COMPUTING EXPERIENCE WITH LINEAR PROGRAMMING AND ITS VARIANTS

William Orchard-Hays

P-688

August 8, 1955

The RAND Corporation

1700 MAIN ST. • SANTA MONICA • CALIFORNIA

SUMMARY

The application of linear programming to complex problems gives rise to the handling of large matrix problems. The simplex method ~~has~~ proven to be the most expedient of any tried but, since this is essentially Gaussian elimination, numerical techniques for handling it have definite limitations, both with regard to running time and memory size.

RAND's simplex codes for the IBM 701 ~~have been~~ were designed, through revisions to the method and special compiling routines, to maintain great precision and facilitate checking, to be adaptable to many variations, and to be as automatic in operation as possible. Several quite large models of considerable variety are discussed along with the prospects for future improvements, both in theory and in coding techniques. ( )

COMPUTING EXPERIENCE WITH LINEAR PROGRAMMING AND ITS VARIANTS

A great deal has been said and written in the past two or three years about the technique known as linear programming which uses a mathematical model to represent a real situation. When one undertakes to study a complex real-life problem by such a model, he quickly finds that the paradise of optimality is guarded by three monsters called Conceptualization, Formulation, and Solution. Any apparent defeat of this trio turns out to be only temporary since they seem capable of escaping any conceivable kind of shackles. However, they are not completely invulnerable and linear programming is interesting and useful precisely because it is linear and hence amenable to relatively simple mathematical procedures. Since the formal definition refers to the model rather than to its antitype, it is meaningful only after one has had some exposure to the method. It will be assumed that the reader is familiar with the form of such a model and with the more common terms of the subject [1,2,3].*

The old cliche about not seeing the forest for the trees applies especially well to the question of conceptualization. Oftentimes, one has lived to close to a tough problem for so long that he has a mental block against a new approach which appears to make ruthless simplifications. Yet such a step is often necessary in applying a new technique. Clearly, it will sometimes be necessary to oversimplify, or at least to

---

* Bracketed numbers refer to references at end of paper.

take a different view of some factors, in order to fit a problem into a linear context. Furthermore, one must usually be very selective in chocsing the conditions that one is going to impose on a model. One cannot hope to take account of, say, every facet of the daily activities of an industry in a quantitative statement. Nevertheless, in spite of these conceptual difficulties, the range of problems for which useful results can be obtained is, I believe, surprisingly large.

Once one has conceived of a way to represent a realistic number of constraints without running their number up beyond computational capabilities, he is faced with the task of actually formulating the numerical model. This process, it should be noted, forces someone to acquire an intimate and detailed knowledge of the complex being studied. Our customers are often able to interpret the results of numerical computations very lucidly, often while the problem is running. This sort of insight is valuable in itself so long as one distinguishes between the model and the real world.

When the numerical model is set down, there is the problem of how to handle the computations. Furthermore, one optimal solution is quite likely to be insufficient to answer all the formulator's questions. (In fact, we have come to view with suspicion any problem which is to be run only once.) If this is known in advance, we can often suggest changes in the model, or at least in the way of looking at it, which will render these continuing computations easier to perform. For example, in some problems which were not too large, it

was discovered that it would be necessary to add additional restraints after each optimal solution was computed. By dualizing the problem, it was possible to hold the number of restraints fixed throughout and merely increase the number of activity vectors from time to time, proceeding from one optimal solution to the next without starting all over with a larger system.

It should be explained, incidentally, that all the problems whose solutions we have been concerned with have been solved with the simplex method or some variant thereof. The number of these variants is constantly growing and it appears that the trend will be to specialized procedures for problems classified by types, at least for large problems [4,5]. However, the present discussion pertains to general methods.

The first problem for which extensive runs were made at RAND was a petroleum blending study [6, 7]. The calculations were carried out on an IBM CPC Model II with a special set-up. The product form for the inverse of the basis matrix was first used in this set-up to speed up operation [8, 9]. This form has been carried over into all our later simplex codes for the IBM 701 and has proved to be an extremely versatile tool for working with matrices and their inverses on high-speed computers.

We also performed what has become known as parametric programming, or PLP, on this first problem. PLP can be divided into three major types, according as

(1) the elements of the right-hand side

(2) the coefficients of the objective function, or

(3) elements of the restraint matrix

are expressed as linear functions of a single parameter [3, Chap. 4]. All three of these were applied, although quite crudely at the time. Parametrizing elements of the matrix is not a very satisfactory device since one must simultaneously maintain optimality and feasibility and also avoid passing through a point of singularity. As a result, the calculations become so involved that the cost of performing them eats up the desired saving. However, alternate techniques for making changes within the matrix have been devised, mainly for correcting errors found after much computing time has been invested. It is now very seldom that a problem must be restarted from the beginning.

We have been able to develop the parametrization of the right-hand side into a fully automatic procedure, and it has been used extensively. Variations in the optimizing form are not as neat computing-wise. The reason for this is that one must repeatedly alter a set of n numbers, where n is the number of variables, instead of m numbers, where m is the number of constraints and typically much smaller than n. This is at odds with the basic philosophy of our code. However, here again alternate devices have been perfected which in practice accomplish virtually the same result. Our latest code, for example, will optimize successively and automatically on a multiplicity of objective functions. These forms

can be made to differ by non-uniform ratios so that in a sense it is more general than actual PLP. The optimal solution for the first form is used as a starting point for the second, the optimal solution for the second as a starting point for the third, and so on. On the other hand, this method does not permit one to solve for the critical values of a single parameter automatically. We can obtain these one at a time without too much effort but they are not usually required.

It was believed that an extremely fast code using fixed point arithmetic could be programmed for the 701 which would have been capable of handling systems with 100 restraint equations. Such a program was indeed coded but it proved to be impractical due to the fast accumulation of round-off error. A second version was patched up from the first but it gave no better results. We then decided to go to double-precision floating-point arithmetic for the third code and this did permit us to handle 100 order systems successfully. Since the simplex method is essentially selective Gaussian elimination, it is mandatory that one maintain extreme precision even though the input data may not be very accurate itself. An error analysis is small comfort when one must require the machine to make decisions on the basis of intermediate results. Thus we feel that the extra computing time is a necessary cost for a successful code.

However, as the code became more elaborate, it became almost impossibly interwoven so that a fourth code was devised

which emphasized clean logical structure [10]. The present code, which is the fifth, is only a slight refinement of the fourth, emphasizing simplicity of operation. Space has been provided for up to 250 restraint equations in almost any reasonable number of variables -- 1553 being the largest to date -- but solution times are somewhat high for systems with more than 150 equations. However, the various parts of the code are designed so that they can be coupled together to form new routines almost at will. This is accomplished by using a compiling routine which puts together subroutines immediately prior to their use while a problem is running. A master control region activates the compiler through a combination link and pseudo-instruction. Naturally, we have auxiliary routines for use with the system to facilitate re-starting, picking up after a machine failure, or altering the model, as well as to make certain side computations sometimes required. In addition, a composite algorithm built into the basic routine takes over automatically if any infeasibilities exist in the initial solution. This is often used in place of the dual simplex algorithm [11] since the latter is slower in operation and assumes optimality as a starting point. In short, we have attempted to provide for a variety of problems in two ways: first, by having a basic routine which is as general and elaborate as practicable; and second, by allowing a variety of programs to be constructable in our coding system with a minimum of effort, sort of a programmer's set of "pluggable units".

The initial petroleum model was followed by three others. The first was a 105 order system which represented in considerable detail the U. S. refining and crude oil producing industry [12]. This model, like the first, was formulated by Dr. Alan S. Manne. After obtaining an initial optimal solution, the right-hand side was parametrized in two ways and two broken-line trade-off curves were obtained between jet fuel and other refinery products. Using one parameter, it was possible to trace through the effects of varying the jet fuel requirement. The other parameter varied the availability of certain refinery equipment and was used only to get from one curve to the other. Each turning point on each curve represented the solution of a linear programming problem. These critical values of the parameter are determined automatically in this type of PLP and are sometimes of considerable interest, as in this case.

This model was aggregated to cut down the order of the system and parallel computations made to determine the sensitivity to detail. With appropriate adjustments, the aggregated version was then blown up into an inter-regional model involving transportation as well as production facilities. This model has 195 restraint equations in more than 1500 variables. The computations are proceeding well and some PLP is planned. In handling this many variables, however, it is wise to progressively sub-optimize on nested subsets of the variables, a procedure which has been followed on several occasions.

This last technique is well illustrated by a large air-transport scheduling model also developed by Dr. Manne [13]. The system was so large that the generation of the matrix itself was done by machine. Three different cases were run, all with 94 restraint equations plus a minimizing form, and with the following numbers of variables:

Case 1, 1476 variables,

Case 2, 1494 variables including those of case 1,

Case 3, 1553 variables including all but one of those of case 2.

First, however, a set of 94 vectors which were known to be linearly independent were pre-selected to form an initial basis which was inverted in 73 iterations, there being 21 unit vectors in the set. The inversion iterations go much faster than normal simplex cycles, and this inversion took about 50 minutes. This gave a solution which was infeasible in 8 variables. However, by using an expanded set of 123 vectors, including the original 94, the composite algorithm reduced this to one small infeasibility in 27 iterations of which 3 were repetitive ones to correct some errors which had been detected in the data.

The other 1353 vectors for case 1 were then made available and a feasible solution obtained in 8 cycles, meanwhile improving the minimizing form. After 75 more cycles, it became advisable to re-invert the current basis to reduce the length of the transformations tape. This required 89 of the fast iterations, and after 57 more simplex cycles, optimality was obtained for case 1. Cases 2 and 3 were picked up in suc-

cession in a similar manner. The total number of iterations for all cases was 739, of which 359 were fast inversion cycles. Thus 380 iterations were performed to obtain new results in solving all three problems, quite good, I believe, considering the very large number of variables.

Harry Markowitz has used the simplex codes at RAND to solve problems in which the variables are allowed to take on only integral values, or what might be called Diophantine linear programming [14]. This is not done by a single run, but by adding additional constraints from time to time on the basis of previous optimal solutions. A process of elimination is used to reduce the number of admissible variables, not by omitting them, but by forcing them above or below certain bounds with the added constraints. The bounds are computed on the side.

The code has been used for fitting t observations to m variables, minimizing the sum of the absolute values of the errors. This required using 2t + 2m variables as it was set up. This process could be made more efficient by a special routine but we have not taken time yet to code it. It would differ from the regular simplex method mainly in the way in which the variable to be eliminated is chosen on each cycle.

As is fairly well known, the simplex method is a good way to solve linear systems since it often gives valuable information even when no solution exists. We have also found the code very good for inverting badly conditioned matrices, if the order is not too big -- say 50, although the practical

value of such inverses is open to question.

We have recently been working on a nutrition problem which is unusual in that there are only 16 restraint equations, 3 of these somewhat artificial, but several hundred variables. Furthermore, the matrix is very full, in fact it contains more non-zero entries than any other problem we have run. It is b ing used to study objective functions and other aspects of model formulation, with some interesting and bizarre diets as a by-product.

We are already preparing a 704 code which will be essentially the same as our present 701 code. However, two new features are planned. In the 701, the original matrix is carried in condensed form, that is, only non-zero elements are recorded and these are indexed. We plan to do the same with the transformations on the 704. This, together with the higher speed of the 704 itself, should enable us to handle 200 or perhaps 250 order systems in very acceptable times. We now feel that this is an upper limit for a general routine, not only because of time, but also because round-off error again becomes troublesome at this point even with double precision numbers.

The other feature is variables with upper bounds. This can be handled without much cost either complexity or time if certain conventions are adopted. Since several restraint equations are sometimes used merely for bounding certain activities, this will allow us to handle effectively bigger systems for such problems.

We have a code for RAND's own machine, the JOHNNIAC, which uses a different elimination method, developed by Harry Markowitz, for inverting a matrix [15]. It is designed to maintain as many zeros as possible in the representation of the inverse of a sparse matrix, as linear programming bases typically are. The code has not yet been in operation long enough to assess its value but the method is compatible with the simplex algorithm. Beyond this, we know of no other ways to make a general routine handle larger systems efficiently. Incidentally, it seems odd that no one, so far as I know, has before attempted to solve a linear system on a machine using the same rules of thumb that one would use in solving by hand, i.e. to eliminate first on the rows and columns having the fewest elements.

Alan Manne noted that his air-transport model had a matrix structure which was a special case of block triangularity. Upon investigation, he found that many problems fell into this pattern. He and Dr. George Dantzig worked out a means of exploiting this structure and they, Ted Robacker, and myself are now working out the details of a machine method to take advantage of magnetic tape characteristics. Manne is planning to spend considerable time investigating such semi-specialized methods and this is currently the most promising means of handling larger systems.

References

[1] A. Charnes, W. W. Cooper, and A. Henderson, An Introduction to Linear Programming, New York, John Wiley & Sons

[2] G. B. Dantzig, A. Orden, and P. Wolfe, The Generalized Simplex Method for Minimizing a Linear Form Under Linear Inequality Restraints, RM-1264, The RAND Corp., Santa Monica, Calif.

[3] W. Orchard-Hays, Background, Development and Extensions of the Revised Simplex Method, RM-1433, The RAND Corp.

[4] G. B. Dantzig, Developments in Linear Programming, RM-1426, The RAND Corp.

[5] A. Charnes and C. E. Lemke, "Computational Theory of Linear Programming: I. The Bounded Variables Problem", O.N.R. Research Memorandum No. 10, Carnegie Institute of Technology

[6] W. Orchard-Hays, Computational Experience in Solving Linear Programming Problems, P-482, The RAND Corp.

[7] A. S. Manne, Petroleum Refinery Operations Scheduling, Chap. VI, P-493, The RAND Corp.

[8] G. B. Dantzig and W. Orchard-Hays, "The Product Form for the Inverse in the Simplex Method", Mathematical Tables and other Aids to Computation, VIII, No. 46, April 1954.

[9] G. B. Dantzig, W. Orchard-Hays and G. Waters, Product-Form Tableau for Revised Simplex Method, RM-1268-A, The RAND Corp.

[10] W. Orchard-Hays, The RAND Code for the Simplex Method (for the IBM 701), RM-1440, The RAND Corp.

[11] G. B. Dantzig, The Dual Simplex Algorithm, RM-1270, The RAND Corp.

[12] A. Manne, A Linear Programming Model of the U.S. Petroleum Refining Industry, P-563, The RAND Corp.

[13] A. Manne, Air Cargo Transport Scheduling -- An Illustrative Block Triangular System, P-533, The RAND Corp.

[14] A. Manne and H. Markowitz, On the Solution of Discrete Programming Problems, P-711, The RAND Corp.

[15] H. Markowitz, The Elimination Form of the Inverse and its Application to Linear Programming, RM-1452, The RAND Corp.