stichting
mathematisch
centrum

MC

AFDELING MATHEMATISCHE BESLISKUNDE BW 40/74 OCTOBER

JAC. M. ANTHONISSE & P. VAN EMDE BOAS

ARE POLYNOMIAL ALGORITHMS REALLY GOOD?

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O), by the Municipality of Amsterdam, by the University of Amsterdam, by the Free University at Amsterdam, and by industries.

AMS (MOS) subject classification scheme (1970): 68A20 68M20 90C10

Are polynomial algorithms really good?

by

Jac. M. Anthonisse and P. van Emde Boas

ABSTRACT

In the theory of computational complexity the upper bound on the runtime of an algorithm is usually expressed in the length of the input. Problems are believed to be untractable unless they are solved by an algorithm with polynomially bounded runtime. The observation that complete enumeration is an unfeasible but quadratic algorithm for all except a vanishing fraction of the bivalent LP problems suggests that the length of the input is, in general, not the best quantity in terms of which the complexity of a problem should be measured.

KEY WORDS & PHRASES: *polynomial algorithms, bivalent LP problems, input length.*

A starting point for the theory of computational complexity is the working hypothesis that a class of problems can be regarded as *tractable* if and only if there is an algorithm for their solution whose running time is bounded by a polynomial in the size of the input, c.f. KARP [2]. Input for an algorithm is assumed to consist of a finite string of 0's and 1's, the size of the input is defined as

(0) L = number of zeroes + number of ones.

An algorithm is *good* for a class of problems if the running time for each problem is $O(L^p)$, i.e. the running time is bounded by a polynomial in L of degree p. Assuming that the class contains at least one problem such that the algorithm must look at least once at each bit of the input-string it follows that $p \geq 1$.

A central problem in the theory of computational complexity concerns the existence of a good algorithm for the class of bivalent linear programming problems. The general form of a problem from this class is:

$$(1) \qquad \text{maximize} \quad \sum_{j=1}^{n} c_j x_j$$

$$(2) \qquad \text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad (i=1,\ldots,m)$$

$$(3) \qquad x_j \in \{0,1\} \quad (j=1,\ldots,n)$$

where c_j, a_{ij} and b_i denote fixed integer coefficients.

The existence of a good algorithm for these problems implies the existence of good algorithms for many combinatorial and graph-theoretical problems.

The input-string for an algorithm solving this class of problems should contain an encoding of the values of n, m, and all c_j, a_{ij}, b_i.

Now assume that algorithm A is a good algorithm for the bivalent linear programming problem, and that its running time R_A is $O(L^p)$, thus

(4) $$R_A \leq t_A L^p$$

where t_A denotes a technical coefficient.

The trivial algorithm for solving the bivalent LP problems is to generate the 2^n vectors $(x_1,\ldots,x_n)$ satisfying (3) until the optimal solution is found. Assuming that the time needed for testing whether a vector satisfies the constraints (2) and for the computation of the criterion function (1), is of order L the runtime of this algorithm is

(5) $$R_C \leq t_C 2^n L,$$

where t_C denotes another technical coefficient.
As

(6) $$L > \sqrt[p-1]{\frac{t_C}{t_A} 2^n}$$

implies

(7) $$t_A L^p > t_C 2^n L$$

the conclusion is, that for sufficiently large problems, complete enumeration should be preferred above a good algorithm, if the choice is based upon the upper bounds only.

From a practical point of view it is disappointing that complete enumeration, rather than a good algorithm, should be recommended for solving large problems, because the running time becomes prohibitive if, say, $n > 15$.

Within the theory of complexity however, a polynomial algorithm (4) which can solve only small problems, i.e. problems not satisfying (6); together with complete enumeration for larger problems, constitutes a good algorithm for the complete class of problems.

The discovery of such an algorithm, even with p rather large, would be of great theoretical and practical interest. But that algorithm is not

necessarily always better than complete enumeration.

It should be noted that, due to (6), the size of problems to be solved by an algorithm with $R = 0\ (L^p)$ is a decreasing function of p.

Moreover, as we shall see, the fraction of small problems is a decreasing function of n, so complete enumeration becomes good for the majority of all problems.

Before a polynomial algorithm A can solve a problem an input-string defining that problem must be produced. Let C denote the class of problems solved by A, assume that C is partitioned into subclasses C_s (s=1,2,3,...) and let L_s denote a measure of the length of input-strings for the problems in C_s. L_s could be an upper bound for the length of the strings or the length of the average or median input-string.

If $L_s = 0\ (s^q)$ then the running time of A is bounded by a polynomial of degree pq in s, and A is an acceptable algorithm for C.
If $L_s \geq 0\ (2^s)$ then the upper bound on the running time of A is $\geq 0\ (2^{ps})$. Moreover, in this case, the production of the input-string becomes prohibitive. This does not imply that problem C is untractable, because the encoding of a problem into an input-string is determined by the structure of A.

Now consider the class of bivalent LP problems (1), (2), (3) again. A vector $(x_1,\ldots,x_n)$ satisfying (3) either is a feasible solution or violates at least one of the constraints (2). Hence, a set of feasible solutions corresponds to every problem. Conversely, each subset from the set of vectors satisfying (3) can be made the set of feasible solutions of a bivalent LP problem. Thus the number of bivalent LP problems in n variables is

$$N \geq 2^{2^n}. \tag{8}$$

The actual number is still larger since any feasible solution can be made the optimal one. Furthermore, many input-strings correspond to problems with identical feasible set and identical optimal solution.

Different problems should correspond to different input-strings, the minimum-length encoding for a class of problems consists of the assignment of the labels 0,1,2,...,N-1 to the N problems.
As

$$\sum_{i=1}^{L} 2^i \geq 2^{2^n} \tag{9}$$

implies

$$L \geq 2^n \tag{10}$$

at least one problem requires an input-string of length 2^n.
Moreover, each of at least half the number of problems requires an input-string of length 2^n-1.

The conclusion is:
in every encoding of bivalent LP *problems in* n *variables the median length of the input-string is* $\geq 0(2^n)$.

This conclusion can also be formulated in the terminology of Boolean functions, i.e. in every representation of Boolean functions each of at least half the number of functions requires at least 2^n-1 bits.
Thus, if an algorithm creates and manipulates Boolean functions then an exponential growth in storage requirements and running time should be expected, c.f. Anthonisse [1].

For each problem with $L \geq 2^n - k$ the runtime of complete enumeration satisfies

$$R_C \leq t_C 2^n L \leq t_C 2^n L \leq \frac{L}{2^n - k} \leq t_C L^2 \frac{2^n}{2^n - k} \tag{11}$$

The number of problems with $L < 2^n-k$ is bounded by

$$2^{2^n-k} \leq \frac{N}{2^k} \tag{12}$$

Taking $k = 2^{n-1}$ we conclude that with the exception of a fraction

(13) $$\left(2^{2^{n-1}}\right)^{-1}$$

of the problems in n variables, the running time of complete enumeration is $O(L^2)$.

This counter-intuitive conclusion suggests that the length of the input-string might not be the proper quantity in terms of which the complexity of a problem should be measured. Similar observations have been made by SAVAGE and LAMAGNA (private communication). Instead of the length of the string one could consider the contents of the string. For the example of bivalent LP problems expressions in terms of n, m and the size of the coefficients might be considered. Within such a framework the trade-off involved in replacing some or all constraints by a single equivalent one can be studied, c.f. PADBERG [3].

One might argue that the large problems used in the above estimates are unrealistic and can not be formulated due to the length of their input-strings. This would be another argument in favor of fixing a bound on L, expressed in n, m and other parameters, and to study the existence of good algorithms for these problems only.

REFERENCES

[1] Anthonisse, J.M., *A note on prohibitive algorithms*, Mathematical Center, Amsterdam, report BW 10/71, 1971

[2] Karp, R.M., *Reducibility Among Combinatorial Problems* in: Miller, R.E. and Thatcher, J.W. (eds), Complexity of Computer Computations, Plenum Press, New York, 1972

[3] PADBERG, M.W., *Equivalent Knapsack-Type Formulations of Bounded Integer Linear Programs:* An Alternative Approach, NRLQ 19 (1972) 699-708