A Las Vegas Algorithm for Linear Programming When the Dimension is Small

Kenneth L. Clarkson
AT&T Bell Laboratories
Murray Hill, New Jersey 07974

Abstract

This paper gives an algorithm for solving linear programming problems. For a problem with n constraints and d variables, the algorithm requires an expected

$$O(d^2 n) + (\lg(n/d^2))^{\lg d+2} O(d)^{d/2+O(1)},$$

arithmetic operations, as $n \to \infty$. The constant factors do not depend on d. The expectation is with respect to the random choices made by the algorithm, and the bound holds for any given input. The technique can be extended to other convex programming problems. For example, an algorithm for finding the smallest sphere enclosing a set of n points in E^d has the same time bound.

1 Introduction

In some applications of linear and quadratic programming, the number of variables will be small. Such applications include Chebyshev approximation, linear separability, and the smallest enclosing circle problem. Megiddo [Meg84] gave an algorithm for linear programming that requires $O(2^{2^d} n)$ time, where n is the number of constraints and d is the number of variables. (Unless otherwise indicated, we assume unit-cost arithmetic operations.) This time bound is optimal with respect to n, and acceptable when d is very small. Variant algorithms have been found with the slightly better time bound $O(3^{d^2} n)$ [Cla86, Dye86]. Unfortunately, Megiddo's approach must take $\Omega(d!n)$ time, since it recursively solves linear programming problems with fewer variables [Dye86]. Dyer and Frieze [DF87] used random sampling [Cla87, Cla88] to obtain a variant of Megiddo's algorithm, with an expected time bound no better than $O(d^{3d} n)$. This paper gives an algorithm requiring expected time $O(d^2 n) + (\lg(n/d^2))^{\lg d+2} O(d)^{d/2+O(1)}$, as $n \to \infty$, where the constant factors do not depend on d. The leading term in the dependence on n is $O(d^2 n)$, a considerable improvement in d. The second term arises from the solution by the algorithm of $O(\lg n/d^2)^{\lg d+2}$ "small" linear programs with $O(d^2)$ constraints and d variables. The solution of these linear programming problems with the simplex algorithm requires $O(d)^{d/2+O(1)}$ time.

The key idea of the algorithm is random sampling, applied as in [Cla87, Cla88], to quickly throw out redundant constraints.

The next section presents the algorithm and proves that it terminates and is correct. In Section 3, a time bound is given and proven. The last section contains some concluding remarks.

2 The LP algorithm

2.1 The problem

Suppose a system of linear inequality constraints $Ax \le b$ is given, where A is a given $n \times d$ matrix, b is a given n-vector, and x is a d-vector $(x_1, \ldots, x_d)$. We assume for the moment that $b \ge 0$; this implies that 0 satisfies all the constraints, and so is feasible. Each inequality in this system defines a closed halfspace H of points that satisfy that inequality. The collection of these n halfspaces is a set S. The intersection $\cap_{H \in S} H$ is a polyhedral set $\mathcal{P}(S)$.

We consider the LP (Linear Programming) problem of determining the maximum x_1 coordinate of points x satisfying all the constraints, or

$$x_1^* = \max\{x_1 \mid Ax \le b\},$$

We will use the minimum norm solution for this LP, that is, the point $x^*(S)$ with Euclidean norm $\|x^*(S)\|_2$ equal to

$$\min\{\|x\|_2 \mid Ax \le b, x_1 = x_1^*\}.$$

It is readily shown that such a point is unique. (The requirement of a minimum norm is added to obtain this uniqueness. For background on linear programming, see e.g. [Sch86].)

It may be that the given LP problem is unbounded, so that points with arbitrarily large x_1 coordinates are in $\mathcal{P}(S)$. It will be useful to define an answer even for such LP problems. Rather than returning a point, the algorithm will return the ray with an endpoint of 0, in the direction of $x^*(\hat{S})$, where $\hat{S}$ is the set of constraints $Ax \leq 0$, together with the constraint $x_1 = 1$.

Note that a ray $x^*(S) \subset \mathcal{P}(S)$, recalling the assumption $b \geq 0$. Because of this, we will say that $x^*(S)$ satisfies all the constraints of S. In general, a ray z will be said to satisfy a constraint halfspace H just when $z \subset H$; otherwise z violates H.

2.2 The algorithm

The optimum point (or ray) is unique; furthermore, the optimum is determined by some d or fewer constraints of S. That is, there is a set $S^* \subset S$ of size d or less such that $x^*(S^*) = x^*(S)$, so the optimum for S^* alone is the same as for S. The constraints in $S \setminus S^*$ are redundant, in the sense that their deletion from S does not affect the optimum.

The main idea of the new algorithm is the same as for Megiddo's algorithm: throw away redundant constraints quickly. The further development of this idea is very different, however. The new algorithm builds a set $V^* \subset S$ over several phases. In each phase, a set $V \subset S \setminus S^*$ is added to V^*. The set V has two important properties: its size is no more than $2\sqrt{n}$, and it contains a constraint in S^*. After $d+1$ phases, V^* contains S^*, and also V^* has $O(d\sqrt{n})$ elements. That is, the algorithm quickly throws away the large set of redundant constraints $S \setminus V^*$. The algorithm proceeds recursively with V^*, and the recursion terminates for "small" sets of constraints. For these constraints, the appropriate optima are found using the simplex algorithm.

The algorithm is given in pseudo-code in Figure 1. The optimum $x^*(S)$ is computed as follows. Let $C_d = 9d^2$. If $n \leq C_d$, then compute the optimum $x^*(S)$ using the simplex algorithm. (More precisely, compute the maximum x_1 coordinate using simplex, then use a simplex-like "active set" method [GMW81] to find the minimum norm solution.) If $n > C_d$, then repeat the following, with V^* initially empty: let $R \subset S \setminus V^*$ be a random subset of size $r = d\sqrt{n}$, with all subsets of that size equally likely. Let $x^* \leftarrow x^*(R \cup V^*)$, determined recursively, and let V be the set of constraints violated by x^*. If $|V| > 2\sqrt{n}$, then try again with a new subset R, repeating until $|V| \leq 2\sqrt{n}$. Include the final set V in V^*; if V is empty, then exit the algorithm, returning x^* as $x^*(S)$. Otherwise, stay in the loop.

```
function $x^*(S : set\_of\_halfspaces)$
return $x^* : LP\_optimum$;

$V^* \leftarrow \phi$; $C_d \leftarrow 9d^2$;
if $n \leq C_d$ then return $simplex(S)$;
repeat
  repeat
    choose $R \subset S$ at random, $|R| = r = d\sqrt{n}$;
    $x^* \leftarrow x^*(R \cup V^*)$;
    $V \leftarrow \{H \in S \mid x^*$ violates $H\}$
  until $|V| \leq 2\sqrt{n}$;
  $V^* \leftarrow V^* \cup V$;
until $V = \phi$;
return $x^*$ ;
end function $x^*$;
```

Figure 1: The randomized function x^* for LP.

Termination of the inner loop with probability 1 follows from a bound of $\sqrt{n}$ on the expected size of V. This bound is proven in §3. By Markov's inequality, with probability at least 1/2 the set V will contain no more than $2\sqrt{n}$ constraints. This implies that the inner loop will stop after 2 iterations on average.

After the inner loop terminates, the set V has a key property: if nonempty, it contains at least one constraint of S^* that is not in V^*. Suppose x^* is a point, and that on the contrary, $V \neq \phi$ contains no constraints of S^*. Let point $x \succeq y$ if $(x_1, -\|x\|_2)$ is lexicographically greater than or equal to $(y_1, -\|y\|_2)$. We know that x^* satisfies all constraints in S^*, and so $x^*(S^*) \succeq x^*$. Since $R \cup V^* \subset S$, we know that $x^* \succeq x^*(S) = x^*(S^*)$, and so x^* has the same x_1 coordinate and norm as $x^*(S^*)$. There is only one such point in $\mathcal{P}(S^*)$, so $x^* = x^*(S^*) = x^*(S)$, and V must be empty. A similar argument holds if x^* is a ray.

Termination of the algorithm follows from this property of V. We know that after i executions of the outer loop, the set V^* contains at least i constraints of S^*. This means that by the d iterations or possibly before, we have $S^* \subset V^*$. With this true, the computed x^* will be $x^*(S)$ at the next iteration, and V will be empty, so that the algorithm stops.

3 Time complexity analysis

The main result needed for the time bound is the following.

Lemma 3.1 *With $b > 0$ in a given LP problem, the expected size of the set V is no more than $\sqrt{n}$ at any given time.*

This lemma is a corollary of results in [Cla88, §4]. For clarity and completeness, the proof of the results in [Cla88] will be specialized for this particular case. The intuitive idea is that since $x^*(R \cup V^*)$ violates no constraints of $R \cup V^*$, it violates few constraints of S.

Proof. We will assume for the moment that the given problem is nondegenerate, in that no $d+1-k$ hyperplanes meet at a k-flat; in particular, no $d+1$ hyperplanes contain a common point.

We begin by putting $x^*(R \cup V^*)$ in a larger class of "candidate" optima. That is, $x^*(R \cup V^*)$ is a member of a set

$$\mathcal{F}_S = \{x^*(T \cup V^*) \mid T \subset S \setminus V^*, |T| \leq d\}.$$

We can similarly define

$$\mathcal{F}_R = \{x^*(T \cup V^*) \mid T \subset R, |T| \leq d\}.$$

Plainly $\mathcal{F}_R \subset \mathcal{F}_S$, and $x^*(R \cup V^*)$ is the unique member of $\mathcal{F}_R$ that satisfies all the constraints in R.

For a given $x \in \mathcal{F}_S$, let $|x|$ denote the number of constraints of S that it violates, and let I_x be the indicator function for x, so that $I_x = 1$ when $x = x^* = x^*(R \cup V^*)$, and $I_x = 0$ otherwise. That is, I_x is a random variable whose value depends on the random choice of R. With these definitions, the expected size of V is

$$E[|V|] = E\left[\sum_{x \in \mathcal{F}_S} |x| I_x\right] = \sum_{x \in \mathcal{F}_S} |x| E[I_x] = \sum_{x \in \mathcal{F}_S} |x| P_x,$$

where P_x is the probability that $x = x^*$. What is the value of P_x? Since the subsets of size r are equally likely, $P_x\binom{n'}{r}$ is the number of subsets of size r that have $x = x^*$, with $n' = n - |V^*|$. We need to count the number of such subsets. For x to be x^*, two conditions must hold: x must be in $\mathcal{F}_R$, and x must satisfy all the constraints of R. For x to be in $\mathcal{F}_R$, some set T of constraints in $S \setminus V^*$ must be included in R, with $x = x^*(T \cup V^*)$. The nondegeneracy assumption implies that the smallest such set is unique. Let i_x denote the size of that set. Then for x to be x^*, i_x given constraints must be in R, and the remaining $r - i_x$ constraints must be from among the $n' - |x| - i_x$ constraints in $S \setminus V^*$ that neither define x nor are violated by x. We have

$$P_x = \binom{n' - |x| - i_x}{r - i_x} \Big/ \binom{n'}{r}.$$

Now since

$$\begin{aligned}\binom{n' - |x| - i_x}{r - i_x} &= \frac{n' - |x| - r + 1}{r - i_x}\binom{n' - |x| - i_x}{r - i_x - 1} \\ &\leq \frac{n' - r + 1}{r - d}\binom{n' - |x| - i_x}{r - i_x - 1},\end{aligned}$$

we have

$$E[|V|] \leq \frac{n' - r + 1}{r - d} \sum_{x \in \mathcal{F}_S} |x| \binom{n' - |x| - i_x}{r - i_x - 1} \Big/ \binom{n'}{r}.$$

The bound for $E[|V|]$ follows by showing that the sum is no more than d. By a counting argument as above, the summand is the probability that x is a member of $\mathcal{F}_R$ that violates exactly one constraint of R. By an indicator function argument as above, the sum is the expected number of $x \in \mathcal{F}_R$ that violate exactly one constraint of R. However, for any set R, the number of such x is no more than d: such an x is $x^*(R \cup V^* \setminus \{H\})$, for some constraint H, and $x^*(R \cup V^* \setminus \{H\}) = x^*(R \cup V^*)$ unless H is one of the d or fewer constraints determining $x^*(R \cup V^*)$.

Under the nondegeneracy assumption, we have $E[|V|] \leq d(n' - r + 1)/(r - d)$, which is no more than $\sqrt{n}$ for $r(r-1) \geq dn'$.

It remains to check that the complexity analysis follows through when the input set is degenerate. Given a degenerate LP problem, we will show that there is a "perturbed" version of the problem such that the expected size of the set V in the perturbed problem is no smaller than in the original. The perturbation idea goes back to [Cha52], and is equivalent to the lexicographic tie-breaking used to prevent cycling in the simplex algorithm. The idea is that the vector b is replaced by the vector $b + (\epsilon, \epsilon^2, \ldots, \epsilon^n)$, where $\epsilon > 0$ is very small. (A similar perturbation is added to the constraints for $\hat{S}$ in the unbounded case.) The resulting system is nondegenerate, in the sense discussed at the beginning of the proof. Moreover, each $x \in \mathcal{F}_S$ in the original problem is associated with a subset of $\mathcal{F}_{S'}$, where S' is the corresponding perturbed constraint set. For given $x \in \mathcal{F}_S$ and $T \subset S$ with $x = x^*(T \cup V^*)$, the optimum $x^*(T' \cup V^*)$ is in the subset associated with x, where T' is the perturbed version of T. Also, whenever $x = x^*(R \cup V^*)$, some x' associated with x is the optimum for the corresponding perturbed problem, for sufficiently small ϵ. The optimum x' violates at least as many constraints as x does. Thus the expected size of the set

V in the perturbed problem is no less than that in the original problem, and the bound for $E[|V|]$ holds in general. □

Theorem 3.2 *The LP algorithm requires*

$$O(d^2 n) + (\lg(n/d^2))^{\lg d+2} O(d)^{d/2+O(1)}$$

expected time, as $n \to \infty$, where the constant factors do not depend on d.

Proof. We will continue to assume that $b > 0$ until the end of the proof.

The set V^* grows by at most $2\sqrt{n}$ in each phase, with $d+1$ phases. The maximum size of $R \cup V^*$ is therefore $\sqrt{C_d n}$, where C_d is $9d^2$. Let $T(n)$ be the expected time required for a problem with n constraints (and d variables). The time required to find acceptably small sets V is bounded by $\sum_{i\geq 0} T(\sqrt{C_d n})/2^i$, or $2T(\sqrt{C_d n})$, for a total of $2(d+1)T(\sqrt{C_d n})$ over all phases. The time required to test that a given constraint is satisfied by a computed optimum is $O(d)$, for a total of $O(d^2 n)$ over all phases. For $n > C_d$, the expected time $T(n)$ is bounded by

$$T(n) \leq 2(d+1)T(\sqrt{C_d n}) + O(d^2 n),$$

where the constant does not depend on d. In the base case, where $n \leq C_d$, the simplex algorithm takes $d^{O(1)}$ time to visit a vertex of $\mathcal{P}(S)$, and visits each vertex at most once. This gives a time bound $\binom{2C_d}{\lfloor d/2 \rfloor} d^{O(1)}$ for simplex. This is $O(d)^{d/2+O(1)}$ using Stirling's approximation. The simplex algorithm, or a similar active-set algorithm with the same time bound, is called $O(1)$ times in the base case. Thus

$$\begin{aligned} T(n) &\leq 2^M (d+1)^M O(d)^{\lfloor d/2 \rfloor + O(1)} \\ &+ O(d^2) \sum_{0 \leq i < M} (d+1)^i n_i, \end{aligned}$$

where $M = O(\lg \lg n/C_d)$ is the recursion depth, and $n_i = C_d(n/C_d)^{1/2^i}$. The ith term in the sum is dominated by $n/2^i$ for $n_i > 16(d+1)^2 C_d$, and the remaining terms in the sum are dominated by the time required for solving the base cases. The bound

$$O(d^2 n) + (\lg(n/d^2))^{\lg d+1} O(d)^{d/2+O(1)}$$

follows, assuming $b > 0$.

The slightly modified bound of the theorem holds even when we do not have $b \geq 0$. Suppose we have a feasible point $x_0 \in \mathcal{P}(S)$. Then we can solve the LP problem with constraints $A(x - x_0) \leq b - Ax_0$ for $x - x_0$. This problem has $b - Ax_0 > 0$. (The result is not generally a minimum norm solution to the original problem.) To obtain a feasible solution, we solve the LP

$$\max\{t \mid Ax + t1 \leq b, t \leq 0\},$$

where 1 is the n-vector of 1's. This problem is feasible, with a feasible point readily obtainable. If the answer is negative, the original problem is infeasible. Otherwise, we have a feasible point for the original problem. The cost of this generality is one more variable and one more constraint, giving the time bound of the theorem. □

4 Concluding remarks

These ideas should be applicable to other convex programming problems. For example, the problem of finding the smallest sphere enclosing a set of points in E^d is amenable to this approach, and resulting algorithm has the same time bound as the LP algorithm given. The weighted Euclidean one-center problem and various L_1 approximation problems should also be amenable.

These results may be useful in obtaining insight into the observed time complexity of various algorithms and heuristics for convex programming problems.

The new algorithm is a (small) step toward a strongly polynomial (randomized) algorithm for linear programming. Also, for $n \gg d$, the best previous result is Khachian's algorithm[Kha80], requiring $O(nd^3 L)$ operations on numbers with $O(L)$ bits, where L is a measure of problem size. By application of Khachian's algorithm for "small" linear programs and stopping the recursion early, an algorithm is obtained that requires $O(nd^2 + n^{1/4} d^8 L)$ expected operations. This is an improvement when $d = o(n^{1/7})$. (Khachian's algorithm can be adapted to the problem of finding the shortest vector in a convex set, and so minimum norm solutions can be found with it.)

Acknowledgements. I thank David Gay and Margaret Wright for suggesting the use of the minimum norm solution for tie-breaking. I thank Nimrod Megiddo and Alan Frieze for helpful comments.

References

[Cha52] A. Charnes. Optimality and degeneracy in linear programming. *Econometrika*, 20:160–170, 1952.

[Cla86] K. L. Clarkson. Linear programming in $O(n3^{d^2})$ time. *Information Processing Letters*, 22:21-24, 1986.

[Cla87] K. L. Clarkson. New applications of random sampling in computational geometry. *Discrete & Comp. Geom.*, 2:195–222, 1987.

[Cla88] K. L. Clarkson. Random sampling in computational geometry, II. *Proc. Fourth Symp. on Comp. Geometry*, 1988.

[DF87] M. E. Dyer and A. M. Frieze. A randomized algorithm for fixed-dimensional linear programming. unpublished manuscript, 1987.

[Dye86] M. E. Dyer. On a multidimensional search technique and its application to the euclidean one-centre problem. *SIAM Journal on Computing*, 15:725–738, 1986.

[GMW81] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization.* Academic Press, New York, 1981.

[Kha80] L. G. Khachian. Polynomial algorithms in linear programming. *Zhurnal Vychislitelnoi Mathematiki i Matematicheskoi Fiziki*, pages 53–72, 1980.

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

[Sch86] A. Schrijver. *Theory of Linear and Integer Programming.* Wiley, New York, 1986.