49 LINEAR PROGRAMMING

Martin Dyer, Bernd Gärtner, Nimrod Megiddo,
and Emo Welzl

INTRODUCTION

Linear programming has many important practical applications, and has also given rise to a wide body of theory. See Section 49.9 for recommended sources. Here we consider the linear programming problem in the form of maximizing a linear function of d variables subject to n linear inequalities. We focus on the relationship of the problem to computational geometry, i.e., we consider the problem in small dimension. More precisely, we concentrate on the case where $d \ll n$, i.e., $d = d(n)$ is a function that grows very slowly with n. By linear programming duality, this also includes the case $n \ll d$. This has been called *fixed-dimensional* linear programming, though our viewpoint here will not treat d as constant. In this case there are strongly polynomial algorithms, provided the rate of growth of d with n is small enough.

The plan of the chapter is as follows. In Section 49.2 we consider the simplex method, in Section 49.3 we review deterministic linear time algorithms, in Section 49.4 randomized algorithms, and in Section 49.5 we consider the derandomization of the latter. Section 49.6 discusses the combinatorial framework of LP-type problems, which underlie most current combinatorial algorithms and allows their application to a host of optimization problems. We briefly describe the more recent combinatorial framework of unique sink orientations, in the context of striving for algorithms with a milder dependence on d. In Section 49.7 we examine parallel algorithms for this problem, and finally in Section 49.8 we briefly discuss related issues. The emphasis throughout is on complexity-theoretic bounds for the linear programming problem in the form (49.1.1).

49.1 THE BASIC PROBLEM

Any linear program (LP) may be expressed in the inequality form

$$\begin{aligned} \text{maximize} \quad & z = c \cdot x \\ \text{subject to} \quad & Ax \leq b\,, \end{aligned} \tag{49.1.1}$$

where $c \in \mathbb{R}^d$, $b \in \mathbb{R}^n$, and $A \in \mathbb{R}^{n \times d}$ are the input data and $x \in \mathbb{R}^d$ the variables. Without loss of generality, the columns of A are assumed to be linearly independent. The vector inequality in (49.1.1) is with respect to the componentwise partial order on $\mathbb{R}^n$. We will write a_i for the ith row of A, so the constraint may also be expressed in the form

$$a_i \cdot x = \sum_{j=1}^{d} a_{ij} x_j \leq b_i \quad (i = 1, \ldots, n). \tag{49.1.2}$$

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

GLOSSARY

Constraint: A condition that must be satisfied by a solution.

Objective function: The linear function to be maximized over the set of solutions.

Inequality form: The formulation of the linear programming problem where all the constraints are weak inequalities $a_i \cdot x \leq b_i$.

Feasible set: The set of points that satisfy all the constraints. In the case of linear programming, it is a convex polyhedron in $\mathbb{R}^d$.

Defining hyperplanes: The hyperplanes described by the equalities $a_i \cdot x = b_i$.

Tight constraint: An inequality constraint is tight at a certain point if the point lies on the corresponding hyperplane.

Infeasible problem: A problem with an empty feasible set.

Unbounded problem: A problem with no finite maximum.

Vertex: A feasible point where at least d linearly independent constraints are tight.

Nondegenerate problem: A problem where at each vertex precisely d constraints are tight.

Strongly polynomial-time algorithm: An algorithm for which the total number of arithmetic operations and comparisons (on numbers whose size is polynomial in the input length) is bounded by a polynomial in n and d alone.

We observe that (49.1.1) may be infeasible or unbounded, or have multiple optima. A complete algorithm for linear programming must take account of these possibilities. In the case of multiple optima, we assume that we have merely to identify *some* optimum solution. (The task of identifying *all* optima is considerably more complex; see [Dye83, AF92].) An optimum of (49.1.1) will be denoted by x^0. At least one such solution (assuming one exists) is known to lie at a vertex of the feasible set. There is little loss in assuming nondegeneracy for theoretical purposes, since we may "infinitesimally perturb" the problem to ensure this using well known methods [Sch86]. However, a complete algorithm must recognize and deal with this possibility.

It is well known that linear programs can be solved in time polynomial in the total length of the input data [GLS93]. However, it is not known in general if there is a *strongly* polynomial-time algorithm. This is true even if randomization is permitted. (Algorithms mentioned below may be assumed deterministic unless otherwise stated.) The "weakly" polynomial-time algorithms make crucial use of the size of the numbers, so seem unlikely to lead to strongly polynomial-time methods. However, strong bounds are known in some special cases. Here are two examples. If all a_{ij} are bounded by a constant, then É. Tardos [Tar86] has given a strongly polynomial-time algorithm. If every row a_i has at most two nonzero entries, Megiddo [Meg83a] has shown how to find a feasible solution in strongly polynomial time.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

49.2 THE SIMPLEX METHOD

GLOSSARY

Simplex method: For a nondegenerate problem in inequality form, this method seeks an optimal vertex by iteratively moving from one vertex to a neighboring vertex of higher objective function value.

Pivot rule: The rule by which a neighboring vertex is chosen.

Random-edge simplex algorithm: A randomized variant of the simplex method where the neighboring vertex is chosen uniformly at random.

Dantzig's simplex method is probably still the most commonly used method for solving large linear programs in practice, but (with Dantzig's original pivot rule) Klee and Minty showed that the algorithm may require an exponential number of iterations in the worst case. For example, it may require $2^d - 1$ iterations when $n = 2d$. Other variants were subsequently shown to have similar behavior. While it is not known for certain that all suggested variants of the simplex method have this bad worst case, there seems to be no reason to believe otherwise. In fact, two long-standing candidates for a polynomial-time variant (Zadeh's and Cunningham's rule) have recently been shown to have bad worst-case behavior as well [Fri11, AF16]. This breakthrough is due to a new technique by Friedmann who constructed lower bound instances in a combinatorial way from certain games on graphs. Earlier lower bound constructions were typically in the form of "raw" linear programs, with carefully crafted coordinates. But this direct approach (pioneered by Klee and Minty [KM72]) could so far not be applied to history-dependent rules such as Zadeh's and Cunningham's.

When we assume $d \ll n$, the simplex method may require $\Omega(n^{\lfloor d/2 \rfloor})$ iterations [KM72, AZ99], and thus it is polynomial only for $d = O(1)$. This is asymptotically no better than enumerating all vertices of the feasible region.

By contrast, Kalai [Kal92] gave a randomized simplex-like algorithm that requires only $2^{O(\sqrt{d \log n})}$ iterations. (An identical bound was also given by Matoušek, Sharir, and Welzl [MSW96] for a closely related algorithm; see Section 49.4.) Combined with Clarkson's methods [Cla95], this results in a bound of $O(d^2 n) + e^{O(\sqrt{d \log d})}$ (cf. [MSW96]). This is the best "strong" bound known, other than for various special cases, and it is evidently polynomial-time provided $d = O(\log^2 n / \log \log n)$. No complete derandomization of these algorithms is known, and it is possible that randomization may genuinely help here. However, Friedmann, Hansen and Zwick recently showed that the complexity of the random-edge simplex algorithm (where the pivot is chosen uniformly at random) is actually superpolynomial, defeating the most promising candidate for an (expected) polynomial-time pivot rule [FHZ11]. By giving a lower bound also for the random-facet rule, they actually prove that the analysis underlying the best strong bound above is essentially tight. These are the first superpolynomial lower bound constructions for natural randomized simplex variants, again exploiting the new game-based technique by Friedmann. Some less natural randomized variants have not been defeated yet, and it remains to be seen whether this would also be possible through Friedmann's approach.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

49.3 LINEAR-TIME LINEAR PROGRAMMING

The study of linear programming within computational geometry was initiated by Shamos [Sha78] as an application of an $O(n \log n)$ convex hull algorithm for the intersection of halfplanes. Muller and Preparata [MP78] gave an $O(n \log n)$ algorithm for the intersection of halfspaces in $\mathbb{R}^3$. Dyer [Dye84] and Megiddo [Meg83b] found, independently, an $O(n)$ time algorithm for the linear programming problem in the cases $d = 2, 3$.

Megiddo [Meg84] generalized the approach of these algorithms to arbitrary d, arriving at an algorithm of time complexity $O(2^{2^d} n)$, which is polynomial for $d \leq \log \log n + O(1)$. This was subsequently improved by Clarkson [Cla86b] and Dyer [Dye86] to $O(3^{d^2} n)$, which is polynomial for $d = O(\sqrt{\log n})$. Megiddo [Meg84, Meg89] and Dyer [Dye86, Dye92] showed that Megiddo's idea could be used for many related problems: Euclidean one-center, minimum ball containing balls, minimum volume ellipsoid, etc.; see also the derandomized methods and LP-type problems in the sections below.

GLOSSARY

Multidimensional search: Given a set of hyperplanes and an oracle for locating a point relative to any hyperplane, locate the point relative to all the input hyperplanes.

MEGIDDO'S ALGORITHMS

The basic idea in these algorithms is as follows. It follows from convexity considerations that either the constraints in a linear program are tight (i.e., satisfied with equality) at x^0, or they are irrelevant. We need identify only d linearly independent tight constraints to identify x^0. We do this by discarding a fixed proportion of the irrelevant constraints at each iteration. Determining whether the ith constraint is tight amounts to determining which case holds in $a_i \cdot x^0 \stackrel{>}{=} b_i$. This is embedded in a multidimensional search problem. Given *any* hyperplane $\alpha \cdot x = \beta$, we can determine which case of $\alpha \cdot x^0 \stackrel{\geqq}{<} \beta$ holds by (recursively) solving three linear programs in $d-1$ variables. These are (49.1.1) plus $\alpha \cdot x = \gamma$, where $\gamma \in \{\beta - \epsilon, \beta, \beta + \epsilon\}$ for "small" $\epsilon > 0$. (We need not define ϵ explicitly; it can be handled symbolically.) In each of the three linear programs we eliminate one variable to get $d-1$. The largest of the three objective functions tells us where x^0 lies with respect to the hyperplane. We call this an *inquiry* about $\alpha \cdot x = \beta$. The problem now reduces to locating x^0 with respect to a proportion $P(d)$ of the n hyperplanes using only $N(d)$ inquiries.

The method recursively uses the following observation in $\mathbb{R}^2$. Given two lines through the origin with slopes of opposite sign, knowing which quadrant a point lies in allows us to locate it with respect to *at least one* of the lines (see Figure 49.3.1).

We use this on the first two coordinates of the problem in $\mathbb{R}^d$. First rotate until $\frac{1}{2}n$ defining hyperplanes have positive and $\frac{1}{2}n$ negative "slopes" on these

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

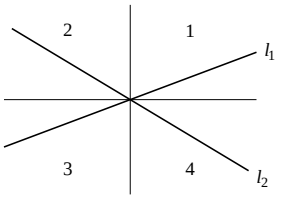

FIGURE 49.3.1
Quadrants $1, 3$ *locate for* l_2*; quadrants* $2, 4$ *locate for* l_1.

coordinates. This can be done in $O(n)$ time using median-finding. Then arbitrarily pair a positive with a negative to get $\frac{1}{2}n$ pairs of the form

$$\begin{aligned} ax_1 + bx_2 + \cdots &= \cdots \\ cx_1 - dx_2 + \cdots &= \cdots, \end{aligned}$$

where a, b, c, and d represent nonnegative numbers, and the $\cdots$ represent linear functions of $x_3, \ldots, x_d$ on the left and arbitrary numbers on the right. Eliminating x_2 and x_1 in each pair gives two families S_1, S_2 of $\frac{1}{2}n$ hyperplanes each in $d-1$ dimensions of the form

$$\begin{aligned} S_1: \quad x_1 \qquad + \cdots &= \cdots \\ S_2: \qquad x_2 \quad + \cdots &= \cdots. \end{aligned}$$

We recursively locate with respect to $\frac{1}{2}P(d-1)n$ hyperplanes with $N(d-1)$ inquiries in S_1, and then locate with respect to a $P(d-1)$-fraction of the corresponding paired hyperplanes in S_2. We have then located $\frac{1}{2}P(d-1)^2 n$ pairs with $2N(d-1)$ inquiries. Using the observation above, each pair gives us location with respect to at least one hyperplane in d dimensions, i.e.,

$$P(d) = \tfrac{1}{2}P(d-1)^2, \qquad N(d) = 2N(d-1). \tag{49.3.1}$$

Since $P(1) = \frac{1}{2}, N(1) = 1$ (by locating with respect to the median in $\mathbb{R}^1$), (49.3.1) yields

$$P(d) = 2^{-(2^d-1)}, \qquad N(d) = 2^{d-1},$$

giving the following time bound $T(n, d)$ for solving (49.1.1).

$$T(n, d) \le 3 \cdot 2^{d-1} T(n, d-1) + T((1 - 2^{-(2^d-1)})n, d) + O(nd),$$

with solution $T(n, d) < 2^{2^{d+2}} n$.

THE CLARKSON-DYER IMPROVEMENT

The Clarkson/Dyer improvement comes from repeatedly locating in S_1 and S_2 to increase $P(d)$ at the expense of $N(d)$.

49.4 RANDOMIZED ALGORITHMS

Dyer and Frieze [DF89] showed that, by applying an idea of Clarkson [Cla86a] to give a *randomized* solution of the multidimensional search in Megiddo's algorithm [Meg84], an algorithm of complexity $O(d^{3d+o(d)}n)$ was possible. Clarkson [Cla88, Cla95] improved this dramatically. We describe this below, but first outline a simpler algorithm subsequently given by Seidel [Sei91].

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

Suppose we order the constraints randomly. At stage k, we have solved the linear program subject to constraints $i = 1, \ldots, k-1$. We now wish to add constraint k. If it is satisfied by the current optimum we finish stage k and move to $k+1$. Otherwise, the new constraint is clearly tight at the optimum over constraints $i = 1, \ldots, k-1$. Thus, recursively solve the linear program subject to this equality (i.e., in dimension $d-1$) to get the optimum over constraints $i = 1, \ldots, k$, and move on to $k+1$. Repeat until $k = n$.

The analysis hinges on the following observation. When constraint k is added, the probability it is not satisfied is exactly d/k (assuming, without loss, nondegeneracy). This is because only d constraints are tight at the optimum and this is the probability of writing one of these *last* in a random ordering of $1, 2, \ldots, k$. This leads to an expected time of $O(d!n)$ for (49.1.1). Welzl [Wel91] extended Seidel's algorithm to solve other problems such as smallest enclosing ball or ellipsoid, and described variants that perform favorably in practice.

Sharir and Welzl [SW92] modified Seidel's algorithm, resulting in an improved running time of $O(d^3 2^d n)$. They put their algorithm in a general framework of solving "LP-type" problems (see Section 49.6 below). Matoušek, Sharir, and Welzl [MSW96] improved the analysis further, essentially obtaining the same bound as for Kalai's "primal simplex" algorithm. The algorithm was extended to LP-type problems by Gärtner [Gär95], with a similar time bound.

CLARKSON'S ALGORITHM

The basic idea is to choose a random set of r constraints, and solve the linear program subject to these. The solution will violate "few" constraints among the remaining $n - r$, and, moreover, one of these must be tight at x^0. We solve a new linear program subject to the violated constraints and a new random subset of the remainder. We repeat this procedure (aggregating the old violated constraints) until there are no new violated constraints, in which case we have found x^0. Each repetition gives an extra tight constraint for x^0, so we cannot perform more than d iterations.

Clarkson [Cla88] gave a different analysis, but using Seidel's idea we can easily bound the expected number of violated constraints (see also [GW01] for further simplifications of the algorithm). Imagine all the constraints ordered randomly, our sample consisting of the first r. For $i > r$, let $I_i = 1$ if constraint i is violated, $I_i = 0$ otherwise. Now $\Pr(I_i = 1) = \Pr(I_{r+1} = 1)$ for all $i > r$ by symmetry, and $\Pr(I_{r+1} = 1) = d/(r+1)$ from above. Thus the expected number of violated constraints is

$$\mathbf{E}\left(\sum_{i=r+1}^{n} I_i\right) = \sum_{i=r+1}^{n} \Pr(I_i = 1) = (n-r)d/(r+1) < nd/r.$$

(In the case of degeneracy, this will be an upper bound by a simple perturbation argument.)

Thus, if $r = \sqrt{n}$, say, there will be at most $d\sqrt{n}$ violated constraints in expectation. Hence, by Markov's inequality, with probability $\frac{1}{2}$ there will be at most $2d\sqrt{n}$ violated constraints in actuality. We must therefore recursively solve about $2(d+1)$ linear programs with at most $(2d^2+1)\sqrt{n}$ constraints. The "small" base cases can be solved by the simplex method in $d^{O(d)}$ time. This can now be applied

recursively, as in [Cla88], to give a bound for (49.1.1) of

$$O(d^2 n) + (\log n)^{\log d+2} d^{O(d)}.$$

Clarkson [Cla95] subsequently modified his algorithm using a different "iterative reweighting" algorithm to solve the $d+1$ small linear programs, obtaining a better bound on the execution time.

Each constraint receives an initial weight of 1. Random samples of total weight $10d^2$ (say) are chosen at each iteration, and solved by the simplex method. If W is the current total weight of all constraints, and W' the weight of the unsatisfied constraints, then $W' \leq 2Wd/10d^2 = W/5d$ with probability at least $\frac{1}{2}$ by the discussion above, regarding the weighted constraints as a multiset. We now double the weights of all violated constraints and repeat until there are no violated constraints (see [BG11] for a further simplification of this algorithm). This terminates in $O(d \log n)$ iterations by the following argument. After k iterations we have

$$W \leq \left(1 + \frac{1}{5d}\right)^k n \leq n e^{k/5d},$$

and W^*, the total weight of the d optimal constraints, satisfies $W^* \geq 2^{k/d}$, since at least one is violated at each iteration. Now it is clear that $W^* < W$ only while $k < Cd \ln n$, for some constant C. Applying this to the $d+1$ small linear programs gives overall complexity

$$O(d^2 n + d^4 \sqrt{n} \log n) + d^{O(d)} \log n.$$

This is almost the best expected time bound known for linear programming, except that Kalai's algorithm (or [MSW96]) can be used to solve the base cases rather than the simplex method. Then we get the improved bound (cf. [GW96])

$$O(d^2 n) + e^{O(\sqrt{d \log d})}.$$

This is polynomial for $d = O(\log^2 n / \log\log n)$, and is the best expected time bound to date.

49.5 DERANDOMIZED METHODS

Somewhat surprisingly, the randomized methods of Section 49.4 can also lead to the best *deterministic* algorithms for (49.1.1). Chazelle and Matoušek [CM96] produced a first derandomized version of Clarkson's algorithm.

The idea, which has wider application, is based on finding (in linear time) *approximations* to the constraint set. If N is a constraint set, then for each $x \in \mathbb{R}^d$ let $V(x, N)$ be the set of constraints violated at x. A set $S \subseteq N$ is an ϵ-approximation to N if, for all x,

$$\left| \frac{|V(x,S)|}{|S|} - \frac{|V(x,N)|}{|N|} \right| < \epsilon.$$

(See also Chapters 40 and 47.) Since $n = |N|$ hyperplanes partition $\mathbb{R}^d$ into only $O(n^d)$ regions, there is essentially only this number of possible cases for x, i.e.,

only this number of different sets $V(x, N)$. It follows from the work of Vapnik and Chervonenkis that a (d/r)-approximation of size $O(r^2 \log r)$ always exists, since a random subset of this size has the property with nonzero probability. If we can find such an approximation deterministically, then we can use it in Clarkson's algorithm in place of random sampling. If we use a (d/r)-approximation, then, if x^* is the linear programming optimum for the subset S, $|V(x^*, S)| = 0$, so that

$$|V(x^*, N)| < |N|d/r = nd/r,$$

as occurs in expectation in the randomized version. The implementation involves a refinement based on two elegant observations about approximations, which both follow directly from the definition.

(i) An ϵ-approximation of a δ-approximation is an $(\epsilon + \delta)$-approximation of the original set.

(ii) If we partition N into q equal sized subsets $N_1, \ldots, N_q$ and take an (equal sized) ϵ-approximation S_i in each N_i $(i = 1, \ldots, q)$, then $S_1 \cup \ldots \cup S_q$ is an ϵ-approximation of N.

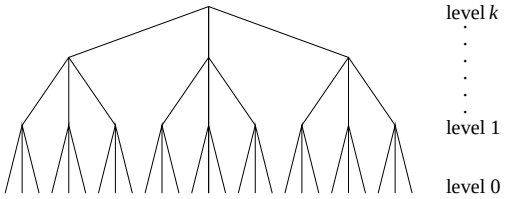

FIGURE 49.5.1
A partition tree of height k, with $q = 3$.

We then *recursively* partition N into q equal sized subsets, to give a "partition tree" of height k, say, as in Figure 49.5.1 (cf. Section 40.2). The sets at level 0 in the partition tree are "small." We calculate an ϵ_0-approximation in each. We now take the union of these approximations at level 1 and calculate an ϵ_1-approximation of this union. This is an $(\epsilon_0 + \epsilon_1)$-approximation of the whole level 1 set, by the above observations. Continuing up the tree, we obtain an overall $(\sum_{i=0}^{k} \epsilon_i)$-approximation of the entire set. At each stage, the sets on which we have to find the approximations remain "small" if the ϵ_i are suitably chosen. Therefore we can use a relatively inefficient method of finding an approximation. A suitable method is the *method of conditional probabilities* due to Raghavan and Spencer. It is (relatively) straightforward to implement this on a set of size m to run in $O(m^{d+1})$ time. However, since this has to be applied only to small sets (in comparison with n), the total work can be bounded by a linear function of n. Chazelle and Matoušek [CM96] used $q = 2$, and an ϵ_i that corresponds to roughly halving the union at each level $i = 1, \ldots, k$.

The algorithm cannot completely mimic Clarkson's, however, since we can no longer use $r = \sqrt{n}$. Such a large approximation cannot be determined in linear time by the above methods. But much smaller values of r suffice (e.g., $r = 10d^3$) simply to get *linear-time* behavior in the recursive version of Clarkson's algorithm. Using this observation, Chazelle and Matoušek [CM96] obtained a deterministic algorithm with time-complexity $d^{O(d)}n$. As far as the asymptotics of the exponent is concerned, this is still the best deterministic time bound known for solving (49.1.1), and it remains polynomial for $d = O(\log n / \log \log n)$. Later research has improved

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

the constant in the exponent. The currently best (and most simple) approach is due to Chan [Cha16].

Recall that with $r = 10d^3$, Chazelle and Matoušek's algorithm as sketched above computes an $O(1/d^2)$-approximation S of the constraint set. Actually, an $O(1/d^2)$-net S suffices. An ϵ-net has the ϵ-approximation property only for vectors x satisfying $V(x, S) = \emptyset$. But this is enough, as (the derandomized version of) Clarkson's algorithm solves the problem subject to the constraints in S first, and then only keeps adding constraints. Hence, all solutions x coming up during the algorithm satisfy $V(x, S) = \emptyset$.

Chan's observation is that the $O(1/d^2)$-net does not have to be of optimal size, hence one might be able to employ a simpler and faster deterministic construction algorithm. Indeed, as long as the subproblems solved in the recursive calls of Clarkson's algorithm are by a factor of $2d$, say, smaller than the original problem, the linear-time analysis still works.

Chan's main contribution is a simple and fast deterministic method to compute an $O(1/d^2)$-net of size roughly $n/(2d)$. For this, he first provides a very simple and fast greedy method to compute an ϵ-net for a set of k constraints, of size $O((d/\epsilon)\log k)$. This is clearly suboptimal, as there are ϵ-nets whose size is independent of k. The trick now is to subdivide the constraint set arbitrarily into $n/f(d)$ groups of size $f(d)$, for a suitable polynomial f. For each group, the fast algorithm is applied with $k = f(d)$, meaning that the size penalty is only logarithmic in d, per group. The final ϵ-net is simply the union of the ϵ-nets for the groups.

The resulting algorithm has runtime $O(d^{3d}n)$, up to $(\log d)^{O(d)}$ factors. A further reduction to $O(d^{d/2}n)$, again up to $(\log d)^{O(d)}$ factors, is obtained through a derandomization of a new variant of Clarkson's second algorithm, and improved ϵ-net constructions. Moreover, this seems to reach a natural barrier in the sense that an exponent of $d/2$ is already incurred by the simplex algorithm that is used as a subroutine for linear programs with $O(d^2)$ constraints.

49.6 LP-TYPE PROBLEMS

The randomized algorithms above by Clarkson and in [MSW96, Gär95] can be formulated in an abstract framework called *LP-type problems*. With an extra condition (involving VC-dimension of certain set-systems) this extends to the derandomization in [CM96]. In this way, the algorithms are applicable to a host of problems including smallest enclosing ball, polytope distance, smallest enclosing ellipsoid, largest ellipsoid in polytope, smallest ball intersecting a set of convex objects, angle-optimal placement in polygon, rectilinear 3-centers in the plane, spherical separability, width of thin point sets in the plane, and integer linear programming (see [MSW96, GW96] for descriptions of these problems and of the reductions needed). A different abstraction called *abstract objective functions* is described by Kalai in [Kal97], and for the even more general setting of *abstract optimization problems* see [Gär95].

For the definitions below, the reader should think of optimization problems in which we are given some set H of constraints and we want to *minimize* some given function under those constraints. For every subset G of H, let $w(G)$ denote the optimum value of this function when all constraints in G are satisfied. The function w is only given implicitly via some basic operations to be specified below. The goal

is to compute an inclusion-minimal subset B_H of H with the same value as H (from which, in general, the value is easy to determine).

GLOSSARY

LP-type problem: A pair (H, w), where H is a finite set and $w : 2^H \to \mathcal{W}$ for a linearly ordered set $(\mathcal{W}, \leq)$ with a minimal element $-\infty$, so that the monotonicity and locality axioms below are satisfied.

Monotonicity axiom: For any F, G with $F \subseteq G \subseteq H$, we have $w(F) \leq w(G)$.

Locality axiom (for LP-type problems): For any $F \subseteq G \subseteq H$ with $-\infty \neq w(F) = w(G)$ and for any $h \in H$, $w(G) < w(G \cup \{h\})$ implies $w(F) < w(F \cup \{h\})$.

Constraints of LP-type problem: Given an LP-type problem (H, w), the elements of H are called constraints.

Basis: A set B of constraints is called a basis if $w(B') < w(B)$ for every proper subset of B.

Basis of set of constraints: Given a set G of constraints, a subset $B \subseteq G$ is called a basis of G if it is a basis and $w(B) = w(G)$ (i.e., an inclusion-minimal subset of G with equal w-value).

Combinatorial dimension: The maximum cardinality of any basis in an LP-type problem (H, w), denoted by $\delta = \delta_{(H,w)}$.

Basis regularity: An LP-type problem (H, w) is basis-regular if, for every basis B with $|B| = \delta$ and for every constraint h, all bases of $B \cup \{h\}$ have exactly δ elements.

Violation test: Decides whether or not $w(B) < w(B \cup \{h\})$, for a basis B and a constraint h.

Basis computation: Delivers a basis of $B \cup \{h\}$, for a basis B and a constraint h.

Violator space: A pair $(H, \mathcal{V})$, where H is a finite set and $\mathcal{V} : 2^H \to 2^H$ such that the consistency and locality axioms below are satisfied.

Consistency axiom: For any $G \subseteq H$, we have $G \cap \mathcal{V}(G) = \emptyset$.

Locality axiom (for violator spaces): For any $F \subseteq G \subseteq H$, $G \cap \mathcal{V}(F) = \emptyset$ implies $\mathcal{V}(F) = \mathcal{V}(G)$.

Unique Sink Orientation: Orientation of the n-dimensional hypercube graph such that every subgraph induced by a nonempty face has a unique sink.

A simple example of an LP-type problem is the smallest enclosing ball problem (this problem traces back to J.J. Sylvester [Syl57]): Let S be a finite set of points in $\mathbb{R}^d$, and for $G \subseteq S$, let $\rho(G)$ be the radius of the ball of smallest volume containing G (with $\rho(\emptyset) = -\infty$). Then (S, ρ) is an LP-type problem with combinatorial dimension at most $d + 1$. A violation test amounts to a test deciding whether a point lies in a given ball, while an efficient implementation of basis computations is not obvious (cf. [Gär95]).

Many more examples have been indicated above. As the name suggests, linear programming can be formulated as an LP-type problem, although some care is needed in the presence of degeneracies. Let us assume that we want to maximize the objective function $-x_d$ in (49.1.1), i.e., we are looking for a point in $\mathbb{R}^d$ of smallest x_d-coordinate. In the underlying LP-type problem, the set H of constraints is given

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

by the halfspaces as defined by (49.1.2). For a subset G of these constraints, let $v(G)$ be the backwards lexicographically smallest point satisfying these constraints, with $v(G) := -\infty$ if G gives rise to an unbounded problem, and with $v(G) := \infty$ in case of infeasibility. We assume the backwards lexicographical ordering on $\mathbb{R}^d$ to be extended to $\mathbb{R}^d \cup \{-\infty, \infty\}$ by letting $-\infty$ and ∞ be the minimal and maximal element, resp. The resulting pair (H, v) is LP-type of combinatorial dimension at most $d + 1$. In fact, if the problem is feasible and bounded, then the LP-type problem is basis-regular of combinatorial dimension d. The violation test and basis computation (this amounts to a dual pivot step) are easy to implement.

Matoušek, Sharir, and Welzl [MSW96] showed that a basis-regular LP-type problem (H, w) of combinatorial dimension δ with n constraints can be solved (i.e., a basis of H can be determined) with an expected number of at most

$$\min\{e^{2\sqrt{\delta \ln((n-\delta)/\sqrt{\delta})}+O(\sqrt{\delta}+\ln n)}, 2^{\delta+2}(n-\delta)\} \tag{49.6.1}$$

violations tests and basis computations, provided an initial basis B_0 with $|B_0| = \delta$ is available. (For linear programming one can easily generate such an initial basis by adding d symbolic constraints at "infinity.") Then Gärtner [Gär95] generalized this bound to all LP-type problems. Combining this with Clarkson's methods, one gets a bound (cf. [GW96]) of

$$O(\delta n) + e^{O(\sqrt{\delta \log \delta})}$$

for the expected number of violation tests and basis computations, the best bound known up to now. In the important special case where $n = O(\delta)$, the resulting bound of $e^{O(\sqrt{\delta \log \delta})}$ can be improved to $e^{O(\sqrt{\delta})}$ by using a dedicated algorithm for this case [Gär95]. Recently, Hansen and Zwick [HZ15] have developed and analyzed a new variant of the general algorithm, essentially replacing the denominator of $\sqrt{\delta}$ under the root in (49.6.1) with δ. For $n = O(\delta)$, this also results in an improved bound of $e^{O(\sqrt{\delta})}$.

Matoušek [Mat94] provided a family of LP-type problems, for which the bound (49.6.1) is almost tight for the algorithm provided in [MSW96], for the case $n = 2d$. It was an open problem, though, whether the algorithm performs faster when applied to linear programming instances. In fact, Gärtner [Gär02] showed that the algorithm is quadratic on the instances in Matousek's lower bound family which are realizable as linear programming problems as in (49.1.1). Only the recent subexponential lower bound for the random facet simplex algorithm due to Friedmann, Hansen and Zwick [FHZ11] implies that for $n = 2d$, the bound (49.6.1) is tight up to a possible replacement of the square root by a different root.

Amenta [Ame94] considers the following extension of the abstract framework: Suppose we are given a family of LP-type problems (H, w_λ), parameterized by a real parameter λ; the underlying ordered value set $\mathcal{W}$ has a maximum element ∞ representing *infeasibility*. The goal is to find the smallest λ for which (H, w_λ) is feasible, i.e., $w_\lambda(H) < \infty$. [Ame94] provides conditions under which such a problem can be transformed into a single LP-type problem, and she gives bounds on the resulting combinatorial dimension. This work exhibits interesting relations between LP-type problems and Helly-type theorems (see also [Ame96]).

An interesting combinatorial generalization of LP-type problems that removes the objective function w has been introduced by Škovroň [Ško07]. A *violator space* is a pair $(H, \mathcal{V})$, where H is a finite set that we again think of as being constraints.

$\mathcal{V} : 2^H \to 2^H$ is a function that assigns to each subset G of constraints a set $\mathcal{V}(G)$ of constraints violated by (the solution subject to the constraints in) G. $\mathcal{V}$ is required to satisfy *consistency* ($G \cap \mathcal{V}(G) = \emptyset$ for all G) as well as *locality* (if $F \subseteq G$ and $G \cap \mathcal{V}(F) = \emptyset$, then $\mathcal{V}(F) = \mathcal{V}(G)$). If (H, w) is an LP-type problem (without a set of value $-\infty$) and $\mathcal{V}(G) := \{h \in H \setminus G : w(G \cup \{h\}) > w(G)\}$, then $(H, \mathcal{V})$ is a violator space. A variant of the definition that covers $-\infty$ also exists [Ško07].

It turns out that violator spaces form in a well-defined sense the most general framework in which Clarkson's random-sampling based methods still work [GMRS08, BG11]. In this sense, violator spaces form a "combinatorial core" of linear programming, useful for theoretical considerations. The concept properly generalizes LP-type problems: in the absence of an objective function, simplex-type algorithms may cycle even in nondegenerate situations [GMRS08]. In the applications, however, most violator spaces can actually be cast as LP-type problems.

UNIQUE SINK ORIENTATIONS

As seen in Section 49.4, the currently best "strong" bound for linear programming is

$$O(d^2 n) + e^{O(\sqrt{d \log d})}$$

in expectation. In order to further improve the (subexponential) dependence on d, one would have to make progress on "small" problems, most notably the case $n = O(d^2)$.

A natural and important but poorly understood special case is that of linear programs whose feasible set is a (deformed) d-dimensional cube. Here, $n = 2d$.

The combinatorial framework of unique sink orientations (USO) has been introduced to deal with such small problems; in fact, there is only one parameter, namely n. A USO is an orientation of the n-dimensional hypercube graph, with the property that every subgraph induced by a nonempty face of the cube has a unique sink [SW01].

The algorithmic problem is the following: given a *vertex evaluation* oracle that for a given vertex returns the orientations of the n incident edges, how many vertex evaluations need to be performed in order to find the unique global sink?

Under suitable nondegeneracy assumptions, a linear program over a (deformed) cube directly yields an acyclic USO; from its global sink, one can read off the solution to the linear program. But in fact, any linear program with n inequality constraints reduces to sink-finding in a (possibly cyclic) n-cube USO [GS06].

Simplex-type methods (following a directed path in the USO) are the most natural sink-finding strategies; in fact, the idea behind the subexponential algorithm for LP-type problems [MSW96] can be understood in its purest form on acyclic USO [Gär02]. This, however, does not imply any new strong bounds for linear programming.

But the cube structure also allows for algorithms that "jump," meaning that they do not necessarily follow a path in the cube graph but visit cube vertices in a random access fashion. The Fibonacci Seesaw [SW01] is a deterministic jumping algorithm that is able to find the sink of any n-cube USO with $O(1.61^n)$ vertex evaluations. Via the reduction to USO, this yields the best known deterministic strong bound for linear programs with $n = 2d$ constraints [GS06]. Most notably, this bound is smaller than the worst-case number of vertices, all of which the known deterministic simplex algorithms might have to visit.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

49.7 PARALLEL ALGORITHMS

GLOSSARY

PRAM: Parallel Random Access Machine. (See Section 46.1 for more information on this and the next two terms.)

EREW: Exclusive Read Exclusive Write.

CRCW: Concurrent Read Concurrent Write.

P: The class of polynomial time problems.

NC: The class of problems that have poly-logarithmic parallel time algorithms running a polynomial number of processors.

P-*complete problem:* A problem in P whose membership in NC implies P = NC.

Expander: A graph in which, for every set of nodes, the set of the neighbors of the nodes is relatively large.

We will consider only PRAM algorithms. (See also Section 46.2.)

The general linear programming problem has long been known to be P-complete, so there is little hope of very fast parallel algorithms. However, the situation is different in the case $d \ll n$, where the problem is in NC if d grows slowly enough.

First, we note that there is a straightforward parallel implementation of Megiddo's algorithm [Meg83b] that runs in $O((\log n)^d)$ time on an EREW PRAM. However, this algorithm is rather inefficient in terms of processor utilization, since at the later stages, when there are few constraints remaining, most processors are idle. However, Deng [Den90] gave an "optimal" $O(n)$ work implementation in the plane running in $O(\log n)$ time on a CRCW PRAM with $O(n/\log n)$ processors. Deng's method does not seem to generalize to higher dimensions.

Alon and Megiddo [AM94] gave a *randomized* parallel version of Clarkson's algorithm which, with high probability, runs in constant time on a CREW PRAM in fixed dimension. Here the "constant" is a function of dimension only, and the probability of failure to meet the time bound is small for $n \gg d$.

Ajtai and Megiddo [AM96] attempted to improve the processor utilization in parallelizing Megiddo's algorithm for general d. They gave an intricate algorithm based on using an expander graph to select more nondisjoint pairs so as to utilize all the processors and obtain more rapid elimination. The resulting algorithm for (49.1.1) runs in $O((\log\log n)^d)$ time, but in a nonuniform model of parallel computation based on Valiant's comparison model. The model, which is stronger than the CRCW PRAM, requires $O(\log\log n)$ time median selection from n numbers using n processors, and employs an $O(\log\log n)$ time scheme for compacting the data after deletions, again based on a nonuniform use of expander graphs. A lower bound of $\Omega(\log n/\log\log n)$ time for median-finding on the CRCW PRAM follows from results of Beame and Håstad. Thus Ajtai and Megiddo's algorithm could not be implemented directly on the CRCW PRAM. Within Ajtai and Megiddo's model there is a lower bound $\Omega(\log\log n)$ for the case $d = 1$ implied by results of Valiant. This extends to the CRCW PRAM, and is the only lower bound known for solving (49.1.1) in this model.

Dyer [Dye95] gave a different parallelization of Megiddo's algorithm, which avoids the use of expanders. The method is based on forming groups of size $r \geq 2$,

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

rather than simple pairs. As constraints are eliminated, the size of the groups is gradually increased to utilize the extra processors. Using this, Dyer [Dye95] establishes an $O(\log n(\log\log n)^{d-1})$ bound in the EREW model. It is easy to show that there is an $\Omega(\log n)$ lower bound for solving (49.1.1) on the EREW PRAM, even with $d = 1$. (See [KR90].) Thus improvements on Dyer's bound for the EREW model can only be made in the $\log\log n$ term. However, there was still an open question in the CRCW model, since exact median-finding and data compaction cannot be performed in time polynomial in $\log\log n$.

Goodrich [Goo93] solved these problems for the CRCW model by giving fast implementations of derandomization techniques similar to those outlined in Section 49.5. However, the randomized algorithm that underlies the method is not a parallelization of Clarkson's algorithm, but is similar to a parallelized version of that of Dyer and Frieze [DF89]. He achieves a work-optimal (i.e., $O(n)$ work) algorithm running in $O(\log\log n)^d$ time on the CRCW PRAM. The methods also imply a work-optimal EREW algorithm, but only with the same time bound as Dyer's. Neither Dyer nor Goodrich is explicit about the dependence on d of the execution time of their algorithms.

Independently of Goodrich's work, Sen [Sen95] has shown how to directly modify Dyer's algorithm to give a work-optimal algorithm with $O((\log\log n)^{d+1})$ execution time in the CRCW model. The "constant" in the running time is shown to be $2^{O(d^2)}$. To achieve this, he uses approximate median-finding and approximate data compaction operations, both of which can be done in time polynomial in $\log\log n$ on the common CRCW PRAM. These additional techniques are, in fact, both examples of derandomized methods and similar to those Goodrich uses for the same purpose. Note that this places linear programming in NC provided $d = O(\sqrt{\log n})$. This is the best result known, although Goodrich's algorithm may give a better behavior once the "constant" has been explicitly evaluated. We may also observe that the Goodrich/Sen algorithms improve on Deng's result in $\mathbb{R}^2$.

There is still room for some improvements in this area, but there now seems to be a greater need for sharper lower bounds, particularly in the CRCW case.

49.8 RELATED ISSUES

GLOSSARY

Integer programming problem: A linear programming problem with the additional constraint that the solution must be integral.

k-Violation linear programming: A problem as in (49.1.1), except that we want to maximize the linear objective function subject to all but at most k of the given linear constraints.

Average case analysis: Expected performance of an algorithm for random input (under certain distributions).

Smoothed analysis: Expected performance of an algorithm under small random perturbations of the input.

Linear programming is a problem of interest in its own right, but it is also representative of a class of geometric problems to which similar methods can be

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

applied. Many of the references given below discuss closely related problems, and we have mentioned them in passing above.

An important related area is integer programming. Here the size of the numbers cannot be relegated to a secondary consideration. In general this problem is NP-hard, but in fixed dimension is polynomial-time solvable. See [Sch86] for further information. It may be noted that Clarkson's methods and the LP-type framework are applicable in this situation; some care with the interpretation of the primitive operations is in place, though.

We have considered only the solution of a single linear program. However, there are some situations where one might wish to solve a sequence of closely related linear programs. In this case, it may be worth the computational investment of building a data structure to facilitate fast solution of the linear programs. For results of this type see, for example, [Epp90, Mat93, Cha96, Cha98].

Finally there has been some work about optimization, where we are asked to satisfy all but at most k of the given constraints, see, e.g., [RW94, ESZ94, Mat95b, DLSS95, Cha99]. In particular, Matoušek [Mat95a] has investigated this question in the general setting of LP-type problems. Chan [Cha02] solved this problem in $\mathbb{R}^2$ with a randomized algorithm in expected time $O((n+k^2)\log k)$ (see this paper for the best bounds known for $d = 3, 4$.)

A direction we did not touch upon here is *average-case analysis*, where we analyze a deterministic algorithm for random inputs [Bor87, Sma83, AM85, Tod86, AKS87]. In the latter three, bounds are proven with respect to classes of probability distributions, where the numerical entries of A, b, and c are fixed in a nondegenerate way and the directions of inequalities are picked at random. More recently, there has been an interesting new direction, smoothed analysis, where a simplex algorithm is analyzed for small random perturbations of the input. The main result of Spielmann and Teng [ST04] is that (superpolynomial) worst-case instances for the shadow vertex simplex algorithm are "isolated." This means that after small random perturbations of the input, the algorithm will be fast in expectation. Indeed, if one looks at the geometry of the known (contrived) worst-case instances, one sees 2-dimensional projections ("shadows") with exponentially many vertices, tightly packed along the boundary of a convex polygon. It is not surprising that such "bad" projections disappear even under small random perturbations of the input. What is more surprising—and this is the main contribution of Spielmann and Teng—is that not only these bad instances, but *all* instances have small shadows after random perturbations.

49.9 SOURCES AND RELATED MATERIAL

BOOKS AND SURVEYS

Linear programming is a mature topic, and its foundations are well-covered in classic textbooks. A good general introduction to linear programming may be found in Chvátal's book [Chv83]. A theoretical treatment is given in Schrijver's book [Sch86]. The latter is a very good source of additional references. The book by Matoušek and Gärtner [MG07] is a concise introduction from a computer science perspective. Karp and Ramachandran [KR90] is a good source of information on models of

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

parallel computation. See [Mat96] for a survey of derandomization techniques for computational geometry.

RELATED CHAPTERS

Chapter 15: Basic properties of convex polytopes
Chapter 19: Polytope skeletons and paths
Chapter 36: Computational convexity
Chapter 46: Parallel algorithms in geometry
Chapter 67: Software

REFERENCES

[AF92] D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete Comput. Geom.*, 8:295–313, 1992.

[AF16] D. Avis and O. Friedmann. An exponential lower bound for Cunningham's rule. *Math. Program. Ser. A*, to appear.

[AKS87] I. Adler, R.M. Karp, and R. Shamir. A simplex variant solving an $(m \times d)$ linear program in $O(\min(m^2, d^2))$ expected number of pivot steps. *J. Complexity*, 3:372–387, 1987.

[AM85] I. Adler and N. Megiddo. A simplex algorithm whose average number of steps is bounded between two quadratic functions of the smaller dimension. *J. ACM*, 32:871–895, 1985.

[AM94] N. Alon and N. Megiddo. Parallel linear programming in fixed dimension almost surely in constant time. *J. ACM*, 41:422–434, 1994.

[AM96] M. Ajtai and N. Megiddo. A deterministic poly$(\log \log n)$-time n-processor algorithm for linear programming in fixed dimension. *SIAM J. Comput.*, 25:1171–1195, 1996.

[Ame94] N. Amenta. Helly-type theorems and generalized linear programming. *Discrete Comput. Geom.*, 12:241–261, 1994.

[Ame96] N. Amenta. A new proof of an interesting Helly-type theorem. *Discrete Comput. Geom.*, 15:423–427, 1996.

[AZ99] N. Amenta and G.M. Ziegler. Deformed products and maximal shadows. In B. Chazelle, J.E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, vol. 223 of *Contemp. Math.*, pages 57–90, AMS, Providence, 1998.

[BG11] Y. Brise and B. Gärtner. Clarkson's algorithm for violator spaces. *Comput. Geom.*, 44:70–81, 2011.

[Bor87] K.H. Borgwardt. *The Simplex Method: A Probabilistic Analysis.* Vol. 1 of *Algorithms Combin.*, Springer-Verlag, Berlin, 1987.

[Cha96] T.M. Chan. Fixed-dimensional linear programming queries made easy. In *Proc. 12th Sympos. Comput. Geom.*, pages 284–290, ACM Press, 1996.

[Cha98] T.M. Chan. Deterministic algorithms for 2-d convex programming and 3-d online linear programming. *J. Algorithms*, 27:147–166, 1998.

[Cha99] T.M. Chan. Geometric applications of a randomized optimization technique. *Discrete Comput. Geom.*, 22:547–567, 1999.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

[Cha02] T.M. Chan. Low-dimensional linear programming with violations. In *Proc. 43rd IEEE Sympos. Found. Comput. Sci.*, pages 570–579, 2002.

[Cha16] T.M. Chan. Improved deterministic algorithms for linear programming in low dimensions. In *Proc. 27th ACM-SIAM Sympos. Discrete Algorithms*, pages 1213–1219, 2016.

[Chv83] V. Chvátal. *Linear Programming*. Freeman, New York, 1983.

[Cla86a] K.L. Clarkson. Further applications of random sampling to computational geometry. In *Proc. 18th ACM Sympos. Theory Comput.*, pages 414–423, 1986.

[Cla86b] K.L. Clarkson. Linear programming in $O(n3^{d^2})$ time. *Inform. Process. Lett.*, 22:21–24, 1986.

[Cla88] K.L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. In *Proc. 29th IEEE Sympos. Found. Comput. Sci.*, pages 452–456, 1988.

[Cla95] K.L. Clarkson. Las Vegas algorithms for linear and integer programming when the dimension is small. *J. ACM*, 42:488–499, 1995. (Improved version of [Cla88].)

[CM96] B. Chazelle and J. Matoušek. On linear-time deterministic algorithms for optimization in fixed dimension. *J. Algorithms* 21:579–597, 1996.

[Den90] X. Deng. An optimal parallel algorithm for linear programming in the plane. *Inform. Process. Lett.*, 35:213–217, 1990.

[DF89] M.E. Dyer and A.M. Frieze. A randomized algorithm for fixed-dimensional linear programming. *Math. Program.*, 44:203–212, 1989.

[DLSS95] 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.

[Dye83] M.E. Dyer. The complexity of vertex enumeration methods. *Math. Oper. Res.*, 8:381–402, 1983.

[Dye84] M.E. Dyer. Linear time algorithms for two- and three-variable linear programs. *SIAM J. Comput.*, 13:31–45, 1984.

[Dye86] M.E. Dyer. On a multidimensional search problem and its application to the Euclidean one-centre problem. *SIAM J. Comput.*, 15:725–738, 1986.

[Dye92] M.E. Dyer. A class of convex programs with applications to computational geometry. In *Proc. 8th Sympos. Comput. Geom.*, pages 9–15, ACM Press, 1992.

[Dye95] M.E. Dyer. A parallel algorithm for linear programming in fixed dimension. In *Proc. 11th Sympos. Comput. Geom.*, pages 345–349, ACM Press, 1995.

[Epp90] D. Eppstein. Dynamic three-dimensional linear programming. *ORSA J. Comput.*, 4:360–368, 1990.

[ESZ94] A. Efrat, M. Sharir, and A. Ziv. Computing the smallest k-enclosing circle and related problems. *Comput. Geom.*, 4:119–136, 1994.

[FHZ11] O. Friedmann, T.D. Hansen, and U. Zwick. Subexponential lower bounds for randomized pivoting rules for the simplex algorithm. In *Proc. 43rd ACM Sympos. Theory Comput.*, pages 283–292, 2011.

[Fri11] O. Friedmann. A subexponential lower bound for Zadehs pivoting rule for solving linear programs and games. In *Proc. 15th Conf. Integer Program. Combin. Opt.*, pages 192–206, vol. 6655 of *LNCS*, Springer, Berlin, 2011.

[Gär95] B. Gärtner. A subexponential algorithm for abstract optimization problems. *SIAM J. Comput.*, 24:1018–1035, 1995.

[Gär02] B. Gärtner. The random-facet simplex algorithm on combinatorial cubes. *Random Structures Algorithms*, 20:353–381, 2002.

[GLS93] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, 2nd corrected edition. Vol. 2 of *Algorithms and Combinatorics*, Springer, Berlin, 1993

[GMRS08] B. Gärtner, J. Matoušek, L. Rüst, and P. Škovroň. Violator spaces: Structure and algorithms. *Discrete Appl. Math.*, 156:2124–2141, 2008.

[Goo93] M.T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proc. 9th Sympos. Comput. Geom.*, pages 73–82, ACM Press, 1993.

[GS06] B. Gärtner and I. Schurr. Linear programming and unique sink orientations. In *Proc. 17th ACM-SIAM Sympos. Discrete Algorithms*, pages 749–757, 2006

[GW01] B. Gärtner and E. Welzl. A simple sampling lemma: Analysis and applications in geometric optimization. *Discrete Comput. Geom.*, 25:569–590, 2001.

[GW96] B. Gärtner and E. Welzl. Linear programming – randomization and abstract frameworks. In *Proc. 13th Sympos. Theoret. Aspects Comp. Sci.*, vol. 1046 of *LNCS*, pages 669–687, Springer, Berlin, 1996.

[HZ15] T.D. Hansen and U. Zwick. An improved version of the random-facet pivoting rule for the simplex algorithm. In *Proc. 47th ACM Sympos. Theory Comput.*, pages 209–218, 2015.

[Kal92] G. Kalai. A subexponential randomized simplex algorithm. In *Proc. 24th ACM Sympos. Theory Comput.*, pages 475–482, 1992.

[Kal97] G. Kalai. Linear programming, the simplex algorithm and simple polytopes. *Math. Program.*, 79:217–233, 1997.

[KM72] V. Klee and G.J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities III*, pages 159–175, Academic Press, New York, 1972.

[KR90] R.M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. A: Algorithms and Complexity*, pages 869–941, Elsevier, Amsterdam, 1990.

[Mat93] J. Matoušek. Linear optimization queries. *J. Algorithms*, 14:432–448, 1993.

[Mat94] J. Matoušek. Lower bounds for a subexponential optimization algorithm. *Random Structures Algorithms*, 5:591–607, 1994.

[Mat95a] J. Matoušek. On geometric optimization queries with few violated constraints. *Discrete Comput. Geom.*, 14:365–384, 1995.

[Mat95b] J. Matoušek. On enclosing k points by a circle. *Inform. Process. Lett.*, 53:217–221, 1995.

[Mat96] J. Matoušek. Derandomization in computational geometry. *J. Algorithms*, 20:545–580, 1996.

[Meg83a] N. Megiddo. Towards a genuinely polynomial algorithm for linear programming. *SIAM J. Comput.*, 12:347–353, 1983.

[Meg83b] N. Megiddo. Linear time algorithms for linear programming in $\mathbb{R}^3$ and related problems. *SIAM J. Comput.*, 12:759–776, 1983.

[Meg84] N. Megiddo. Linear programming in linear time when dimension is fixed. *J. ACM*, 31:114–127, 1984.

[Meg89] N. Megiddo. On the ball spanned by balls. *Discrete Comput. Geom.*, 4:605–610, 1989.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.

[MG07] J. Matoušek and B. Gärtner. *Understanding and Using Linear Programming*. Springer, Berlin, 2007.

[MP78] D.E. Muller and F.P. Preparata. Finding the intersection of two convex polyhedra. *Theoret. Comput. Sci.*, 7:217–236, 1978.

[MSW96] J. Matoušek, M. Sharir, and E. Welzl. A subexponential bound for linear programming. *Algorithmica*, 16:498–516, 1996.

[RW94] T. Roos and P. Widmayer. k-violation linear programming. *Inform. Process. Lett.*, 52:109–114, 1994.

[Sch86] A. Schrijver. *Introduction to the Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.

[Sei91] R. Seidel. Low dimensional linear programming and convex hulls made easy. *Discrete Comput. Geom.*, 6:423–434, 1991.

[Sen95] S. Sen. A deterministic poly$(\log \log n)$ time optimal CRCW PRAM algorithm for linear programming in fixed dimension. Technical Report 95-08, Dept. of Comp. Sci., Univ. of Newcastle, Australia, 1995.

[Sha78] M.I. Shamos. *Computational Geometry*. PhD thesis, Yale Univ., New Haven, 1978.

[Ško07] P. Škovroň. *Abstract Models of Optimization Problems*. PhD thesis, Charles University, Prague, 2007.

[Sma83] S. Smale. On the average number of steps in the simplex method of linear programming. *Math. Program.*, 27:241–262, 1983.

[ST04] D.A. Spielman and S.-H. Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51:385–463, 2004.

[SW92] M. Sharir and E. Welzl. A combinatorial bound for linear programming and related problems. In *Proc. 9th Sympos. Theoret. Aspects Comput. Sci.*, vol. 577 of *LNCS*, pages 569–579, Springer, Berlin, 1992.

[SW01] T. Szabó and E. Welzl. Unique sink orientations of cubes. In *Proc. 42nd IEEE Sympos. Found. Comp. Sci.*, pages 547–555, 2001.

[Syl57] J.J. Sylvester. A question in the geometry of situation. *Quart. J. Math.*, 1:79, 1857.

[Tar86] É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Oper. Res.*, 34:250–256, 1986.

[Tod86] M.J. Todd. Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems. *Math. Program.*, 35:173–192, 1986.

[Wel91] E. Welzl. Smallest enclosing disks (balls and ellipsoids). In H. Maurer, editor, *New Results and New Trends in Computer Science*, vol. 555 of *LNCS*, pages 359–370, Springer, Berlin, 1991.

Preliminary version (August 10, 2017). To appear in the Handbook of Discrete and Computational Geometry, J.E. Goodman, J. O'Rourke, and C. D. Tóth (editors), 3rd edition, CRC Press, Boca Raton, FL, 2017.