to compute $w(P) = \min_{p,q \in P} d(p,q)$ for a given set $P \subset U$ of size n. The *closest-pair decision problem* is to determine whether $w(P) < t$ for a given P and $t \in \mathbb{R}$.

Theorem 3.1 *If the closest-pair decision problem can be solved in $D(n)$ time, then the closest-pair problem can be solved in $O(D(n))$ expected time, assuming that $D(n)/n$ is monotone increasing.*

Proof: Arbitrarily partition P into three subsets P_1, P_2, P_3 of roughly equal size. Then

$$w(P) \;=\; \min\{w(P_1 \cup P_2),\, w(P_1 \cup P_3),\, w(P_2 \cup P_3)\}.$$

This divides the problem into three subproblems, each of size roughly $2n/3$. Now apply Lemma 2.1 (with $r = 3$ and $\alpha = 2/3$). □

Note: the above theorem easily extends to finding closest b-tuples, for any constant integer b.

Rabin [58] was historically the first to apply randomized techniques to the standard closest-pair problem, where U is a fixed-dimensional space and $d(\cdot,\cdot)$ is the Euclidean metric. Clarkson and Shor in their seminal work [24] used randomization on one geometric optimization problem, the *Euclidean diameter problem* in three dimensions, where $U = \mathbb{R}^3$ and $d(\cdot,\cdot)$ is the negated Euclidean distance (the objective is to find the farthest pair of a point set). Agarwal and Sharir [2] considered a certain closest-pair problem that arises in finding the width of a point set in $\mathbb{R}^3$ and in other problems; here, the elements of U are bichromatic lines in $\mathbb{R}^3$ under a particular distance function (where $d(p,q) = \infty$ when p and q have the same color). In all three papers, randomized algorithms are obtained by modifying certain implicit decision algorithms. (Incidentally, Clarkson and Shor's approach, which was later adopted by Agarwal and Sharir, may be regarded as some kind of randomized prune-and-search.) Our technique gives alternative randomized algorithms that unify these previous results in a simple way and separate the decision component more clearly.

3.2 Ray Shooting

Let U be a collection of objects and V be a collection of rays. Let $\tau : U \times V \to \mathbb{R}$ be an *ordering function*, where $\tau(p_1, q) < \tau(p_2, q)$ means that ray q hits object p_1 before p_2. The *ray-shooting problem* is to preprocess a given set $P \subset U$ of size n into a data structure that answers queries of the following type: given $q \in V$, compute $w(P, q) = \min_{p \in P} \tau(p, q)$. (The object p that attains this minimum represents the first object hit by the query ray q.) In the *ray-shooting decision problem*, a query has the type: given any $q \in V$ and $t \in \mathbb{R}$, determine whether $w(P, q) < t$.

Theorem 3.2 *If the ray-shooting decision problem can be solved with $P(n)$ preprocessing and $D(n)$ query time, then the ray-shooting problem can be solved with $O(P(n))$ preprocessing and $O(D(n))$ expected query time, assuming that $P(n)/n^{1+\varepsilon}$ and $D(n)/n^{\varepsilon}$ are monotone increasing for some constant $\varepsilon > 0$.*

Proof: In the preprocessing, partition P into two subsets P_1, P_2 of roughly equal size, build the decision data structures for P_1 and P_2, and recursively preprocess P_1 and P_2. The new preprocessing time $P'(n)$ satisfies the recurrence

$$P'(n) = 2P'(n/2) + O(P(n)),$$

which solves to $P'(n) = O(P(n))$ if $P(n)/n^{1+\varepsilon}$ is monotone increasing. To compute $w(P, q)$ for a given $q \in V$, we can divide the problem into two subproblems, each of size roughly $n/2$: $w(P, q) = \min\{w(P_1, q), w(P_2, q)\}$. Now we can apply Lemma 2.1. $\square$

Note: if we only assume that $P(n)/n$ is monotone increasing in the above theorem, then the preprocessing time is $P'(n) = O(P(n) \log n)$.

Previously, Agarwal and Matoušek [1] described a general deterministic reduction of the above ray-shooting problem to the decision problem (which they called *segment emptiness*). The reduction uses parametric search and is not likely to be practical, besides increasing the query time by a polylogarithmic factor and requiring a parallel version of the decision algorithm.

In an earlier paper [13], the author gave a randomized reduction of linear-programming queries to halfspace range reporting. This reduction, when specialized to ray shooting, yields a different approach in which the preprocessing algorithm employs random sampling. (Arya and Mount [8] noted a similar reduction in the context of nearest neighbor searching.) In contrast, in the proof of Theorem 3.2, randomization is used in the query algorithm but not the preprocessing. The random sampling approach however achieves high-probability bounds in many instances and is more susceptible to derandomization.

Schömer and Thiel [61] investigated a certain collision-detection problem that also fits into the above framework. Randomization can again be used in place of parametric search.

4 Rectilinear Applications

For many geometric problems, the process of dividing a problem into subproblems is usually more involved than the simple applications from the previous section. We take a number of specific problems in the rectilinear plane and demonstrate how this division can be accomplished in each case. In what follows, all squares and rectangles are implicitly assumed to be axis-parallel. Given two rectangles R_1, R_2, we let $R_1 \vee R_2$ denote the smallest rectangle enclosing their union.

4.1 Minimal k-Point Subsets

Motivated by applications in clustering and statistical analysis, a number of researchers [6, 10, 28, 33, 35, 50] have looked at problems of the type: given an n-point set P, compute a "minimal" k-point subset. We illustrate our technique on one specific case, where the point set S is planar, and the measure of minimality is the L_∞-diameter (another case will be examined in Section 6.2). This particular problem is identical to:

> PROBLEM. Given $2 \leq k \leq n$ and an n-point set $P \subset \mathbb{R}^2$, find the smallest square enclosing at least k points of P.

Eppstein and Erickson [35] observed that the decision version of the problem can be solved in $O(n \log n)$ time by a straightforward plane-sweep algorithm. (The decision problem reduces to computing the *depth* of an arrangement of squares.) Using the search technique of Frederickson and Johnson [37], they then solved the optimization problem in $O(n \log^2 n)$ time.

Using a simple grid scheme, Datta *et al.* [28] described a general process of dividing the problem into $O(n/k)$ subproblems each of size $O(k)$, so that the solution is the minimum of the solutions of these subproblems; this division takes $O(n \log n)$ time. Immediately, the $O(n \log^2 n)$ time bound reduces slightly to $O(n \log n + (k \log^2 k)(n/k))$. If we apply Algorithm RAND-MIN to find the minimum, we get a randomized time bound $O(n \log n + (k \log k)(n/k) + (k \log^2 k) \log(n/k))$; this bound was recently observed by Bhattacharya and ElGindy [10], using a more cumbersome derivation. The randomized bound matches the $O(n \log n)$ decision time bound when $k = O(n/(\log n \log \log n))$. We give a different algorithm that runs in $O(n \log n)$ expected time for *all* values of k.

Theorem 4.1 *The smallest (axis-aligned) square containing k of n given planar points can be computed in $O(n \log n)$ expected time.*

Proof: Before applying Lemma 2.1, we find it necessary to extend the problem (and its decision problem) slightly: given an n-point set $P \subset \mathbb{R}^2$ and a rectangle R, we will compute $w(P, R, k)$, the side length of the smallest square S^* that contains at least k points of P and, in addition, contains R. The decision algorithm by Eppstein and Erickson [35] can be modified for this extended problem. Alternatively, we can directly reduce the extended problem to the original problem, since $w(P, R, k) = w(P', \emptyset, k + 4n)$, if we let P' to be the union of P with n copies of the four corner vertices of R.

Our division process is as follows. First, draw vertical lines at the $\lceil n/5 \rceil$-th smallest and $\lceil n/5 \rceil$-th largest x-coordinates of the points in P. Similarly, draw horizontal lines at the $\lceil n/5 \rceil$-th smallest and $\lceil n/5 \rceil$-th largest y-coordinates. Let R_0 be the rectangle bounded by these four lines. Write R_0 as an intersection of four halfplanes $H_1, \ldots, H_4$; see Figure 1. The optimal square S^* must belong to one of two cases:

Case 1. S^* contains R_0. Then $w(P, R, k) \;=\; w(P,\, R \vee R_0,\, k) \;=\; w(P \setminus R_0,\, R \vee R_0,\, k - |P \cap R_0|)$.

Case 2. $S^* \subset H_i$ for some $i \in \{1, \ldots, 4\}$. Then $w(P, R, k) \;=\; w(P \cap H_i, R, k)$.

In any case, one can check the identity:

$$w(P, R, k) \;=\; \min\{\, w(P \setminus R_0,\, R \vee R_0,\, k - |P \cap R_0|),\, w(P \cap H_1, R, k),\, \ldots,\, w(P \cap H_4, R, k) \,\}.$$

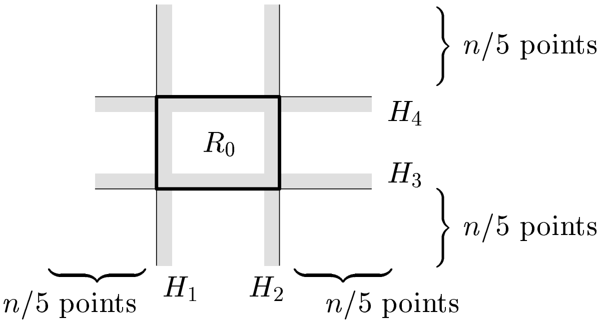

Figure 1: The rectangle R_0 and the halfplanes $H_1, \ldots, H_4$.

Each of the five subproblems $P \setminus R_0$, $P \cap H_1$, $\ldots$, $P \cap H_4$ has size at most $4\lceil n/5 \rceil$. The theorem follows. $\square$

For k not too large, the above result is probably optimal in view of the known $\Omega(n \log(n/k))$ lower bound for the so-called "k-equal problem" [67]. For k very close to n, the time bound can be further reduced to $O(n + (n-k)\log n)$; we leave the proof as an easy exercise. The technique can be extended to higher dimensions [35] or to the problem of finding the smallest homothet of a fixed convex polygon enclosing k points [33].

4.2 Matching Point Sets

Inspired by pattern recognition applications such as in computer vision, numerous papers in computational geometry (e.g., [18, 19, 20, 41]) have studied the problem of matching point sets under a class of transformations by minimizing the Hausdorff distance. We investigate a specialized case where the point sets are two-dimensional, the allowable transformations are translations, and the metric is L_∞:

> PROBLEM. Given two n-point sets $A, B \subset \mathbb{R}^2$, find vector $v \in \mathbb{R}^2$ minimizing the *directed Hausdorff distance*
> $$H(A + v, B) = \max_{a \in A} \min_{b \in B} \| a + v - b \|_\infty.$$

(Our algorithm can be easily modified if instead the *undirected Hausdorff distance* $\min\{H(A + v, B), H(B, A + v)\}$ is minimized.)

Chew and Kedem [20] showed that the decision version of the above problem can be solved in $O(n^2 \log n)$ time. Frederickson and Johnson's search technique then yields an $O(n^2 \log^2 n)$ time bound for the optimization problem. We show how to remove the extra $\log n$ factor.

Theorem 4.2 *The translation that minimizes the directed L_∞-Hausdorff distance between two planar n-point sets can be found in $O(n^2 \log n)$ expected time.*

Proof: Consider the set $P = B - A$ of $N = O(n^2)$ points. Assign colors to each point of P so that two points have the same color if and only if they belong to $B - a$ for a common $a \in A$. Let $c(p)$ denote the color of a point p and $c(P) = \{c(p) : p \in P\}$. Our problem reduces to the following if we set $k = n$:

Given a set P of N colored points in $\mathbb{R}^2$, find the smallest square S^* that contains k different colors, i.e., $|c(P \cap S^*)| \geq k$.

Chew and Kedem's algorithm solves the decision problem (finding the *color-depth* of an arrangement of squares) in $O(N \log N)$ time [20]. (Their algorithm proceeds by first partitioning the union of the squares in each color class into disjoint rectangles and then computing the depth of the arrangement of all such rectangles by a straightforward plane sweep.) We show how to solve the optimization problem in $O(N \log N)$ expected time by Lemma 2.1.

We need to extend the problem: we will compute $w(P, R, k)$, the side length of the smallest square S^* such that $|c(P \cap S^*)| \geq k$ and, in addition, contains R. In the same manner as in the previous proof, we can modify the decision algorithm or directly get rid of the extra constraint about R.

Construct the rectangle R_0 and its four bounding halfplanes $H_1, \ldots, H_4$ as in the previous proof. As before, the optimal square S^* belongs to one of two cases:

Case 1. S^* contains R_0. Then $w(P, R, k) = w(P,\, R \vee R_0,\, k) = w(P_0,\, R \vee R_0,\, k_0)$, where we let

$$P_0 = \{p \in P \setminus R_0 \,:\, c(p) \notin c(P \cap R_0)\}, \quad \text{and} \quad k_0 = k - |c(P \cap R_0)|.$$

Case 2. $S^* \subset H_i$ for some $i \in \{1, \ldots, 4\}$. Then $w(P, R, k) = w(P \cap H_i, R, k)$.

In any case, one can prove the identity:

$$w(P, R, k) = \min\{\, w(P_0,\, R \vee R_0,\, k_0),\, w(P \cap H_1, R, k),\, \ldots,\, w(P \cap H_4, R, k)\,\}.$$

Each of these five subproblems has size at most $4\lceil N/5 \rceil$. □

Chew *et al.* [18] discussed the extension of the problem to higher dimensions. Our technique again improves their running time by a logarithmic factor.

4.3 Rectilinear p-Centers

A class of facility location problems known as *p-center problems* has received much attention in the computational geometry literature (e.g., see [5, 30, 34, 64, 66]), due to applications in various areas such as operations research and clustering. We consider the case where p is a constant, the dimension is two, and the metric is L_∞:

Problem. Given an n-point set $P \subset \mathbb{R}^2$, find p congruent squares of the smallest size covering P.

The decision problem reduces to a *rectangular p-piercing problem*: given a set of n rectangles in the plane, determine whether there exists a set of p points that intersects every rectangle. Sharir and Welzl [66] described efficient algorithms for this piercing problem for $p \leq 5$. As they noted, reduction from the p-center problem to the p-piercing problem can be accomplished by Frederickson and Johnson's technique [37], which increases the running time by a factor of $\log n$. We show:

Theorem 4.3 *If the rectangular p-piercing problem can be solved in $D_p(n)$ time, then the rectilinear p-center problem can be solved in $O(D_p(n))$ expected time, where the hidden constant depends on p.*

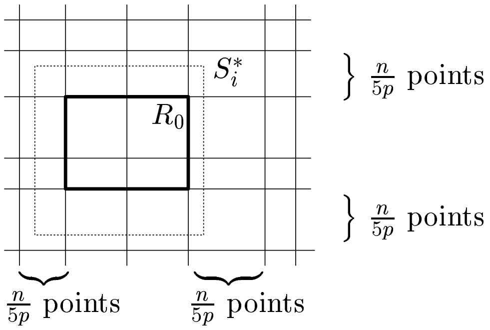

Figure 2: Grid lines and the grid rectangle R_0.

Proof: We again need to consider a generalized problem (which incidentally includes the rectilinear p-center problem with "additive weights"):

> Given a set $\mathcal{R}$ of n rectangles in $\mathbb{R}^2$, find p congruent squares $S_1^*, \ldots, S_p^*$ of the smallest size such that for each $R \in \mathcal{R}$, there exists an S_i^* that contains R.

Let $w(\mathcal{R})$ denote the side length of the optimal squares. Observe that the decision $w(\mathcal{R}) < t$ reduces to the above p-piercing problem.

The division process is a little more involved than in the previous two proofs. First list the $2n$ x-coordinates of $\mathcal{R}$ and draw a vertical line at the $i\lfloor\frac{n}{5p}\rfloor$ smallest x-coordinate for $i = 1, \ldots, 10p$. Similarly, list the $2n$ y-coordinates and draw a horizontal line the $j\lfloor\frac{n}{5p}\rfloor$ smallest y-coordinate for $j = 1, \ldots, 10p$. Call the above $O(p)$ vertical and horizontal lines *grid lines* and call a rectangle bounded by four grid lines a *grid rectangle*. Clearly, there are $O(p^4)$ possible grid rectangles.

A rectangle is said to be *δ-dense* if it contains at least δn rectangles of $\mathcal{R}$. We prove:

> Claim. One of the optimal squares S_i^* contains a $(\frac{1}{5p})$-dense grid rectangle R_0.

The claim can be seen from the following argument. Since every rectangle of $\mathcal{R}$ is contained in S_i^* for some $i \in \{1, \ldots, p\}$, there exists an S_i^* that is $(1/p)$-dense by the pigeonhole principle. Now, take the largest grid rectangle R_0 inside S_i^*, as shown in Figure 2. A rectangle of $\mathcal{R}$ that is contained in S_i^* but not in R_0 must have a vertex inside $S_i^* \setminus R_0$; by our construction of the grid, there are at most $\frac{4n}{5p}$ such rectangles of $\mathcal{R}$. Thus, R_0 is $(\frac{1}{5p})$-dense.

From the claim, we can easily show the identity

$$w(\mathcal{R}) \;=\; \min_{R_0} w(\mathcal{R} \cup \{R_0\}) \;=\; \min_{R_0}\; w(\{R \in \mathcal{R} : R \not\subseteq R_0\} \cup \{R_0\}),$$

where the minimum is taken over all $O(p^4)$ of the $(\frac{1}{5p})$-dense grid rectangles R_0. The size of each subproblem is bounded roughly by $(1 - \frac{1}{5p})n$, so we can apply Lemma 2.1. $\square$

We get one particular new result for the case $p = 5$: Sharir and Welzl [66] showed that $D_5(n) = O(n \log^4 n)$. Their $O(n \log^5 n)$ bound for the rectilinear 5-center problem can therefore be reduced to $O(n \log^4 n)$ with randomization. Recently, Segal [62] (see also [46]) improved the bound for rectangular 5-piercing to $D_5(n) = O(n \log n)$. This implies an $O(n \log n)$ randomized algorithm for rectilinear 5-centers. (According to Sharir and Welzl [66], there is an $\Omega(n \log n)$ deterministic lower bound.)

5 Non-Rectilinear Applications

As we have seen in the previous section, the division of a problem into subproblems can usually be accomplished by elementary means for rectilinear applications. For non-rectilinear problems, we often need to resort to tools for geometric divide-and-conquer known as *cuttings*.

Given a collection H of n hyperplanes in $\mathbb{R}^d$, a *δ-cutting of size s* is a partition of space into s (possibly unbounded) simplices $\{\Delta_1, \ldots, \Delta_s\}$ such that the interior of each simplex Δ_i intersects at most δn of the hyperplanes of H. We need a (rather weak) lemma on cuttings:

Lemma 5.1 *Given n hyperplanes in $\mathbb{R}^d$ for a sufficiently large n, a δ-cutting of constant size can be constructed in $O(n)$ time for some constant $\delta < 1$.*

The earliest proof of the lemma for $d = 2$ can be traced back to papers by Dyer [31] and Megiddo [55]; they design prune-and-search algorithms for three-dimensional linear programming using construction of a $(7/8)$-cutting of size 4. The construction has been extended to higher dimensions (with worse constants) [56]. A much simpler randomized method was suggested by Clarkson [21]: take a random sample $R \subset H$ of size $O((1/\delta)\log(1/\delta))$ and canonically triangulate the arrangement of R. Derandomization of this method and refinements on the constants are discussed in several papers [16, 40, 48, 52]. (For a variant known as *shallow cuttings* [49], better bounds are still possible.) Since the size of the cutting is not important when applying Lemma 2.1 (as long as it is bounded by a constant), the original method of Dyer and Megiddo or the randomized method of Clarkson is sufficient for our purposes.

5.1 Linear Programming with Violations: Feasible Case

As a first example, we consider a natural variant of the linear programming problem in which a prescribed number k of violated constraints is allowed [51, 60]. We start with the simplest nontrivial version of this problem, where the dimension is two and a feasible solution with no violation exists (without loss of generality, say that v_0 is feasible and the objective is to minimize y):

> Problem. Given $0 \le k \le n$ and a set H of n closed halfplanes in $\mathbb{R}^2$ where $v_0 \in \bigcap H$, find the lowest point in the closed region
>
> $$L_k(H) = \{q \in \mathbb{R}^2 \ : \ q \text{ violates at most } k \text{ halfplanes of } H\}.$$

(A point q *violates* a halfplane h if $q \notin h$.) Set $L_k(H) = \emptyset$ if $k < 0$. Note that for $k = 0$, this reduces to a standard linear program (which has well-known linear-time methods [22, 31, 55, 63]). The difficulty for the general case $k > 0$ lies in the fact that $L_k(H)$ (the so-called *$(\le k)$-level*) need not be convex and may have several local minima.

As Matoušek [51] and Roos and Widmayer [60] independently observed, an $O(n \log^2 n)$-time solution to the problem can be obtained by parametric search or by using an algorithm for slope selection [11, 26, 29, 42, 47]. (Roos and Widmayer also claimed an improved bound of $O(n \log n + k \log^2 k)$, but the validity of their argument is questionable.) A different approach of Matoušek [51] solves the problem in $O(n \log n + k^2 \log^2 n)$ time, which is efficient only when k is roughly smaller than $\sqrt{n}$. The $O(n \log n)$ term can be improved to $O(n \log k)$ by a technique of the author [14].

We derive here the expected time bound $O(n \log n)$ for *all* values of k. Surprisingly, if $k = O(n/\log^\beta n)$ for a constant $\beta > 1$, our bound strengthens to $O(n)$, matching the complexity of the standard linear programming problem. The precise time bound is given in the theorem below:

Theorem 5.2 *Given a set H of n halfplanes with a nonempty intersection, the lowest point in $L_k(H)$ can be computed in $O(n + k(n/k)^\varepsilon \log n)$ expected time for any constant $\varepsilon > 0$.*

Proof: To avoid some (minor) technical complications, we assume that the given halfplanes are in general position. Consider the slightly generalized problem of computing $w(H, \Delta, k)$, the smallest y-coordinate inside $L_k(H) \cap \Delta$ for a given triangle Δ. We first show how to decide whether $w(H, \Delta, k) \leq t$ for a given $t \in \mathbb{R}$. The decision $w(H, \Delta, k) < t$ can be made by a modification of the procedure below.

Let Δ' be the intersection of Δ with the halfplane $y \leq t$. We have $w(H, \Delta, k) \leq t$ if and only if $L_k(H)$ intersects Δ'. As $v_0 \in \bigcap H$, $L_k(H)$ is a connected region containing v_0. Thus, the test holds if and only if $v_0 \in \Delta'$ or $L_k(H)$ intersects one of the (at most four) edges of Δ'. Deciding whether $L_k(H)$ intersects a line segment is equivalent to the following one-dimensional problem:

> Let I be a collection of n half-infinite intervals. Given a, b, decide whether there exists a point in $[a, b]$ that is contained in all but at most k of the intervals of I.

This one-dimensional problem can be solved as follows. Write $I = \{[a_i, \infty)\} \cup \{(-\infty, b_j]\}$, where $a_1 \geq a_2 \geq \cdots$ and $b_1 \leq b_2 \leq \cdots$. One can verify that the answer is yes if and only if there exists some $i \in \{0, \ldots, k\}$ such that

$$\max\{a, a_{i+1}\} \leq \min\{b, b_{k-i+1}\}.$$

The condition can be checked in $O(k)$ time, once we have computed $a_1, \ldots, a_k$ and $b_1, \ldots, b_k$. This computation requires sorting the k smallest/largest elements in a list and takes $O(n + k \log n)$ time (for instance, by a modified heapsort).

Having solved the decision problem in $O(n + k \log n)$ time, we now proceed to solve the optimization problem. First compute a δ-cutting of the n bounding lines of H by Lemma 5.1. Intersect the cutting triangles with Δ and triangulate to form a new collection of triangles $\{\Delta_i\}$. For each Δ_i, let H_i be the set of halfplanes of H whose bounding lines intersect Δ_i and let k_i be the number of halfplanes of H not intersecting Δ_i. Then

$$w(H, \Delta, k) = \min_i w(H, \Delta_i, k) = \min_i w(H_i, \Delta_i, k - k_i).$$

This divides the problem into a constant number of subproblems each of size roughly δn. We can now apply Lemma 2.1 with the upper bound $D(n) = O(n + k(n/k)^\varepsilon \log n)$. (Note that we cannot directly set $D(n) = n + k \log n$, because $D(n)/n^\varepsilon$ needs to be monotone increasing in n.) $\square$

We leave as an open question whether the optimization problem can be solved in $O(n + k \log n)$ expected time. Extending the technique to higher dimensions is also possible, but it is unclear at the moment how much is gained by examining the decision problem.

5.2 Linear Programming with Violations: Infeasible Case

In the case that $\bigcap H = \emptyset$, our technique can be applied to solve a different problem:

> Problem. Given a set H of n halfplanes in $\mathbb{R}^2$, find the smallest k such that there exists a point in $L_k(H)$.

In other words, we want to find a maximal consistent subset of halfplanes (of size $n-k$) [32, 51]. The dual is equivalent to the following line classification problem [36, 51], with motivation from statistics: given n points in $\mathbb{R}^2$ each colored red or blue, find a line ℓ that minimizes the total number k of red points above ℓ and blue points below ℓ. The problem is also related to finding certain line transversals [36].

An $O(n^2)$-time algorithm is immediate after constructing the arrangement of the n bounding lines. Everett *et al.* [36] gave a "quality-sensitive" algorithm for this problem that runs in $O(n \log n + nk \log k)$ time. The algorithm first solves the decision problem (given k, is $L_k(H)$ empty?) in $O(n \log n + nk)$ time, and then uses a binary search to find the optimal k. The theorem below improves their $O(n \log n + nk \log k)$ running time to $O(n \log n + nk)$, answering one of their open problems. We note that when k is small (roughly less than $\sqrt{n}$), a different algorithm of Matoušek [51], with a refinement by the author [14], yields a better running time of $O(n \log k + k^3 \log^2 n)$. It is open what the optimal time bound should be for the entire range of k.

Theorem 5.3 *Suppose that we can decide whether $L_k(H) \neq \emptyset$ in $D_K(n)$ time for any collection H of n halfplanes and $k \leq K$. Then we can find the smallest $k \leq K$ with $L_k(H) \neq \emptyset$ in $O(D_K(n))$ expected time, assuming that $D_K(n)/n^\varepsilon$ is monotone increasing in n for some constant $\varepsilon > 0$.*

Proof: Given a triangle Δ, let $w(H, \Delta, K)$ be the smallest $k \leq K$ such that $L_k(H) \cap \Delta$ is nonempty (use ∞ if no such k exists). We can get rid of Δ as follows: $w(H, \Delta, K) = w(H', \mathbb{R}^2, K)$, where H' is the union of H with $n+1$ copies of the defining halfplanes of Δ. By assumption, we can decide whether $w(H, \mathbb{R}^2, K) < t$ for a given integer t in $D_K(n)$ time (if t exceeds K, the question reduces to deciding whether $L_K(H) \neq \emptyset$).

To compute $w(H, \Delta, K)$, construct a δ-cutting of constant size. Define the triangulation $\{\Delta_i\}$ of Δ and the corresponding sets of halfplanes $\{H_i\}$ and numbers $\{k_i\}$ as in the previous proof. Then

$$w(H, \Delta, K) \;=\; \min_i w(H, \Delta_i, K) \;=\; \min_i (w(H_i, \Delta_i, K - k_i) + k_i).$$

Modifying the problem format to account for the additive term, we can apply Lemma 2.1. $\square$

To use the theorem to find the optimal k, we still need an upper bound K. This can be obtained by a standard trick of "guessing" k with an increasing sequence. For instance, with Everett *et al.*'s bound $D_K(n) = O(n \log n + nK)$, we can apply our algorithm on $K = \lceil \log n \rceil, 2\lceil \log n \rceil, 4\lceil \log n \rceil, \ldots$ until $K \geq k$. The total cost remains $O(n \log n + nk)$. Of course, we can remove the $O(n \log n)$ term here by switching to the method of Matoušek for small values of k.

5.3 Discrete 1-Centers

As a final example that uses cuttings, we examine another instance of facility location, namely, the *discrete 1-center problem* in three dimensions under the Euclidean metric:

> PROBLEM. Given an n-point set $P \subset R^3$, find the smallest ball enclosing P whose center belongs to P.

It is a variant of the standard Euclidean 1-center problem, which asks for the smallest enclosing ball with no restriction on the center. (The standard version is an LP-type problem and has linear-time solutions in any fixed dimension [22, 55, 56, 66].)

Recently, the discrete 2-center problem in the plane was studied by Agarwal *et al.* [5]. As they noted, the discrete 1-center problem in the plane is solvable in $O(n \log n)$ time by constructing the farthest-point Voronoi diagram. However, for points in three dimensions, the size of the Voronoi diagram can be $\Omega(n^2)$ in the worst case. A better approach in $\mathbb{R}^3$ is to first solve the decision problem, which involves the construction of an intersection of congruent balls. This intersection has $O(n)$ size and can be computed in $O(n \log n)$ time by a randomized incremental method of Clarkson and Shor [24] or its derandomization [7, 12]. Like the Euclidean diameter problem, the discrete 1-center problem in $\mathbb{R}^3$ can then be solved by parametric search in $O(n \operatorname{polylog} n)$ time [7, 12, 17, 53, 59].

We now show how parametric search can be replaced by a randomized search, solving the three-dimensional discrete 1-center problem in $O(n \log n)$ expected time. The randomized reduction to the decision problem is more involved than for the diameter problem (Section 3.1). In particular, we need to borrow a certain prune-and-search idea that was used previously by Megiddo [55] for the standard 1-center problem.

We have learned just recently of a randomized algorithm by Clarkson [23] that solves the same problem without using Megiddo-style prune-and-search; its expected running time is however slightly slower—more precisely, $O(n \log n \log \log n)$.

Theorem 5.4 *The discrete 1-center problem in $\mathbb{R}^3$ can be solved in $O(n \log n)$ expected time.*

Proof: Given an m-point set P and an n-point set Q in $\mathbb{R}^3$, we solve a "min-max" problem: compute

$$w(P,Q) \;=\; \min_{p \in P} \max_{q \in Q} d(p,q),$$

where $d(\cdot,\cdot)$ is the Euclidean metric. The radius of the optimal ball for the discrete 1-center problem is $w(P,P)$. (Note that the "max-max" problem corresponds to the diameter problem.)

To decide whether $w(P,Q) < t$, we compute the intersection of the balls centered at the points of Q with radius t. If there exists a point of P in the interior of this intersection, then we return yes. By the algorithm of Clarkson and Shor [24], the decision can be made in $O((n+m)\log n)$ time.

Now, we apply Lemma 2.1 to solve the optimization problem. First, it is easy to reduce the size of P by a constant factor: arbitrarily partition P into two subsets P_1, P_2 of roughly size $m/2$, and observe that

$$w(P,Q) \;=\; \min\{w(P_1,Q),\, w(P_2,Q)\}. \tag{2}$$

To reduce the size of Q by a constant factor, we arbitrarily pair points of Q and construct the bisector (a plane) for each of the $\lfloor n/2 \rfloor$ pairs. By Lemma 5.1, construct a δ-cutting $\{\Delta_i\}$ of these $\lfloor n/2 \rfloor$ bisectors.

For each tetrahedron Δ_i, we build a point set Q_i. Initially, Q_i is set to Q. Examine all pairs (q_1, q_2) whose bisectors avoid the interior of Δ_i. Say q_1 lies on the same side of the bisector as Δ_i. Then any point in Δ_i is closer to q_1 than q_2. Remove q_1 from Q_i. As approximately $(1-\delta)n/2$ points are pruned, each Q_i has size bounded by roughly $(1+\delta)n/2$. Furthermore, $\max_{q \in Q} d(p,q) = \max_{q \in Q_i} d(p,q)$ for any $p \in \Delta_i$. Hence,

$$w(P,Q) \;=\; \min_i w(P \cap \Delta_i, Q) \;=\; \min_i w(P \cap \Delta_i, Q_i). \tag{3}$$

By applying equations (2) and (3) alternately, we can reduce both m and n by a constant factor using a constant number of subproblems. The result follows. □

Clearly, the same reduction extends to the discrete 1-center problem in any fixed dimension. The reduction also works for "max-min" problems, such as the computation of the Hausdorff distance of two point sets; unfortunately, in this case, the decision problem does not seem to be any easier to solve.

An $\Omega(n \log n)$ lower bound for the diameter problem is known even in the two-dimensional case [57]. Similar arguments imply the same lower bound for the discrete 1-center problem (by a reduction from the "set equality problem" [9]).

6 Discussions

We have obtained improved expected time bounds (in some cases, matching known worst-case lower bounds) for several geometric optimization problems using a common randomized technique. The technique reduces these problems to their corresponding decision problems with only a constant-factor increase in the running time. The constants are quite large though (especially for the rectilinear p-center problem), and it is of practical interest to lower them.

Take for example the closest-pair application (Section 3.1). The reduction in the proof of Theorem 3.1 partitions the given set into three subsets and as a result reduces the problem into $r = 3$ subproblems of a fraction $\alpha = 2/3$ of the original size. When applying Lemma 2.1, we need to compress $\ell = 5$ levels of the recursion, in effect, raising the number of subproblems to 3^5. A better approach is to modify the reduction directly by partitioning the given set into b subsets for a constant $b > 3$. Then we can create $\binom{b}{2}$ subproblems of $2/b$ times the size. By choosing $b = 10$, we avoid the compression and have only 45 subproblems. A similar approach can be taken to optimize the constants in some of our other applications.

We see two more directions for further research. The first is to find more geometric applications (or perhaps non-geometric ones). The second is to find deterministic algorithms matching the performance of our randomized algorithms.

6.1 Limitation

Regarding the first direction, we note two requirements in the application of our technique, as shown by many of our examples: (i) a sufficiently "robust" decision algorithm that can handle an appropriate extension of the problem, and (ii) a way to divide a problem so as to reduce the problem size by a constant factor.

Requirement (ii) needs more technical consideration but actually can be dealt with in a fairly general way using existing tools on geometric cuttings. For an illustration, take the well-studied *planar Euclidean 2-center problem*—find two congruent balls of the smallest size covering n given points in $\mathbb{R}^2$—where there is an $O(n \log n)$ solution for the decision problem and an $O(n \log^2 n)$ randomized solution for the optimization problem from the recent work of Sharir [64] and Eppstein [34] (see also [15]); parametric search was used here in one case. We can transform the problem into one involving an arrangement of n surfaces in five dimensions (using four variables for the coordinates of the two centers and one variable for the radius). Constructing a cutting for this collection of piecewise-algebraic surfaces establishes (ii). However, we need to solve an extension of the decision problem where the solution is constrained to lie inside a given Tarski cell in the transformed five-dimensional space. Unfortunately, the current $O(n \log n)$ decision algorithm does not work for this extended problem, so requirement (i) is not satisfied.

There are a number of other problems for which parametric search is applicable but our randomized technique is currently not (including problems with a nonconstant number of variables). It is hard to make a clear comparison of the power of the two different techniques, though. The decomposability requirement (ii) of our technique (see Lemma 2.1) indeed implies a special (tree-like) form of a parallel decision algorithm, much more restrictive than in parametric search. On the other hand, the processor bound of this parallel algorithm is not important in our technique, but is in parametric search. It is intriguing to explore further frameworks to widen the applicability of randomization in geometric (and non-geometric) optimization.

6.2 Derandomization?

Regarding the second direction, we point out that derandomization of our generic recursive algorithm (Section 2) is probably hard, if at all possible. We cannot even achieve good high-probability bounds. The problem lies in the fact that algorithm RAND-MIN is applied to a constant number of elements.

Consider one special case where we want to find the minimum $t^* = \min\{A[1], \dots, A[r]\}$, knowing that t^* belongs to a given search space of size U, say $\{1, \dots, U\}$. Suppose that a decision $A[i] < t$ takes D time. Then evaluating an $A[i]$ takes $O(D \log U)$ time by binary search, and algorithm RAND-MIN runs in $O(Dr + D \log U \log r)$ expected time.

Here is a simple deterministic algorithm that beats the obvious $O(Dr \log U)$ time bound for small values of U:

Algorithm DET-MIN
1. $i \leftarrow 1$
2. while $i \leq r$ do
3. if $A[i] \geq U$ then $i \leftarrow i + 1$ else $U \leftarrow U - 1$
4. return U

The correctness of the algorithm is easily seen from the invariant $\min\{A[1], \dots, A[i-1]\} \geq U$. The running time of the algorithm is clearly $O(D(r + U))$.

For larger values of $U > r$, the process can be sped up as follows. Apply Algorithm DET-MIN to find $t_0 = \min_i \lfloor A[i] \cdot (r/U) \rfloor$. As t_0 lies in $\{1, \dots, r\}$, it can be found in $O(Dr)$ time. Now, the minimum t^* lies between $t_0 U/r$ and $(t_0 + 1)U/r$, so we just need to solve the problem recursively on a search space of size U/r. The depth of the recursion is $O(\lceil \log_r U \rceil)$, so the minimum can be found in $O(Dr \lceil \log_r U \rceil)$.

The deterministic complexity of this minimization problem is therefore the same as the randomized complexity—namely, $O(Dr)$—if the size of the search space U is polynomial in r. For still larger U, one can apply a more complicated procedure due to Gao *et al.* [38].

We are only able to find one application of this deterministic approach, a variant of a problem studied in Section 4.1:

> PROBLEM. Given $2 \le k \le n$ and an n-point set $P \subset \mathbb{R}^2$, find a k-point subset with the minimum Euclidean diameter.

For k close to a fraction of n, the best algorithm for this problem is due to Eppstein and Erickson [35] and runs in $O(n^3 \log^2 n)$ time. We improve their time bound by a logarithmic factor.

Theorem 6.1 *The minimum-diameter k-point subset of a given planar n-point set can be found in $O(n^3 \log n)$ time deterministically.*

Proof: Eppstein and Erickson's approach [35] actually breaks the problem into n subproblems. The decision version of each subproblem is solved in $O(n^2 \log n)$ time by a dynamic matching algorithm, so that the decision version of the overall problem is solved in $O(n^3 \log n)$ time.

To be precise, let $B(p,t)$ denote the ball centered at p of radius t and define the following predicate given $p \in P$ and $t \in \mathbb{R}$:

> LUNE-TEST(p,t) is true if and only if there exists a point q of distance t to p such that the *lune* $B(p,t) \cap B(q,t)$ contains a k-point subset of P with diameter at most t.

Eppstein and Erickson showed that this predicate can be evaluated in $O(n^2 \log n)$ time.

Say $P = \{p_1, \dots, p_n\}$. For each i, let $A[i]$ be the smallest t such that LUNE-TEST(p_i,t) is true. The minimum diameter t^* over all k-point subsets of P is precisely $\min\{A[1], \dots, A[n]\}$, because the minimal subset must be contained in some lune of radius t^*. A decision $A[i] < t$ reduces to a predicate evaluation because of monotonicity: if $t_1 \le t_2$, then LUNE-TEST(p,t_1) implies LUNE-TEST(p,t_2).

The search space is the set of all interpoint distances, enumerable in $O(n^2 \log n)$ time. In the above setting, we have here $r = n$, $D = O(n^2 \log n)$, and $U = O(n^2)$. We can thus find t^* in $O(Dr\lceil \log_r U \rceil) = O(n^3 \log n)$ worst-case time. □

For small values of k, one can combine the above result with known techniques of Eppstein and Erickson [35] or Datta *et al.* [28] to improve the previous deterministic time bound of $O(n \log n + nk^2 \log^2 k)$ to $O(n \log n + nk^2 \log k)$ for this particular problem. The corresponding randomized result was recently discovered by Bhattacharya and ElGindy [10].

References

[1] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22:764–806, 1993.

[2] P. K. Agarwal and M. Sharir. Efficient randomized algorithms for some geometric optimization problems. *Discrete Comput. Geom.*, 16:317–337, 1996.

[3] P. K. Agarwal and M. Sharir. Efficient algorithms for geometric optimization. Technical Report CS-1996-19, Department of Computer Science, Duke University, Durham, NC, 1996.

[4] P. K. Agarwal, M. Sharir, and S. Toledo. Applications of parametric searching in geometric optimization. *J. Algorithms*, 17:292–318, 1994.

[5] P. K. Agarwal, M. Sharir, and E. Welzl. The discrete 2-center problem. *Discrete Comput. Geom.*, 20:287–305, 1998.

[6] A. Aggarwal, H. Imai, N. Katoh, and S. Suri. Finding k points with minimum diameter and related problems. *J. Algorithms*, 12:38–56, 1991.

[7] N. M. Amato, M. T. Goodrich, and E. Ramos. Parallel algorithms for higher-dimensional convex hulls. In *Proc. 35th IEEE Sympos. Found. Comput. Sci.*, pages 683–694, 1994.

[8] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 271–280, 1993.

[9] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proc. 15th ACM Sympos. Theory Comput.*, pages 80–86, 1983.

[10] B. K. Bhattacharya and H. ElGindy. Biased search and k-point clustering. In *Proc. 9th Canad. Conf. Comput. Geom.*, pages 141–146, 1997.

[11] H. Brönnimann and B. Chazelle. Optimal slope selection via cuttings. *Comput. Geom. Theory Appl.*, 10:23–29, 1998.

[12] H. Brönnimann, B. Chazelle, and J. Matoušek. Product range spaces, sensitive sampling, and derandomization. In *Proc. 34th IEEE Sympos. Found. Comput. Sci.*, pages 400–409, 1993.

[13] T. M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th ACM Sympos. Comput. Geom.*, pages 284–290, 1996.

[14] T. M. Chan. Output-sensitive results on convex hulls, extreme points, and related problems. *Discrete Comput. Geom.*, 16:369–387, 1996.

[15] T. M. Chan. More planar two-center algorithms. Manuscript, 1997.

[16] B. Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete Comput. Geom.*, 9:145–158, 1993.

[17] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir. Diameter, width, closest line pair and parametric searching. *Discrete Comput. Geom.*, 10:183–196, 1993.

[18] L. P. Chew, D. Dor, A. Efrat, and K. Kedem. Geometric pattern matching in d-dimensional space. *Discrete Comput. Geom.*, 21:257–274, 1999.

[19] L. P. Chew, M. T. Goodrich, D. P. Huttenlocher, K. Kedem, J. M. Kleinberg, and D. Kravets. Geometric pattern matching under Euclidean motion. *Comput. Geom. Theory Appl.*, 7:113–124, 1997.

[20] L. P. Chew and K. Kedem. Getting around a lower bound for the minimum Hausdorff distance. *Comput. Geom. Theory Appl.*, 10:197–202, 1998.

[21] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete Comput. Geom.*, 2:195–222, 1987.

[22] K. L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995.

[23] K. L. Clarkson. Algorithms for the minimum diameter of moving points and for the discrete 1-center problem. Manuscript, 1997.

[24] K. L. Clarkson and P. W. Shor. Applications of random sampling in computational geometry, II. *Discrete Comput. Geom.*, 4:387–421, 1989.

[25] R. Cole. Slowing down sorting networks to obtain faster sorting algorithms. *J. ACM*, 34:200–208, 1987.

[26] R. Cole, J. Salowe, W. Steiger, and E. Szemerédi. An optimal-time algorithm for slope selection. *SIAM J. Comput.*, 18:792–810, 1989.

[27] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* MIT Press, Cambridge, Mass., 1990.

[28] A. Datta, H.-P. Lenhof, C. Schwarz, and M. Smid. Static and dynamic algorithms for k-point clustering problems. *J. Algorithms*, 19:474–503, 1995.

[29] M. Dillencourt, D. Mount, and N. Netanyahu. A randomized algorithm for slope selection. *Int. J. Comput. Geom. Appl.*, 2:1–27, 1992.

[30] Z. Drezner. On the rectangular p-center problem. *Naval Res. Logist. Quart.*, 34:229–234, 1987.

[31] M. E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13:31–45, 1984.

[32] A. Efrat, M. Lindenbaum, and M. Sharir. Finding maximally consistent sets of halfspaces. In *Proc. 5th Canad. Conf. Comput. Geom.*, pages 432–436, 1993.

[33] A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. *Comput. Geom. Theory Appl.*, 4:119–136, 1994.

[34] D. Eppstein. Faster construction of planar two-centers. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 131–138, 1997.

[35] D. Eppstein and J. Erickson. Iterated nearest neighbors and finding minimal polytopes. *Discrete Comput. Geom.*, 11:321–350, 1994.

[36] H. Everett, J.-M. Robert, and M. van Kreveld. An optimal algorithm for the ($\leq k$)-levels, with applications to separation and transversal problems. *Int. J. Comput. Geom. Appl.*, 6:247–261, 1996.

[37] G. N. Frederickson and D. B. Johnson. The complexity of selection and ranking in $X + Y$ and matrices with sorted rows and columns. *J. Comput. Sys. Sci.*, 24:197–208, 1982.

[38] F. Gao, L. J. Guibas, D. G. Kirkpatrick, W. T. Laaser, and J. Saxe. Finding extrema with unary predicates. *Algorithmica*, 9:591–600, 1993.

[39] A. Glozman, K. Kedem, and G. Shpitalnik. On some geometric selection and optimization problems via sorted matrices. *Comput. Geom. Theory Appl.*, 11:17–28, 1998.

[40] S. Har-Peled. Constructing cuttings in theory and practice. In *Proc. 14th ACM Sympos. Comput. Geom.*, pages 327–336, 1998.

[41] D. P. Huttenlocher, K. Kedem, and M. Sharir. The upper envelope of Voronoi surfaces and its applications. *Discrete Comput. Geom.*, 9:267–291, 1993.

[42] M. J. Katz and M. Sharir. Optimal slope selection via expanderes. *Inform. Process. Lett.*, 47:115–122, 1993.

[43] M. J. Katz and M. Sharir. An expander-based approach to geometric optimization. *SIAM J. Comput.*, 26:1384–1408, 1997.

[44] D. E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming.* Addison-Wesley, 1968.

[45] A. Lubotzky, R. Phillips, and P. Sarnak. Explicit expanders and the Ramanujan conjectures. In *Proc. 18th ACM Sympos. Theory Comput.*, pages 240–246, 1986.

[46] C. Makris and A. Tsakalidis. Fast piercing of iso-oriented rectangles. In *Proc. 9th Canad. Conf. Comput. Geom.*, pages 217–222, 1997.

[47] J. Matoušek. Randomized optimal algorithm for slope selection. *Inform. Process. Lett.*, 39:183–187, 1991.

[48] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[49] J. Matoušek. Reporting points in halfspaces. *Comput. Geom. Theory Appl.*, 2:169–186, 1992.

[50] J. Matoušek. On enclosing k points by a circle. *Inform. Process. Lett.*, 53:217–221, 1995.

[51] J. Matoušek. On geometric optimization with few violated constraints. *Discrete Comput. Geom.*, 14:365–384, 1995.

[52] J. Matoušek. On constants for cuttings in the plane. *Discrete Comput. Geom.*, 20:427–448, 1998.

[53] J. Matoušek and O. Schwarzkopf. A deterministic algorithm for the three-dimensional diameter problem. *Comput. Geom. Theory Appl.*, 6:253–262, 1996.

[54] N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM*, 30:852–865, 1983.

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

[56] N. Megiddo. Linear programming in linear time when the dimension is fixed. *J. ACM*, 31:114–127, 1984.

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

[58] M. O. Rabin. Probabilistic algorithms. In *Algorithms and Complexity* (J. F. Traub, ed.), pages 21–30, Academic Press, New York, NY, 1976.

[59] E. Ramos. Construction of 1-d lower envelopes and applications. In *Proc. 13th ACM Sympos. Comput. Geom.*, pages 57–66, 1997.

[60] T. Roos and P. Widmayer. k-Violation linear programming. *Inform. Process. Lett.*, 52:109–114, 1994.

[61] E. Schömer and C. Thiel. Efficient collision detection for moving polyhedra. In *Proc. 11th ACM Sympos. Comput. Geom.*, pages 51–60, 1995.

[62] M. Segal. On piercing of axis-parallel rectangles and rings. In *Proc. 5th European Sympos. Algorithms*, Lect. Notes in Comput. Sci., vol. 1284, Springer-Verlag, pages 430–442, 1997.

[63] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[64] M. Sharir. A near-linear algorithm for the planar 2-center problem. *Discrete Comput. Geom.*, 18:125–134, 1997.

[65] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, Lect. Notes in Comput. Sci., vol. 577, Springer-Verlag, pages 569–579, 1992.

[66] M. Sharir and E. Welzl. Rectilinear and polygonal p-piercing and p-center problems. In *Proc. 12th ACM Sympos. Comput. Geom.*, pages 122–132, 1996.

[67] A. C.-C. Yao. Decision tree complexity and Betti numbers. *J. Comput. Sys. Sci.*, 55:36–43, 1997.