Algorithmic Techniques for Geometric Optimization*

Pankaj K. Agarwal[1] and Micha Sharir[2,3]

[1] Department of Computer Science
Box 90129, Duke University, Durham, NC 27708-0129, USA.
[2] School of Mathematical Sciences
Tel Aviv University, Tel Aviv 69978, ISRAEL.
[3] Courant Institute of Mathematical Sciences
New York University, New York, NY 10012, USA.

Abstract. We review the recent progress in the design of efficient algorithms for various problems in geometric optimization. The emphasis in this survey is on the techniques used to attack these problems, such as parametric searching, geometric alternatives to parametric searching, prune-and-search techniques for linear programming and related problems, and LP-type problems and their efficient solution.

1 Introduction

In this survey we describe several techniques that have lead to efficient algorithms for a variety of geometric optimization problems. We also list many applications of these techniques, and discuss some of them in more detail. The first technique that we present is the *parametric searching* technique. Although restricted forms of parametric searching already existed earlier [49, 57, 58], the parametric searching in its full generality was proposed by Megiddo in the late 1970's and the early 1980's [87, 88]. The technique was originally motivated by so-called *parametric optimization* problems in combinatorial optimization, and did not receive much attention by the computational geometry community until the late 1980's. In the last seven years, though, it has become one of the major techniques for solving geometric optimization problems efficiently. We outline the technique in detail in Section 2, first exemplifying it on the *slope selection problem* [36], and then presenting various extensions of the technique.

Despite its power and versatility, parametric searching has certain drawbacks, which we discuss below. Consequently, there have been several recent attempts to replace parametric searching by alternative techniques, including *randomization* [11, 33, 79], *expander graphs* [16, 70, 71, 72], *geometric cuttings* [12, 22], and

*Pankaj Agarwal has been supported by National Science Foundation Grant CCR-93–01259, an NYI award, and by matching funds from Xerox Corp. Micha Sharir has been supported by NSF Grants CCR-91-22103 and CCR-93-11127, by a Max-Planck Research Award, and by grants from the U.S.-Israeli Binational Science Foundation, the Israel Science Fund administered by the Israeli Academy of Sciences, and the G.I.F., the German-Israeli Foundation for Scientific Research and Development.

matrix searching [50, 51, 52, 53, 55]. We mention these alternative techniques in Section 3.

Almost concurrently with the development of the parametric searching technique, Megiddo devised another ingenious technique for solving linear programming and several related optimization problems [89, 90]. This technique, now known as *decimation* or *prune-and-search*, was later refined and extended by Dyer [41], Clarkson [30], and others. The technique can be viewed as an optimized version of parametric searching, in which certain special properties of the problem allows one to improve further the efficiency of the algorithm. For example, it yields linear-time deterministic algorithms for *linear programming* and for several related problems, such as the *smallest enclosing ball* problem, when the dimension is fixed. (However, the dependence of the running time of these algorithms on the dimension is at least exponential.) We present the technique and its applications in Section 4. Section 5 enumerates many recent applications of the techniques reviewed so far.

In the past decade, *randomized algorithms* have been developed for a wide variety of problems in computational geometry and in other fields; see, e.g., the books by Mulmuley [96] and by Motwani and Raghavan [95]. In particular, Clarkson [31] and Seidel [103] gave randomized algorithms for linear programming, whose expected time is linear in any fixed dimension, which are much simpler than their earlier deterministic counterparts, and the dependence of their running time on the dimension is better (though still exponential). Additional significant progress was made about four years ago, when new randomized algorithms for linear programming were obtained independently by Kalai [68], and by Matoušek et al. [86, 110] (these two algorithms are essentially dual versions of the same technique). The expected number of arithmetic operations performed by these algorithms is 'subexponential' in the input size, and is still linear in any fixed dimension, so they constitute an important step toward the still open goal of obtaining strongly-polynomial algorithms for linear programming. (Recall that the polynomial-time algorithms by Khachiyan [73] and Karmarkar [69] are not strongly polynomial, as the number of arithmetic operations performed by these algorithms depends on the size of coefficients of the input constraints.) This new technique is presented in Section 6. The algorithm in [86, 110] is formulated in a general abstract framework, which fits not only linear programming but many other problems. Such 'LP-type' problems are also reviewed in Section 6, including the connection, recently noted by Amenta [19, 20], between abstract linear programming and 'Helly-type' theorems.

2 Parametric Searching

2.1 Outline of the Technique

The parametric searching technique of Megiddo [87, 88] can be described in the following general terms (which are not as general as possible, but suffice for our purposes). Suppose we have a decision problem $\mathcal{P}(\lambda)$ that depends on a real parameter λ, and is *monotone* in λ, meaning that if $\mathcal{P}(\lambda_0)$ is true for some λ_0, then $\mathcal{P}(\lambda)$ is true for all $\lambda < \lambda_0$. Our goal is to find the maximum λ for which

$\mathcal{P}(\lambda)$ is true, assuming such a maximum exists. Suppose further that $\mathcal{P}(\lambda)$ can be solved by a (sequential) algorithm $A_s(\lambda)$ whose input is a set of data objects (independent of λ) and λ, and whose control flow is governed by comparisons, each of which amounts to testing the sign of some low-degree polynomial in λ. Megiddo's technique then runs A_s 'generically' at the unknown optimum λ^*. Whenever A_s reaches a branching point that depends on some comparison with an associated polynomial $p(\lambda)$, it computes all the roots of p and runs (the standard, non-generic version of) A_s with the value of λ equal to each of these roots. The outputs of these runs of A_s confine λ^* to an interval between two adjacent roots, which then enables the generic A_s to determine the sign of $p(\lambda^*)$, thereby resolving the comparison and allowing the generic execution to proceed. As this generic computation advances, the interval known to contain λ^* keeps shrinking as a result of resolving further comparisons, and, at the end, either the interval becomes a singleton, which is thus the desired λ^*, or else λ^* can be shown to be equal to its upper endpoint. (A third possibility is that the algorithm finds λ^* 'accidentally', during one of its comparison-resolving steps.)

If A_s runs in time T_s and makes C_s comparisons, then the cost of the procedure just described is $O(C_s T_s)$, and is thus generally quadratic in the original complexity. To speed up the execution, Megiddo proposes to implement the generic algorithm by a parallel algorithm A_p (under Valiant's *comparison model* of computation [115]). If A_p uses P processors and runs in T_p parallel steps, then each parallel step involves at most P *independent* comparisons; that is, we do not need to know the output of such a comparison to be able to execute other comparisons in the same 'batch'. We can then compute the roots of all the polynomials associated with these comparisons, and perform a binary search to locate λ^* among them, using (the non-generic) A_s at each binary step. The cost of simulating a parallel step of A_p is thus $O(P + T_s \log P)$, for a total running time of $O(PT_p + T_p T_s \log P)$. In most cases, the second term dominates the running time. The technique has been generalized further in [10, 37, 112, 113].

2.2 An Example: The Slope Selection Problem

As an illustration, consider the *slope selection* problem, which we formulate in a dual setting, as follows: We are given a set L of n nonvertical lines in the plane, and an integer $1 \leq k \leq \binom{n}{2}$, and we wish to find an intersection point between two lines of L that has the k-th smallest x-coordinate. (We assume, for simplicity, *general position* of the lines, so that no three lines are concurrent, and no two intersection points have the same x-coordinate.) We are thus seeking the k-th leftmost vertex of the *arrangement* $\mathcal{A}(L)$ of the lines in L; see [45, 108] for more details concerning arrangements. (The name of the problem comes from its primal setting, where we are given a set of n points and a parameter k as above, and wish to determine a segment connecting two input points that has the k-th smallest slope among all such segments.)

The parameter that we seek is the x-coordinate λ^* of the vertical line passing through the desired vertex, and the decision step is to compare λ^* with a given λ, that is, to determine how many vertices of $\mathcal{A}(L)$ lie to the left of (or on) the line $x = \lambda$. If we denote this number by k_λ, then we have $\lambda^* < \lambda$ (resp. $\lambda^* > \lambda$,

$\lambda^* = \lambda$) if and only if $k_\lambda > k$ (resp. $k_\lambda < k$, $k_\lambda = k$).

Let $(\ell_1, \ell_2, \ldots, \ell_n)$ denote the sequence of lines in L sorted in the decreasing order of their slopes, and let $(\ell_{\pi(1)}, \ell_{\pi(2)}, \ldots, \ell_{\pi(n)})$ denote the sequence of these lines sorted by their intercepts with $x = \lambda$. An easy observation is that two lines ℓ_i, ℓ_j, with $i < j$, intersect to the left of $x = \lambda$ if and only if $\pi(i) > \pi(j)$. In other words, the number of intersection points to the left of $x = \lambda$ can be counted, in $O(n \log n)$ time, by counting the number of *inversions* in the permutation π [78]: Construct a balanced binary tree T storing the lines of L in its leaves, in the decreasing slope order $\ell_1, \ldots, \ell_n$, by adding the lines one after the other, in the order $\ell_{\pi(1)}, \ldots, \ell_{\pi(n)}$. When a line ℓ_q is added, count the number of lines that are already present in T and have a larger index, and add up these counts, to obtain the total number of inversions. Since the insertion of a line can be done in $O(\log n)$ time, the whole decision procedure takes $O(n \log n)$ time. Any parallel sorting algorithm, which runs in $O(\log n)$ time using $O(n)$ processors [15], can count the number of inversions within the same time and processor bounds. (Notice that the construction of T itself does not involve any comparison that depends on the value of λ, and so need not be performed at all in the generic execution.) Plugging these algorithms into the parametric searching paradigm, we obtain an $O(n \log^3 n)$-time algorithm for the slope selection problem.

2.3 Improvements and Extensions

Cole [35] observed that in certain applications of parametric searching, including the slope selection problem, the running time can be improved to $O((P + T_s)T_p)$, as follows. Consider a parallel step of the above generic algorithm. Suppose that, instead of invoking the decision procedure $O(\log n)$ times in this step, we call it only $O(1)$ times, say, three times. This will determine the outcome of 7/8 of the comparisons, and will leave 1/8 of them unresolved. Suppose further that each of the unresolved comparisons can influence only a constant (and small) number of (say, two) comparisons executed at the next parallel step. Then 3/4 of these comparisons can still be simulated generically with the currently available information. This modified scheme mixes the parallel steps of the algorithm, since it forces us to perform together new comparisons and yet unresolved old comparisons. Nevertheless, Cole shows that, if carefully implemented, the number of parallel steps of the algorithm increases only by a constant factor, which leads to the improvement stated above. An ideal setup for Cole's improvement is when the parallel algorithm is described as a circuit (or network), each of whose gates has a constant fan-out.

Cole's idea improves the running time of the slope selection algorithm to $O(n \log^2 n)$. Later, Cole et al. [36] gave an optimal $O(n \log n)$-time solution. They observe that one can compare λ^* with a value λ that is 'far away' from λ^*, in a faster manner, by counting inversions only approximately. This approximation is progressively refined as λ approaches λ^* in subsequent comparisons. Cole et al. show that the overall cost of $O(\log n)$ calls to the approximating decision procedure is only $O(n \log n)$, so this also bounds the running time of the whole algorithm. This technique was subsequently simplified in [22]. Chazelle et al. [24] have shown that the algorithm of [36] can be extended to compute, in

$O(n \log n)$ time, the k-th leftmost vertex in an arrangement of n line segments.

Multi-dimensional parametric searching. The parametric searching technique can be extended to higher dimensions in a natural manner. Suppose we have a decision problem $\mathcal{P}(\lambda)$ as above, but now λ varies in $\mathbb{R}^d$. Assume also that the set Λ of points at which the answer of $\mathcal{P}(\lambda)$ is true is a convex region. We wish to compute the lexicographically largest point λ^* for which $P(\lambda)$ is true. Let A_s be, as above, an algorithm that solves $\mathcal{P}(\lambda_0)$ at any given λ_0, and can also compare λ_0 with λ^* (lexicographically). As above, we run A_s generically at λ^*. Each comparison depending on λ now amounts to evaluating the sign of some d-variate polynomial $p(\lambda_1, \ldots, \lambda_d)$.

First consider the case when p is a linear function of the form $a_0 + \sum_{1 \le i \le d} a_i \lambda_i$, such that $a_d \neq 0$. Consider the hyperplane $h : \lambda_d = -(a_0 + \sum_{i=1}^{d-1} a_i \lambda_i)/a_d$, and let h^+, h^- be the two open halfspaces bounded by h. Evaluating the sign of $p(\lambda^*)$ is equivalent to determining whether λ^* lies in h, h^+, or h^-. We solve this problem by considering the following more general problem: Let h be a k-flat in $\mathbb{R}^d$, and let $\hat{h}$ be the vertical d-hyperplane erected on h. If Λ intersects $\hat{h}$, we wish to return the lexicographically largest point of $\Lambda \cap \hat{h}$. Otherwise, we wish to determine which of the two open half-spaces bounded by $\hat{h}$ contains Λ. Inductively, assume that we have an algorithm $A_d^{(k-1)}$ that can solve this problem for any $(k-1)$-dimensional flat. Notice that $A_d^0 = A_s$, and that $k = d$ corresponds to the original problem. By running $A_d^{(k-1)}$ generically, such that λ varies over $\hat{h}$, one can determine whether Λ intersects $\hat{h}$, and, if not, determine which of the two halfspaces contains Λ. The details of this algorithm can be found in [13, 34, 81, 98]. Recently, Toledo [114] showed how to handle nonlinear polynomials, using Collins' cylindrical algebraic decomposition scheme [35].

3 Alternatives Approaches to Parametric Searching

Despite its power and versatility, the parametric searching technique, nevertheless, has some shortcomings:

(i) Parametric searching requires the design of an efficient parallel algorithm for the generic version of the decision procedure. This is not always easy, and it often tends to make the overall solution quite complicated and impractical.

(ii) The generic algorithm requires exact computation of the roots of the polynomials whose signs determine the outcome of the comparisons made by the algorithm. Such computation is possible, using standard computational algebra techniques, but it is often a time-consuming step.

(iii) Finally, from an aesthetic point of view, the execution of an algorithm based on parametric searching may appear to be somewhat chaotic, and its behavior is often difficult to explain in terms of the geometry of the problem.

These shortcomings have led several researchers to look for alternative approaches to parametric searching for geometric optimization problems. Roughly

speaking, parametric searching effectively conducts an implicit binary search over a set $\Lambda = \{\lambda_1, \ldots, \lambda_t\}$ of 'critical values' of the parameter λ, to locate the optimum λ^* among them. (For example, in the slope selection problem, the critical values are the $\Theta(n^2)$ x-coordinates of the vertices of the arrangement $\mathcal{A}(L)$.) The power of the technique stems from its ability to generate only a small number of critical values during the search.

There are alternative ways of generating a small number of critical values in certain special cases. For example, one can use randomization: Suppose we know that λ^* lies in some interval $I = [\alpha, \beta]$. If we can randomly choose an element $\lambda_0 \in I \cap \Lambda$, where each item is chosen with probability $1/|I \cap \Lambda|$, then it follows that, with high probability, the size of $I \cap \Lambda$ will shrink significantly by performing just a few comparisons with the randomly chosen elements. Proceeding along these lines, Matoušek [79] gave a very simple algorithm for the slope selection problem, which runs in $O(n \log n)$ expected time. Other randomized techniques in geometric optimization will be mentioned in Section 5.

This randomized approach can be derandomized, without affecting the asymptotic running time, using standard techniques, such as *expanders* and *geometric partitionings*. Ajtai and Megiddo [16] gave an efficient parallel linear programming algorithm based on expanders, and later Katz [70] and Katz and Sharir [71, 72] applied expanders to solve several geometric optimization problems, including the slope selection problem. Brönniman and Chazelle [22] used geometric partitionings (also known as *cuttings*), to obtain a simpler $O(n \log n)$-time deterministic algorithm for the slope selection problem.

An entirely different approach to parametric searching was proposed by Frederickson and Johnson [50, 51, 52, 53], which is based on searching in *sorted matrices*. It is applicable in cases where the set Λ of candidate critical values for the optimum parameter λ^* can be stored in an $n \times n$ matrix A, each of whose rows and columns is sorted. The size of the matrix is too large for explicit binary search through its elements, so an implicit search is needed. This is done in a logarithmic number of iterations. In each stage, we have a collection of submatrices of A. We decompose each matrix into four submatrices, and run the decision procedure on the median value of the smallest elements in each submatrix, and on the median value of the largest elements in each submatrix. It is shown in [50, 52, 53] that this allows us to discard many submatrices, so that the number of matrices only roughly doubles at each stage, and the final number of matrices is only $O(n)$. This implies that the technique can be implemented with only $O(\log n)$ calls to the decision procedure and with only $O(n)$ other constant-time operations. The technique, when applicable, is both efficient and simple, as compared with standard parametric searching.

4 Prune-and-Search Technique

Like parametric searching, the *prune-and-search* (or *decimation*) technique also performs an implicit binary search over the finite set of candidate values for λ^*, but, while doing so, it also tries to eliminate input objects that can be determined not to affect the value of λ^*. Each phase of the technique eliminates a constant fraction of the remaining objects, so that, after a logarithmic number of steps, the

problem size becomes a constant, and the problem can be solved in a final, brute-force step. The overall cost of the resulting algorithm remains proportional to the cost of a single pruning stage. The prune-and-search technique was originally introduced by Megiddo [89, 90], in developing an $O(2^{2^d} n)$-time algorithm for linear programming with n constraints in $\mathbb{R}^d$, but was later applied to many other geometric optimization problems (see Section 5). We illustrate the technique by describing Megiddo's two-dimensional linear programming algorithm.

We are given a set $H = \{h_1, \ldots, h_n\}$ of n halfplanes and a vector c, and we wish to minimize cx over the *feasible region* $K = \bigcap_{i=1}^{n} h_i$. Without loss of generality, assume that $c = (0, 1)$. Let L denote the set of lines bounding the halfplanes of H, and let L^+ (resp. L^-) denote the subset of lines $\ell_i \in L$ whose associated halfplane h_i lies below (resp. above) ℓ_i. The algorithm pairs up the lines of L into disjoint pairs (ℓ_1, ℓ_2), $(\ell_3, \ell_4), \ldots$, such that the lines in a pair either both belong to L^+ or both belong to L^-. The algorithm computes the intersection points of the lines in each pair, and chooses the median, x_m, of their x-coordinates. Let x^* denote the x-coordinate of the optimal point in K (if such a point exists). The algorithm then uses a linear-time decision procedure that compares x_m with x^*. If $x_m = x^*$ we stop, since we have found the optimum. Suppose that $x_m < x^*$. If (ℓ, ℓ') is a pair of lines, such that they both belong to L^-, and such that their intersection point lies to the left of x_m, then we can discard the line with the smaller slope from any further consideration, because that line is known to pass below the optimal point of K. All other cases can be treated in a fully symmetric manner, so we have managed to discard about $n/4$ lines. The running time of this pruning step is $O(n)$.

We have thus computed, in $O(n)$ time, a subset $H' \subseteq H$ of about $3n/4$ constraints such that the optimal point of $K' = \bigcap_{h \in H'} h$ is the same as that of K. We now apply the whole procedure once again to H', and keep repeating this, for $O(\log n)$ stages, until either the number of remaining lines falls below some small constant, in which case we solve the problem by brute force (in constant time), or the algorithm has hit x^* 'accidentally', in which case it stops right away. (We omit here the description of the decision procedure, and of handling cases in which K is empty or unbounded; see [45, 89, 90] for details.) It is now easy to see that the overall running time of the algorithm is $O(n)$.

This technique can be extended to higher dimensions, although it becomes more complicated, and requires recursive invocations of the algorithm on subproblems in lower dimensions. It yields a deterministic algorithm for linear programming that runs in $O(C_d n)$ time. The original algorithm of Megiddo gives $C_d = 2^{2^d}$, which was improved by Clarkson [30] and Dyer [41] to 3^{d^2}. Using randomization techniques, a number of simpler randomized algorithms have been developed for the problem [31, 43, 103, 117], with a better dependence on d, of which the best expected running time, $O(d^2 n + d^{d/2+O(1)} \log n)$, is due to Clarkson [31]. By derandomizing the algorithms in [31, 43], one can obtain $d^{O(d)} n$-time deterministic algorithms for linear programming in $\mathbb{R}^d$ [13, 26]. In Section 6, we will describe further improved randomized algorithms, due to Kalai [68] and to Matoušek et al. [86] (see also [110]), with subexponential expected running time.

5 Applications

In this section we briefly survey many geometric applications of parametric searching and its variants, and of the prune-and-search technique. Numerous nongeometric optimization problems have also benefited from these techniques (see [14, 34, 50, 59, 98] for a sample of such applications). Although the common theme of all the problems mentioned below is that they can be solved efficiently using parametric-searching and prune-and-search techniques, each of them requires a specific, and often fairly sophisticated, approach, involving the design of efficient sequential and parallel algorithms for solving the appropriate decision steps.

5.1 Facility Location Problems

A typical facility location problem is: Given a set D of n demand points in the plane, and a parameter p, we wish to find p supply objects (points, lines, segments, etc.), so that the maximum (Euclidean) distance between each point of D and its nearest supply object is minimized. Instead of minimizing this L_∞-norm, one can ask for the minimization of the L_1 or the L_2 norm of those 'deviations'. If p is considered as part of the input, most facility location problems are known to be NP-hard, even when the supply objects are points in the plane [94]. However, for fixed values of p, most of these problems can be solved in polynomial time. In this subsection we review efficient algorithms for some special cases of these problems.

p-center. Here we wish to compute the smallest real value r^* and a set S of p points, such that the union of disks of radius r^* centered at the points of S cover the given planar point set D. The decision problem is to determine, for a given radius r, whether D can be covered by the union of p disks of radius r. The decision problem for the 1-center is thus to determine whether D can be covered by a disk of radius r, which can be done in $O(\log n)$ parallel steps using $O(n)$ processors. This yields an $O(n \log^3 n)$-time algorithm for the 1-center problem. Using the prune-and-search paradigm, one can, however, solve the 1-center problem in linear time [41].

There is a trivial $O(n^3)$-time algorithm for the 2-center problem [40], which was improved by Agarwal and Sharir [10] to $O(n^2 \log^3 n)$, and then by Hershberger [61] to $O(n^2 \log^2 n)$. Matoušek [79] gave a simpler $O(n^2 \log^2 n)$ algorithm by replacing parametric searching by randomization. The best near-quadratic solution is due to Jaromczyk and Kowaluk [66], and runs in $O(n^2 \log n)$ time. A major progress on this problem was made recently by Sharir [107], who gave an $O(n \log^9 n)$-time algorithm, by combining the parametric searching technique with several additional tricks, including a variant of the matrix searching algorithm of Frederickson and Johnson [52]. See also [21, 39, 48, 74, 92, 93] for other results on p-center problems.

p-line-center. Here we wish to compute the smallest real value w^* such that D can be covered by the union of p strips of width w^*. For $p = 1$, this is the classical *width* problem, which can be solved in $O(n \log n)$ time [62]. For $p = 2$, Agarwal and Sharir [10] (see also [8]) gave an $O(n^2 \log^5 n)$-time algorithm. The running

time was improved to $O(n^2 \log^4 n)$ by Katz and Sharir [72] and by Glozman et al. [55], using expander graphs and the matrix searching technique, respectively; see also [67].

Segment-center. Given a segment e, we wish to find a translated and rotated copy of e such that the maximum distance from this copy to the points of D is minimized. This problem was originally considered in [64], where an $O(n^4 \log n)$ algorithm was given. An improved solution, based on parametric searching, with $O(n^2 \alpha(n) \log^3 n)$ running time, was later obtained in [4]. The best known solution, due to Efrat and Sharir [47], runs in time $O(n^{1+\varepsilon})$, for any $\varepsilon > 0$; it is also based on parametric searching, but uses a deeper combinatorial analysis of the problem structure.

5.2 Proximity Problems

Diameter. Given a set S of n points in $\mathbb{R}^3$, we wish to compute the *diameter* of S, that is, the maximum distance between two points of S. A very simple $O(n \log n)$ expected-time randomized algorithm (which is worst-case optimal) was given by Clarkson and Shor [33], but no optimal deterministic algorithm is known. The best known deterministic solution is due to Brönniman et al. [23], and runs in $O(n \log^3 n)$ time. It is based on parametric searching, and uses some interesting derandomization techniques. See also [25, 85, 104] for earlier close-to-linear time algorithms based on parametric searching.

Closest line pair. Given a set L of n lines in the $\mathbb{R}^3$, we wish to compute a closest pair of lines in L. Independently, Chazelle et al. [25] and Pellegrini [99] gave parametric-searching based algorithms for this problem, whose running time is $O(n^{8/5+\varepsilon})$, for any $\varepsilon > 0$. If we are interested in computing a pair with the minimum vertical distance, the running time can be improved to $O(n^{4/3+\varepsilon})$ [99].

Selecting distances. Let S be a set of n points in the plane, and let $1 \leq k \leq \binom{n}{2}$ be an integer. We wish to compute the k-th smallest distance between a pair of points of S. The decision problem is to compute, for a given real r, the sum $\sum_{p \in S} |D_r(p) \cap (S - \{p\})|$, where $D_r(p)$ is the disk of radius r centered at p. (This sum is twice the number of pairs of points of S at distance $\leq r$.) Agarwal et al. [2] gave an $O(n^{4/3} \log^{4/3} n)$ expected-time randomized algorithm for the decision problem, which yielded an $O(n^{4/3} \log^{8/3} n)$-time algorithm for the distance selection problem. Goodrich [56] derandomized this algorithm. Katz and Sharir [71] gave an expander-based $O(n^{4/3} \log^{3+\varepsilon} n)$-time algorithm for this problem, for any $\varepsilon > 0$. See also [102].

Minimum Hausdorff distance between polygons. Let P and Q be two polygons with m and n edges, respectively. The problem is to compute the *minimum Hausdorff distance under translation* between P and Q in the Euclidean metric. The Hausdorff distance is one of the common ways of measuring the resemblance between two sets P and Q [63]; it is defined as

$$H(P,Q) = \max \left\{ \max_{a \in P} \min_{b \in Q} d(a,b),\ \max_{a \in Q} \min_{b \in P} d(a,b) \right\},$$

and we wish to compute $\min_v H(P+v, Q)$. The problem has been solved in [12], using parametric searching, in $O((mn)^2 \log^3(mn))$ time, which is significantly faster than the previously best known algorithm of [17]. See [27, 28] for other parametric-searching based results on this problem.

5.3 Statistical Estimators and Related Problems

Plane fitting. Given a set S of n points in $\mathbb{R}^3$, we wish to fit a plane h through S so that the maximum distance between h and the points of S is minimized. This is the same problem as computing the *width* of S, which is considerably harder than the two-dimensional variant mentioned above. Chazelle et al. [25] gave an algorithm that is based on parametric searching and runs in time $O(n^{8/5+\varepsilon})$, for any $\varepsilon > 0$ (see also [1] for an improved bound). By replacing parametric searching with randomization, and by applying a more involved combinatorial analysis of the problem structure, Agarwal and Sharir [11] obtained the currently best solution, which runs in $O(n^{3/2+\varepsilon})$ expected time, for any $\varepsilon > 0$. See also [75, 84, 111, 116] for other results on hyperplane fitting.

Circle fitting. Given a set S of n points in the plane, we wish to fit a circle C through S, so that the maximum distance between C and the points of S is minimized. This is equivalent to finding an annulus of minimum width that contains S. This problem was initially solved in [44], by a quadratic-time algorithm, which was improved, using parametric searching, to $O(n^{17/11+\varepsilon})$, for any $\varepsilon > 0$ [1]. Using randomization and an improved analysis, this can be improved to $O(n^{3/2+\varepsilon})$ time, for any $\varepsilon > 0$ [11]. Finding an annulus of minimum area that contains S is a simpler problem, since it can be formulated as an instance of linear programming in $\mathbb{R}^4$, and can thus be solved in $O(n)$ time [90].

Center points. Given a set S of n points in the plane, we wish to determine a point $\sigma \in \mathbb{R}^2$, such that any halfplane containing σ also contains at least $\lfloor n/3 \rfloor$ points of S. (It is known that such σ always exists [45].) Cole et al. [37] gave an $O(n \log^3 n)$-time algorithm for computing σ, using multi-dimensional parametric searching. Using the prune-and-search paradigm, Matoušek [80] gave an $O(n \log^3 n)$-time algorithm for computing the set of all center points. Recently, Jadhav and Mukhopadhyay [65] gave a linear-time algorithm for computing a center point, using a direct and elegant technique.

For computing a center point in three dimensions, near-quadratic algorithms were developed in [37, 97]. Clarkson et al. [32] gave an efficient algorithm for computing an approximate center point.

Ham-sandwich cuts. Let $A_1, \ldots, A_d$ be d point sets in $\mathbb{R}^d$. A *ham-sandwich cut* is a hyperplane that simultaneously bisects all the A_i's. The ham-sandwich theorem (see, e.g., [45]) guarantees the existence of such a cut.

Several prune-and-search algorithms have been proposed for computing a ham-sandwich cut in the plane. For the special case when A_1 and A_2 are linearly separable, Megiddo [91] gave a linear time algorithm. Modifying his algorithm, Edelsbrunner and Waupotitsch [46] gave an $O(n \log n)$-time algorithm when A_1 and A_2 are not linearly separable. The running time was then improved to linear

by Lo and Steiger [77]. Efficient algorithms for higher dimensions are given by Lo et al. [76].

5.4 Placement and Intersection

Polygon placement. Let P be a polygonal object with m edges, and let Q be a closed planar polygonal environment with n edges. We wish to find the largest similar copy of P (under translation, rotation, and scaling) that can be placed inside Q. Sharir and Toledo [109] gave an $O(m^3 n^2 2^{\alpha(mn)} \log^3 mn \log\log mn)$-time algorithm, using parametric searching. If both P and Q are convex and rotations are not allowed, then the problem can be solved in $O(m + n\log^2 n)$ time [112]. See also [29] for related results.

A special case of this problem is the so called *biggest-stick* problem, where, given a simple polygon Q, we wish to find the longest segment that can be placed inside Q. A randomized algorithm with expected-time $O(n^{3/2+\varepsilon})$, for any $\varepsilon > 0$, is given in [11].

Intersection of polyhedra. Given a set $\mathcal{P} = \{P_1, \ldots, P_m\}$ of m convex polyhedra in $\mathbb{R}^d$, with a total of n facets, do they have a common intersection point? Reichling [105] gave an $O(\log m \log n)$-time prune-and-search algorithm for $d = 2$, and later extended his approach to $d = 3$ [106]. Using multi-dimensional parametric searching, his approach can be generalized to higher dimensions.

5.5 Query Type Problems

Agarwal and Matoušek [5] gave a general technique, based on parametric searching, to answer *ray-shooting queries* (where we wish to preprocess a given set of objects in $\mathbb{R}^d$, so that the first object hit by a query ray can be computed efficiently). This technique, further elaborated in [6, 9], has yielded fast algorithms for several related problems, including hidden surface removal, nearest neighbor searching, computing convex layers, and computing higher-order Voronoi diagrams; see [5, 6, 7, 100] for some of these results. Using multi-dimensional parametric searching, Matoušek presented in [81] efficient algorithms for *linear optimization queries*, where we wish to preprocess a set H of halfspaces in $\mathbb{R}^d$ into a linear-size data structure, so that, given a query linear objective function c, we can efficiently compute the vertex of $\bigcap H$ that minimizes c. See [3, 7, 81] for additional applications of multi-dimensional parametric searching for query type problems.

6 Abstract Linear Programming

In this section we present an abstract framework that captures both linear programming and many other geometric optimization problems, including computing smallest enclosing balls (or ellipsoids) of finite point sets in $\mathbb{R}^d$, computing largest balls (ellipsoids) in convex polytopes in $\mathbb{R}^d$, computing the distance between polytopes in d-space, general convex programming, and many other problems. Sharir and Welzl [110] and Matoušek et al. [86] (see also Kalai [68]) presented a randomized algorithm for optimization problems in this framework, whose expected running time is linear in terms of the number of constraints

whenever the dimension d is fixed. More importantly, the running time is 'subexponential' for many of the LP-type problems, including linear programming. To be more precise, what is measured here is the number of primitive operations that the algorithm performs on the constraints (see below for details). This is the first subexponential 'combinatorial' bound for linear programming (a bound that counts the number of arithmetic operations and is independent of the bit complexity of the input), and is a first step toward the major open problem of obtaining a strongly polynomial algorithm for linear programming.

6.1 An Abstract Framework

Let us consider optimization problems specified by pairs (H, w), where H is a finite set, and $w : 2^H \to \mathcal{W}$ is a function with values in a linearly ordered set $(\mathcal{W}, \leq)$; we assume that $\mathcal{W}$ has a minimum value $-\infty$. The elements of H are called *constraints*, and for $G \subseteq H$, $w(G)$ is called the *value of* G. Intuitively, $w(G)$ denotes the smallest value attainable for certain objective function while satisfying all the constraints of G. The goal is to compute a minimal subset B_H of H with the same value as H (from which, in general, the value of H is easy to determine), assuming the availability of three basic operations to be specified below.

Such a minimization problem is called *LP-type* if the following two axioms are satisfied:

Axiom 1. (*Monotonicity*) For any F, G with $F \subseteq G \subseteq H$, we have
$$w(F) \leq w(G)$$

Axiom 2. (*Locality*) For any $F \subseteq G \subseteq H$ with $-\infty < w(F) = w(G)$ and any $h \in H$,
$$w(G) < w(G \cup \{h\}) \Rightarrow w(F) < w(F \cup \{h\}).$$

Linear programming is easily shown to be an LP-type problem, if we set $w(G)$ to be the vertex of the feasible region which minimizes the objective function and which is lexicographically smallest (this definition is important to satisfy Axiom 2), and if we define $w(G)$ in an appropriate manner to handle empty or unbounded feasible regions.

A *basis* B is a set of constraints with $-\infty < w(B)$, and $w(B') < w(B)$ for all proper subsets B' of B. For $G \subseteq H$, if $-\infty < w(G)$, a *basis of* G is a minimal subset B of G with $w(B) = w(G)$. (For linear programming, a basis is a minimal set of halfspace constraints such that the minimal vertex of their intersection is some prescribed vertex.) A constraint h is *violated by* G, if $w(G) < w(G \cup \{h\})$, and it is *extreme in* G, if $w(G - \{h\}) < w(G)$. The *combinatorial dimension of* (H, w), denoted as $\dim(H, w)$, is the maximum cardinality of any basis. We call an LP-type problem *basis regular* if for any basis with $|B| = \dim(H, w)$ and for any constraint h, every basis of $B \cup \{h\}$ has exactly $\dim(H, w)$ elements. (Clearly, linear programming is basis-regular, where the dimension of every basis is d.)

We assume that the following primitive operations are available.

(*Violation test*) 'h `is violated by` B', for a constraint h and a basis B, tests whether h is violated by B or not.

(*Basis computation*) '`basis`(B, h)', for a constraint h and a basis B, computes a basis of $B \cup \{h\}$.

(*Initial basis*) An initial basis B_0 with exactly $\dim(H, w)$ elements is available.

For linear programming, the first operation can be done in $O(d)$ time, by substituting the coordinates of the vertex $w(B)$ into the equation of the hyperplane defining h. The second operation can be regarded as a dual version of the pivot step in the simplex algorithm, and can be implemented in $O(d^2)$ time. The third operation is also easy to implement.

We are now in position to describe the algorithm. Using the initial-basis primitive, we compute a basis B_0 and use the following recursive algorithm to compute B_H.

```
function procedure SUBEX_lp(H,C);          /* H: n constraints in R^d;
    if H = C then                                 /* C ⊆ H: a basis;
        return C                                /* returns a basis of H.
    else
        choose a random h ∈ H − C;
        B := SUBEX_lp(H − {h}, C);
        if h is violated by B then                          /* ⇔ v_B ∉ h
            return SUBEX_lp(H, basis(B, h))
        else
            return B;
```

A simple inductive argument shows the expected number of primitive operations performed by the algorithm is $O(2^\delta n)$, where $n = |H|$ and $\delta = \dim(H, w)$ is the combinatorial dimension. However, using a more involved analysis, which can be found in [86], one can show that basis-regular LP-type problems can be solved with an expected number of at most $e^{2\sqrt{\delta \ln((n-\delta)/\sqrt{\delta})+O(\sqrt{\delta}+\ln n)}}$ violation tests and basis computations. This is the 'subexponential' bound that we alluded to.

6.2 Linear Programming

We are given a set H of n halfspaces in $\mathbb{R}^d$. We assume that the objective vector is $c = (1, 0, 0, \ldots, 0)$, and the goal is to minimize cx over all points in the common intersection $\bigcap_{h \in H} h$. For a subset $G \subseteq H$, define $w(G)$ to be the lexicographically smallest point (vertex) of the intersection of halfspaces in G.

As noted above, linear programming is a basis-regular LP-type problem, with combinatorial dimension d, and violation tests and basis changes operations can be implemented in time $O(d)$ and $O(d^2)$, respectively. In summary, we obtain a randomized algorithm for linear programming, which performs $e^{2\sqrt{d \ln(n/\sqrt{d})+O(\sqrt{d}+\ln n)}}$ expected number of arithmetic operations. Combining this algorithm with Clarkson's randomized linear programming algorithm

mentioned in Section 4, the expected number of arithmetic operations can be reduced to $O(d^2 n) + e^{O(\sqrt{d \log d})}$.

Matoušek [82] has given examples of abstract LP-type problems of combinatorial dimension d and with $2d$ constraints, for which the above algorithm requires $\Omega(e^{\sqrt{2d}}/\sqrt[4]{d})$ primitive operations. Hence, in order to obtain a better bound on the performance of the algorithm for linear programming, one should aim to exploit additional properties of linear programming; this is still open.

6.3 Smallest Enclosing Ball and Related Problems

In Section 5.1 we mentioned that the smallest enclosing ball (i.e., disk) of a set of n points in the plane can be computed in linear time. In higher dimensions, one can solve this problem by an $d^{O(d)}n$-time algorithm [13, 26, 42]. It can be shown that the smallest enclosing ball problem is an LP-type problem, with combinatorial dimension $d+1$. It is, however, not basis-regular, and a naive implementation of the basis-changing operation may be quite costly (in d). Nevertheless, Gärtner [54] showed that this operation can be performed in this case using expected $e^{O(\sqrt{d})}$ arithmetic operations. Hence, the expected running time of the algorithm is $O(d^2 n) + e^{O(\sqrt{d \log d})}$.

There are several extensions of the smallest enclosing ball problem. They include: (i) computing the smallest enclosing ellipsoid of a point set [26, 42, 101, 117], (ii) computing the largest ellipsoid (or ball) inscribed inside a convex polytope in $\mathbb{R}^d$ [54], (iii) computing a smallest ball that intersects (or contains) a given set of convex objects in $\mathbb{R}^d$, and (iv) computing a smallest volume annulus containing a given planar point set. All these problems are known to be LP-type, and thus can be solved using the above algorithm. However, not all of them run in subexponential expected time because they are not basis regular.

6.4 Distance Between Polytopes

We wish to compute the Euclidean distance $d(\mathcal{P}_1, \mathcal{P}_2)$ between two given closed polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$. If the polytopes intersect, then this distance is 0. If they do not intersect, then this distance equals the maximum distance between two parallel hyperplanes separating the polytopes; such a pair of hyperplanes is unique, and they are orthogonal to the segment connecting two points $a \in \mathcal{P}_1$ and $b \in \mathcal{P}_2$ with $d(a, b) = d(\mathcal{P}_1, \mathcal{P}_2)$. It is shown by Gärtner [54] that this problem is LP-type, with combinatorial dimension at most $d+2$ (or $d+1$, if the polytopes do not intersect). It is also shown there that the primitive operations can be performed with expected $e^{O(\sqrt{d})}$ arithmetic operations. Hence, as above, the expected number of arithmetic operations is $O(d^2 n) + e^{O(\sqrt{d \log d})}$.

6.5 Extensions

Recently, Chazelle and Matoušek [26] gave a deterministic algorithm for solving LP-type problems in time $O(\delta^{O(\delta)} n)$, provided an additional axiom holds (together with an additional computational assumption). Still, these extra requirements are satisfied in many natural LP-type problems. Matoušek [83] investigates the problem of finding the best solution, for abstract LP-type problems, which satisfies all but k of the given constraints.

Amenta [18] considers the following extension of the abstract framework: Suppose we are given a family of LP-type problems (H, w_λ), monotonically parameterized by a real parameter λ; the underlying ordered value set $\mathcal{W}$ has a maximum element $+\infty$ representing *infeasibility*. The goal is to find the smallest λ for which (H, w_λ) is feasible, i.e. $w_\lambda(H) < +\infty$. See [19, 20] for related work.

6.6 Abstract Linear Programming and Helly-type Theorems

In this subsection we describe an interesting connection between Helly-type theorems and LP-type problems, as originally noted by Amenta [18].

Let $\mathbf{K}$ be an infinite collection of sets in $\mathbb{R}^d$, and let t be an integer. We say that $\mathbf{K}$ satisfies a *Helly-type* theorem, with *Helly number* t, if the following holds: If $\mathcal{K}$ is a finite subcollection of $\mathbf{K}$ with the property that every subcollection of t elements of $\mathcal{K}$ has a nonempty intersection, then $\bigcap \mathcal{K} \neq \emptyset$. (The best known example of a Helly-type theorem is Helly's theorem itself [60], which applies for the collection $\mathbf{K}$ of all convex sets in $\mathbb{R}^d$, with the Helly number $d+1$.) Suppose further that we are given a collection $\mathcal{K}(\lambda)$, consisting of n sets $K_1(\lambda), \ldots, K_n(\lambda)$ that are parametrized by some real parameter λ, with the property that $K_i(\lambda) \subseteq K_i(\lambda')$, for $i = 1, \ldots, n$ and for $\lambda \leq \lambda'$, and that, for any fixed λ, the family $\{K_1(\lambda), \ldots, K_n(\lambda)\}$ admits a Helly-type theorem, with Helly number t. Our goal is to compute the smallest λ for which $\bigcap_{i=1}^n K_i(\lambda) \neq \emptyset$, assuming that such a minimum exists. Amenta proved that this problem can be transformed to an LP-type problem, whose combinatorial dimension is at most t.

As an illustration, consider the smallest enclosing ball problem. Let $P = \{p_1, \ldots, p_n\}$ be the given set of n points in $\mathbb{R}^d$, and let $K_i(\lambda)$ be the ball of radius λ centered at p_i, for $i = 1, \ldots, n$. Since the K_i's are convex, the collection in question has Helly number $d+1$. It is easily seen that the minimal λ for which the $K_i(\lambda)$'s have nonempty intersection is the radius of the smallest enclosing ball of P.

There are several other examples where Helly-type theorems can be turned into LP-type problems. They include (i) computing a line transversal to a family of translates of some convex objects in the plane, (ii) computing a smallest homothet of a given convex set that intersects (or contains, or is contained in) every member in a given collection of n convex sets in $\mathbb{R}^d$, and (iii) computing a line transversal to certain families of convex objects in 3-space. We refer the reader to [19, 20] for more details and for additional examples.

References

[1] P. Agarwal, B. Aronov, and M. Sharir, Computing lower envelopes in four dimensions with applications, *Proc. 10th ACM Symp. Comput. Geom.*, 1994, pp. 348–358.

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

[3] P. Agarwal, A. Efrat, and M. Sharir, Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications, *Proc. 11th ACM Symp. Comput. Geom.*, 1995, 39–50.

[4] P. Agarwal, A. Efrat, M. Sharir, and S. Toledo, Computing a segment-center for a planar point set, *J. Algorithms* 15 (1993), 314–323.

[5] P. Agarwal and J. Matoušek, Ray shooting and parametric search, *SIAM J. Comput.* 22 (1993), 794–806.

[6] P. Agarwal and J. Matoušek, Range searching with semialgebraic sets, *Discr. Comput. Geom.* 11 (1994), 393–418.

[7] P. Agarwal and J. Matoušek, Dynamic half-space range searching and its applications, *Algorithmica* 14 (1995), 325–345.

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

[9] P. Agarwal and M. Sharir, Ray shooting among convex polytopes in three dimensions, *SIAM J. Comput.*, to appear.

[10] P. Agarwal and M. Sharir, Planar geometric location problems, *Algorithmica* 11 (1994), 185–195.

[11] P. Agarwal and M. Sharir, Efficient randomized algorithms for some geometric optimization problems, *Proc. 11th ACM Symp. Comput. Geom.*, 1995, pp. 326–335.

[12] P. Agarwal, M. Sharir, and S. Toledo, New applications of parametric searching in computational geometry, *J. Algorithms* 17 (1994), 292–318.

[13] P. Agarwal, M. Sharir, and S. Toledo, An efficient multi-dimensional searching technique and its applications, Tech. Rept. CS-1993-20, Dept. Comp. Sci., Duke University, 1993.

[14] R. Agarwala and D. Fernández-Baca, Weighted multidimensional search and its applications to convex optimization, *SIAM J. Comput.*, to appear.

[15] M. Ajtai, J. Komlós, and E. Szemerédi, Sorting in $c \log n$ parallel steps, *Combinatorica*, 3 (1983), 1–19.

[16] M. Ajtai and N. Megiddo, A deterministic $poly(\log \log N)$-time N-processor algorithm for linear programming in fixed dimension, *Proc. 24th ACM Symp. Theory of Comput.*, 1992, pp. 327–338.

[17] H. Alt, B. Behrends, and J. Blömer, Approximate matching of polygonal shapes, *Proc. 7th ACM Symp. Comput. Geom.*, 1991, pp. 186–193.

[18] N. Amenta, Finding a line transversal of axial objects in three dimensions, *Proc. 3rd ACM-SIAM Symp. Discr. Algo.*, 1992, pp. 66–71.

[19] N. Amenta, Helly-type theorems and generalized linear programming, *Discr. Comput. Geom.* 12 (1994), 241–261.

[20] N. Amenta, Bounded boxes, Hausdorff distance, and a new proof of an interesting Helly-type theorem, *Proc. 10th ACM Symp. Comput. Geom.*, 1994, pp. 340–347.

[21] R. Bar-Yehuda, A. Efrat, and A. Itai, A simple algorithm for maintaining the center of a planar point-set, *Proc. 5th Canad. Conf. Comput. Geom.*, 1993, pp. 252–257.

[22] H. Brönnimann and B. Chazelle, Optimal slope selection via cuttings, *Proc. 6th Canad. Conf. Comput. Geom.*, 1994, pp. 99–103.

[23] H. Brönnimann, B. Chazelle, and J. Matoušek, Product range spaces, sensitive sampling, and derandomization, *Proc. 34th IEEE Symp. Found. of Comp. Sci.*, 1993, pp. 400–409.

[24] B. Chazelle, H. Edelsbrunner, L. Guibas, and M. Sharir, Algorithms for bichromatic line segment problems and polyhedral terrains, *Algorithmica* 11 (1994), 116–132.

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

[26] B. Chazelle and J. Matoušek, On linear-time deterministic algorithms for optimization problems in fixed dimensions, *Proc. 4th ACM-SIAM Symp. Discr. Algo.*, 1993, pp. 281–290.

[27] P. Chew, D. Dor, A. Efrat, and K. Kedem, Geometric pattern matching in d-dimensional space, *Proc. 3rd European Symp. Algo.*, 1995, to appear.

[28] P. Chew and K. Kedem, Improvements on geometric pattern matching problems, *Proc. 3rd Scand. Work. Algo. Theory*, 1992, pp. 318–325.

[29] P. Chew and K. Kedem, A convex polygon among polygonal obstacles: Placement and high-clearance motion, *Comput. Geom. Theory Appls.* 3 (1993), 59–89.

[30] K. Clarkson, Linear Programming in $O(n \cdot 3^{d^2})$ time, *Inform. Process. Lett.* 22 (1986), 21–24.

[31] K. Clarkson, Las Vegas algorithms for linear and integer programming when the dimension is small, *J. ACM* 45 (1995), 488–499.

[32] K. Clarkson, D. Eppstein, G. Miller, C. Sturtivant, and S.-H. Teng, Approximating center points with iterated Radon points, *Proc. 9th ACM Symp. Comput. Geom.*, 1993, pp. 91–98.

[33] K. Clarkson and P. Shor, Applications of random sampling in computational geometry, II, *Discr. Comput. Geom.* 4 (1989), 387–421.

[34] E. Cohen and N. Megiddo, Maximizing concave functions in fixed dimension, in *Complexity in Numeric Computation* (P. Pardalos, ed.), World Scientific, 1993.

[35] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. ACM* 31 (1984), 200–208.

[36] R. Cole, J. Salowe, W. Steiger, and E. Szemerédi, Optimal slope selection, *SIAM J. Comput.* 18 (1989), 792–810.

[37] R. Cole, M. Sharir, and C. Yap, On k-hulls and related problems, *SIAM J. Comput.* 16 (1987), 61–77.

[38] G. Collins, Quantifier elimination for real closed fields by cylindrical algebraic decomposition, in *Second GI Conf. on Automata Theory and Formal Languages*, Lecture Notes in Comp. Sci., 33, Springer-Verlag, 1975, pp. 134–183.

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

[40] Z. Drezner, The P-center problem: Heuristics and optimal algorithms, *J. Oper. Res. Soc.* 35 (1984), 741–748.

[41] M. Dyer, On a multidimensional search technique and its application to the Euclidean one-center problem, *SIAM J. Comput.* 15 (1986), 725–738.

[42] M. Dyer, A class of convex programs with applications to computational geometry, *Proc. 8th ACM Symp. Comput. Geom.*, 1992, pp. 9–15.

[43] M. Dyer and A. Frieze, A randomized algorithm for fixed-dimensional linear programming, *Math. Prog.* 44 (1989), 203–212.

[44] H. Ebara, N. Fukuyama, H. Nakano and Y. Nakanishi, Roundness algorithms using the Voronoi diagrams, *First Canad. Conf. Comput. Geom.*, 1989.

[45] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[46] H. Edelsbrunner and R. Waupotitsch, Computing a ham-sandwich cut in two dimensions, *J. Symb. Comput.* 2 (1986), 171–178.

[47] A. Efrat and M. Sharir, A near-linear algorithm for the planar segment center problem, *Discr. Comput. Geom.*, to appear.

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

[49] M. Eisner and D. Severance, Mathematical techniques for efficient record segmen-

tation in large shared databases, *JACM* 23 (1976), 619–635.
[50] G. Frederickson, Optimal algorithms for tree partitioning, *Proc. 2nd ACM-SIAM Symp. Discr. Algo.*, 1991, pp. 168–177.
[51] G. Frederickson and D. Johnson, The complexity of selection and ranking in $X+Y$ and matrices with sorted columns, *J. Comp. Syst. Sci.* 24 (1982), 197–208.
[52] G. Frederickson and D. Johnson, Finding the k-th shortest paths and p-centers by generating and searching good data structures, *J. Algorithms* 4 (1983), 61–80.
[53] G. Frederickson and D. Johnson, Generalized selection and ranking: sorted matrices, *SIAM J. Comput.* 13 (1984), 14–30.
[54] B. Gärtner, A subexponential algorithm for abstract optimization problems, *Proc. 33rd IEEE Symp. Found. of Comp. Sci.*, 1992, pp. 464–472.
[55] A. Glozman, K. Kedem, and G. Shpitalnik, On some geometric selection and optimization problems via sorted matrices, *Proc. 4th Workshop Algo. and Data Struct.*, 1995, to appear.
[56] M. Goodrich, Geometric partitioning made easier, even in parallel, *Proc. 9th ACM Symp. Comput. Geom.*, 1993, pp. 73–82.
[57] D. Gusfield, Sensitivity analysis for combinatorial optimization, Tech. Rept. UCB/ERLM80/22, Univ. of California, Berkeley, 1980.
[58] D. Gusfield, Parametric combinatorial computing and a problem in program module allocation, *JACM* 30 (1983), 551–563.
[59] D. Gusfield, K. Balasubramanian, and D. Naor, Parametric optimization of sequence alignment, *Algorithmica* 12 (1994), 312–326.
[60] E. Helly, Über systeme von abgeschlossenen Mengen mit gemeinschaftlichen Punkten, *Monaths. Math. und Physik* 37 (1930), 281–302.
[61] J. Hershberger, A faster algorithm for the two-center decision problem, *Inform. Process. Lett.* 47 (1993), 23–29.
[62] M. Houle and G. Toussaint, Computing the width of a set, *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-10 (1988), 761–765.
[63] D. Huttenlocher and K. Kedem, Efficiently computing the Hausdorff distance for point sets under translation, *Proc. 6th ACM Symp. Comput. Geom.*, 1990, pp. 340–349.
[64] H. Imai, D.T. Lee, and C. Yang, 1-Segment center covering problems, *ORSA J. Comput.* 4 (1992), 426–434.
[65] S. Jadhav and A. Mukhopadhyay, Computing a centerpoint of a finite planar set of points in linear time, *Proc. 9th ACM Symp. Comput. Geom.*, 1993, pp. 83–90.
[66] J. Jaromczyk and M. Kowaluk, An efficient algorithm for the Euclidean two-center problem, *Proc. 10th ACM Symp. Comput. Geom.*, 1994, pp. 303–311.
[67] J. Jaromczyk and M. Kowaluk, The two-line center problem from a polar view: A new algorithm, *Proc. 4th Workshop Algo. Data Struct.*, 1995.
[68] G. Kalai, A subexponential randomized simplex algorithm, *Proc. 24th ACM Symp. Theory of Comput.*, 1992, pp. 475–482.
[69] N. Karmarkar, A new polynomial-time algorithm for linear programming, *Combinatorica* 4 (1984), 373–395.
[70] M. Katz, Improved algorithms in geometric optimization via expanders, *Proc. 3rd Israeli Symp. Theory of Comput. & Syst.*, 1995.
[71] M. Katz and M. Sharir, Optimal slope selection via expanders, *Inform. Process. Lett.* 47 (1993), 115–122.
[72] M. Katz and M. Sharir, An expander-based approach to geometric optimization, *Proc. 9th ACM Symp. Comput. Geom.*, 1993, pp. 198–207.
[73] L.G. Khachiyan, Polynomial algorithm in linear programming, *U.S.S.R. Comput.*

Math. and Math. Phys. 20 (1980), 53–72.

[74] M. Ko and R. Lee, On weighted rectilinear 2-center and 3-center problems, *Info. Sci.* 54 (1991), 169–190.

[75] N. Korneenko and H. Martini, Hyperplane approximation and related topics, in *New Trends in Discrete and Computational Geometry*, (J. Pach, ed.), Springer-Verlag, 1993, 163–198.

[76] C.-Y. Lo, J. Matoušek, and W. Steiger, Algorithms for ham-sandwich cuts, *Discr. Comput. Geom.* 11 (1994), 433–452.

[77] C.-Y. Lo and W. Steiger, An optimal-time algorithm for ham-sandwich cuts in the plane, *Proc. 2nd Canad. Conf. Comput. Geom.*, 1990, pp. 5–9.

[78] D. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, Reading, MA, 1973.

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

[80] J. Matoušek, Computing the center of planar point sets, in *Computational Geometry: Papers from the DIMACS Special Year* (J. Goodman, et al., eds.), AMS, Providence, RI, 1991, 221–230.

[81] J. Matoušek, Linear optimization queries, *J. Algorithms* 14 (1993), 432–448.

[82] J. Matoušek, Lower bounds for a subexponential optimization algorithm, *Random Struct. & Algo.* 5 (1994), 591–607.

[83] J. Matoušek, On geometric optimization with few violated constraints, *Proc. 10th ACM Symp. Comput. Geom.*, 1994, pp. 312–321.

[84] J. Matoušek, D. Mount, and N. Netanyahu, Efficient randomized algorithms for the repeated median line estimator, *Proc. 4th ACM-SIAM Symp. Discr. Algo.*, 1993, pp. 74–82.

[85] J. Matoušek and O. Schwarzkopf, A deterministic algorithm for the three-dimensional diameter problem, *Proc. 25th ACM Symp. Theory of Comput.*, 1993, pp. 478–484.

[86] J. Matoušek, M. Sharir, and E. Welzl, A subexponential bound for linear programming and related problems, *Algorithmica*, to appear.

[87] N. Megiddo, Combinatorial optimization with rational objective functions, *Math. Oper. Res.* 4 (1979), 414–424.

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

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

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

[91] N. Megiddo, Partitioning with two lines in the plane, *J. Algorithms* 6 (1985), 403–433.

[92] N. Megiddo, The weighted Euclidean 1-center problem, *Math. Oper. Res.* 8 (1983), 498–504.

[93] N. Megiddo, On the ball spanned by balls, *Discr. Comput. Geom.* 4 (1989), 605–610.

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

[95] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, 1995.

[96] K. Mulmuley, *Computational Geometry: An Introduction Through Randomized Algorithms*, Prentice Hall, New York, 1993.

[97] N. Naor and M. Sharir, Computing the center of a point set in three dimensions, *Proc. 2nd Canad. Conf. Comput. Geom.*, 1990, pp. 10–13.

[98] C. Norton, S. Plotkin, and E. Tárdos, Using separation algorithms in fixed dimensions, *J. Algorithms* 13 (1992), 79–98.

[99] M. Pellegrini, Incidence and nearest-neighbor problems for lines in 3-space, *Proc. 8th ACM Symp. Comput. Geom.*, 1992, pp. 130–137.

[100] M. Pellegrini, Repetitive hidden-surface-removal for polyhedral scenes, *Proc. 3rd Workshop Algo. Data Struct.*, 1993, pp. 541–552.

[101] M. Post, Minimum spanning ellipsoids, *Proc. 16th ACM Symp. Theory of Comput.*, 1984, pp. 108–116.

[102] J. Salowe, L_∞-interdistance selection by parametric search, *Inform. Process. Lett.* 30 (1989), 9–14.

[103] R. Seidel, Low dimensional linear programming and convex hulls made easy, *Discr. Comput. Geom.* 6 (1991), 423–434.

[104] E. Ramos, Intersection of unit-balls and diameter of a point set in $\mathbb{R}^3$, manuscript, 1993.

[105] M. Reichling, On the detection of a common intersection of k convex objects in the plane, *Inform. Process. Lett.*, 29 (1988), 25–29.

[106] M. Reichling, On the detection of a common intersection of k convex polyhedra, in *Comput. Geom. & its Appl.*, Lecture Notes in Comp. Sci., 333, Springer-Verlag, 1988, pp. 180–186.

[107] M. Sharir, A near-linear algorithm for the planar 2-center problem, submitted to *Discr. Comput. Geom.*

[108] M. Sharir and P. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications,* Cambridge University Press, Cambridge-New York-Melbourne, 1995.

[109] M. Sharir and S. Toledo, Extremal polygon containment problems, *Comput. Geom. Theory Appls.* 4 (1994), 99–118.

[110] M. Sharir and E. Welzl, A combinatorial bound for linear programming and related problems, *Proc. 9th Symp. Theo. Asp. Comp. Sci.*, (1992), pp. 569–579.

[111] A. Stein and M. Werman, Finding the repeated median regression line, *Proc. 3rd ACM-SIAM Symp. Discr. Algo.*, 1992, pp. 409–413.

[112] S. Toledo, *Extremal Polygon Containment Problems and Other Issues in Parametric Searching*, M.Sc. Thesis, Tel Aviv University, Tel Aviv, 1991.

[113] S. Toledo, Approximate parametric search, *Inform. Process. Lett.* 47 (1993), 1–4.

[114] S. Toledo, Maximizing non-linear convex functions in fixed dimensions, in *Complexity of Numerical Computations* (P. Pardalos, ed.), World Scientific, 1993.

[115] L. Valiant, Parallelism in comparison problems, *SIAM J. Comput.* 4 (1975), 348–355.

[116] K. Vardarajan and P. Agarwal, Linear approximation of simple objects, *Proc. 7th Canad. Conf. Comput. Geom.*, 1995, to appear.

[117] E. Welzl, Smallest enclosing disks (balls and ellipsoids), in *New Results and New Trends in Computer Science* (H. Maurer, ed.), Lecture Notes in Comp. Sci., 555, 1991, Sringer-Verlag, pp. 359–370.