REPRINTED FROM:

operations research letters

Volume 1, No. 2, April 1982

COMPLEXITY OF LINEAR PROGRAMMING *

J.F. TRAUB

Department of Computer Science, Columbia University, New York, NY 10027, U.S.A.

H. WOŹNIAKOWSKI

Institute of Informatics, University of Warsaw, Warsaw, Poland, and Department of Computer Science, Columbia University, New York, NY 10027, U.S.A.

pp. 59–62

NORTH-HOLLAND

COMPLEXITY OF LINEAR PROGRAMMING *

J.F. TRAUB

Department of Computer Science, Columbia University, New York, NY 10027, U.S.A.

H. WOZNIAKOWSKI

Institute of Informatics, University of Warsaw, Warsaw, Poland, and Department of Computer Science, Columbia University, New York, NY 10027, U.S.A.

Received May 1981
Revised December 1981

The complexity of linear programming is discussed in the "integer" and "real number" models of computation. Even though the integer model is widely used in theoretical computer science, the real number model is more useful for estimating an algorithm's running time in actual computation.

Although the ellipsoid algorithm is a polynomial-time algorithm in the integer model, we prove that it has unbounded complexity in the real number model. We conjecture that there exists no polynomial-time algorithm for the linear inequalities problem in the real number model. We also conjecture that linear inequalities are strictly harder than linear equalities in all "reasonable" models of computation.

Computational complexity, models of computation, ellipsoid algorithm, linear programming, linear inequalities, polynomial-time algorithm

1. Introduction

Why does the ellipsoid algorithm show that linear programming problems are "easy" and yet not provide us with a useful algorithm for the linear programming problems which arise in practice? In this article we present a possible answer.

We assume the reader is familiar with the linear programming problem and with the ellipsoid algorithm as defined, for instance, in Aspvall and Stone [2]. An amusing account of the media coverage of "Khachian's algorithm" is given by Lawler [9]. The history of the ellipsoid algorithm was first presented by Traub and Wozniakowski [11]. A comparison of the ellipsoid and simplex algorithms may be found in Lovász [10].

Although the ellipsoid algorithm does not seem of practical importance for linear programming, it can be useful for special instances. It is a powerful new tool for other combinatorial optimization problems; see, for example, Grötschel, Lovász, and Schrijver [7].

Our thesis is that the widespread confusion as to the significance of the ellipsoid algorithm stems from disregarding the implications of the underlying *models of computation*. The linear programming problem was shown to be "easy" in the "integer" model often used in theoretical computer science. This was done by showing that the ellipsoid algorithm is good in this model. If we wish to estimate an algorithm's running time on most computers, the "real number" model should be used. In this second model the ellipsoid algorithm is not a good algorithm.

Numerous improvements on the ellipsoid algorithm have been proposed. We believe that the thesis advanced in this paper would not be affected by these variants.

Before stating our viewpoint more precisely, we must define what we mean by models of computation, problem complexity, and easy problem.

2. Models of computation

The difficulty of a problem can only be discussed in the context of a model of computation. For the purpose

* Portions of this paper were first presented at the Polynomial-Time Workshop, New York, February 1980.
This research was supported in part by the National Science Foundation under Grant MCS-7823676.

of this paper, a model of computation consists of the specification of

a number system,
arithmetic,
costs associated with arithmetic.

In this paper arithmetic operation shall mean: $+$, $-$, $*$, $/$.

Examples of important models of computation are:

Real number model (infinite precision)
number system: reals,
arithmetic: exact,
cost: unit cost for each operation;

Integer model (variable precision fixed point)
number system: integers,
arithmetic: exact or approximate,
cost: proportional to length of numbers.

We comment on these models. Infinite precision does not exist in a finite universe. It is however a very useful mathematical abstraction. It is the model widely used in algebraic complexity, i.e., in the famous matrix multiplication problem.

Fixed precision floating point is almost universally used for numerical calculations whether they occur in science, engineering, or economics. Complexity results are essentially the same as in the infinite precision model. However only approximate results can be computed and algorithm stability is an important issue. We use the real number model rather than the fixed precision floating point model to avoid being distracted by round-off issues.

Variable precision fixed point is often used in theoretical studies. It does not model most numerical calculation. In particular, it is used in Khachian's paper [8] (Khachian uses a Turing machine model but that need not concern us here).

The essence of the difference between integer and real number models is that in the former the cost of an arithmetic operation depends on number size, while in the latter it is independent.

3. Problem complexity and algorithm complexity

Let the model of computation be fixed. Then the minimal cost of solving a problem is called its *computational complexity* (or problem complexity). We often write complexity for brevity.

Algorithm complexity is the cost of a particular algorithm. This should be contrasted with problem complexity which is the minimal cost over all possible algorithms.

The complexity of most problems is unknown and we have to content ourselves with *upper* and *lower bounds*. Typically, an upper bound is the cost of the best algorithm known for solving the problem. A lower bound must be established through a theorem that states there does not exist an algorithm whose cost is less than the lower bound. Not surprisingly, lower bounds are far harder to establish than upper bounds. See however Traub and Wozniakowski [12] where sharp lower bounds are obtained for important classes of problems.

The models of computation defined in Section 1 would have to be more precisely specified in order to rigorously establish lower bound theorems.

We now define easy problem in general. Let n be a measure of problem size in the real number model. If the problem complexity is a polynomial in n, we say the problem has polynomial complexity.

In the integer model, let n be a measure of problem size and let L be a measure of the number of digits to represent the input. If the problem complexity is a polynomial in n and L, we say the problem has polynomial complexity.

In either model, we say a problem is easy if it has polynomial complexity. See Aho, Hopcroft and Ullman [1] for a motivation of this definition. Easy and hard provide a crude dichotomy. If one is interested in actual running times, the coefficients and degree of the polynomial are important.

We give a simple example. Consider the linear system $Ax = b$ where A is an n by n nonsingular matrix. This problem has $n^2 + n$ input data and n output data.

Consider the real number model. The Gauss elimination algorithm costs $O(n^3)$ arithmetic operations. Hence the complexity of linear systems (that is, the problem complexity) in this model is at most $O(n^3)$. (Indeed, by other arguments we know the complexity is less than $O(n^{2.5})$.)

Consider next the integer model. Let the total number of digits used to represent the input data be L. Then there is an algorithm (Edmonds [4]) so that the numerators and denominators of the solutions can be computed with polynomial cost in n and L. This proves that also in this model the problem complexity is polynomial.

4. The ellipsoid algorithm

Since linear programming can be reduced to the solution of linear inequalities, we will confine ourselves to a study of the latter. We assume there are m equations in n variables. We consider here only strict in-

equalities and shall assume a solution exists.

We describe the ellipsoid algorithm in qualitative terms. A sequence of ellipsoids of decreasing volume is constructed such that each ellipsoid contains a solution. The initial ellipsoid contains a solution and the process is terminated when an ellipsoid is reached whose center is a solution.

How many steps are required to solve the inequalities? Let the ratio of the volumes of successive ellipsoids by c. Then c depends only on n, and the number of steps, k, of the ellipsoid algorithm is

$$k \leq \left\lceil \frac{\log \dfrac{\lambda(E_0)}{\lambda(S)}}{\log \dfrac{1}{c}} \right\rceil,$$

where $\lambda(E_0)$ and $\lambda(S)$ are the volumes of the initial ellipsoid and of the portion of the solution set in E_0. It is known that this bound is essentially sharp.

5. Complexity of linear systems and linear inequalities

We have now assembled the background which enables us to discuss complexity of linear inequalities in two important models of computation. We also include the complexity of linear systems since it is instructive to contrast it with the complexity of linear inequalities.

Consider first the *integer* model.

Linear systems. As noted above the complexity is a polynomial in n and L.

Linear inequalities. The ellipsoid algorithm proves that in this model of computation the complexity of linear inequalities is a polynomial in n, m, and L. This is why the algorithm has aroused much theoretical interest.

It is important to realize that the values of n, m, L arising in practice are very large. Dantzig [3] has estimated that for typical problems from energy/economic planning, the ellipsoid algorithm would take some fifty million years, while the simplex algorithm performed in fixed precision floating point arithmetic takes half an hour.

Consider next the *real number* model.

Linear systems. As noted above, the complexity is a polynomial in n.

Linear inequalities. The ellipsoid algorithm is not a polynomial algorithm. Indeed we shall show that the complexity of this algorithm is unbounded. We provide a proof below. The complexity of the simplex algorithm is finite. Although in the worst case the complexity of the simplex algorithm is at least exponential, for problems arising in practice the simplex algorithm is surprisingly fast (Dantzig [3]). No polynomial algorithm is known. The complexity of linear inequalities is open in this model.

We prove that in the real number model the ellipsoid algorithm has unbounded complexity. This is established by giving a 2 by 2 example for which the number of steps of the ellipsoid algorithm is arbitrarily large. Hence the ellipsoid algorithm certainly does not have complexity which is polynomial in n. (In contrast, the simplex algorithm would take only a couple of steps for this problem.)

We use the version of the ellipsoid algorithm for the inequalities $Ax \leq b$ as defined, for example, in Aspvall and Stone [2]. The algorithm generates the sequences $\{x^{(k)}\}$, $\{B^{(k)}\}$ by

$$x^{(k+1)} = x^{(k)} - \frac{1}{n+1} B^{(k)} a_j / \sqrt{a_j^{\mathrm{T}} B^{(k)} a_j},$$

$$B^{(k+1)} = \frac{n^2}{n^2-1} \left(B^{(k)} - \frac{2}{n+1} \frac{(B^{(k)} a_j)(B^{(k)} a_j)^{\mathrm{T}}}{a_j^{\mathrm{T}} B^{(k)} a_j} \right),$$

where a_i^{T} is the ith row of A and $a_j^{\mathrm{T}} x^{(k)} \geq b_j$.

Define a 2 by 2 system of inequalities by

$$A = \begin{pmatrix} -1 & 0 \\ -1 & 0 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 - \frac{1}{s} \\ -1 \end{pmatrix}, \qquad s \geq \frac{5}{\sqrt{3}}$$

Let

$$x^{(0)} = \begin{pmatrix} 1 - \sqrt{2}/s \\ 0 \end{pmatrix}, \qquad B^{(0)} = 2I.$$

(These starting values lead to clean formulas. They differ from the starting values of Aspvall and Stone [2] but our conclusions still apply if their starting values are used.) For this example

$$x^{(k)} = \begin{pmatrix} c_k \\ 0 \end{pmatrix}, \qquad c_{k+1} = c_k - s_k \frac{\sqrt{3}}{3} \left(\frac{2}{3} \right)^k$$

with $c_0 = 1 - \sqrt{2}/s$ and

$$s_k = \begin{cases} -1, & c_k \leq 1, \\ -1, & c_k \geq 1 - \dfrac{1}{s} \end{cases}$$

Furthermore

$$B^{(k)} = \begin{pmatrix} f_k & 0 \\ 0 & g_k \end{pmatrix}, \qquad f_k = 2\left(\frac{4}{3}\right)^k, \qquad g_k = 2\left(\frac{2}{3}\right)^k$$

Hence

$$c_k = 1 - (-1)^k \frac{\sqrt{2}}{5}\left(\frac{2}{3}\right)^k$$

as long as $c_k \notin (1, 1 - 1/s)$. Let k be the smallest integer such that $c_k \in (1, 1 - 1/s)$. Then

$$k = 2\left\lfloor \frac{\log \frac{5\sqrt{2}}{2s}}{2\log \frac{2}{3}} - 1 \right\rfloor \approx \frac{\log s}{\log \frac{3}{2}}$$

We have therefore established the following:

Theorem. *The ellipsoid algorithm is not a polynomial algorithm in the real number model.*

Now assume that s is an integer. Multiplying by s gives us inequalities with integer coefficients. Observe that $L \approx \log s$. Hence our example shows that the number of steps of the ellipsoid algorithm (and not just the bound) increases as L.

Although the dependence on L is expected and acceptable in the integer model, it is not expected in the real number model. The fact that the number of steps in the ellipsoid algorithm increases as L (which of course is true in either model) causes the difficulty when the algorithm is used in numerical computation.

It is instructive to compare the linear transportation problem with linear programming. For the former problem Edmonds and Karp [5] give an algorithm whose running time also depends on the size of the input numbers. Yet their algorithm is of practical value. The difference is that for the ellipsoid algorithm typical running times are given by the upper bound whereas this is not the case for the Edmonds-Karp algorithm.

6. Conjectures

We propose

Conjecture 1. *The linear inequalities problem does not have polynomial complexity in the real number model.*

We believe that a central problem in mathematics and computer science is to ascertain a hierarchy of problem difficulty. We say a problem is harder than a second problem if it has higher complexity. If problem A has polynomial complexity while problem B does not, then B is harder than A. If both enjoy polynomial complexity we can still compare their complexities by comparing degrees.

Let the number of variables in the liner system by the maximum of n, m in the inequalities. Gale [6] asked whether inequalities have higher complexity. We propose

Conjecture 2. *The solution of linear inequalities is strictly harder than linear systems in all "reasonable" models of computation.*

Acknowledgement

We are indebted to Richard M. Karp for his perceptive comments.

References

[1] A.V. Aho, J.E. Hopcraft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA (1974).

[2] B. Aspvall and R.E. Stone, "Khachian's linear programming algorithm", *J. Algorithms* 1, 1–13 (1980).

[3] G.B. Dantzig, "Comments on Khachian's algorithm for linear programming", Department of Operations Research, Stanford University (1979).

[4] J. Edmonds, "Systems of distinct representatives and linear algebra", *J. Res. Nat. Bur. Standards* 71B, 241–245 (1967).

[5] J. Edmonds and R.M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems", *J. ACM* 19, 248–264 (1972).

[6] D. Gale, "How to solve linear inequalities", *Amer. Math. Monthly* 76, 589–599 (1969).

[7] M. Grötschel, L. Lovász and A. Schrijver, "The ellipsoid method and its consequences in combinatorial optimization", *Combinatorica* 1, 169–197 (1981).

[8] L.G. Khachian, "A polynomial algorithm in linear programming", *Dokl. Akad. Nauk USSR* 244, 1093–1096 (1979), translated as *Soviet Math. Dokl.* 20, 191–194 (1979).

[9] E. Lawler, "The great mathematical sputnik of 1979", *The Sciences* 20 (7), 12–15 (1980).

[10] L. Lovász, "A new linear programming algorithm – Better or worse than the simplex method?", *Math. Intelligencer* 2 (3), 141–148 (1980).

[11] J.F. Traub and H. Wozniakowski, "On a Soviet algorithm for the linear programming problem", Department of Computer Science, Columbia University (1979).

[12] J.F. Traub and H. Wozniakowski, *A General Theory of Optimal Algorithms*, Academic Press, New York (1980).