Some experiences in price mapping

By Lucy Joan Slater

This paper outlines the details of an efficient computer program for price mapping, i.e. the production not only of the optimum solution, but also the surrounding "next best" solutions to a given linear-programming problem.

Price mapping is one of the first generalizations to arise from pure linear programming. A resource map is a dual version of the same type of calculation. The process of calculating a price map has two stages. The first stage is the determination of the normal optimum solution of a linear program by the usual simplex process for linear programming. The complete matrix forming this solution is called the *optimum plan*. Every one of the constraints in a linear program holds over a given range of values of any one selected variable, and the end points of such a range of values can be calculated quickly. In two dimensions, when two of the constraining variables are allowed to vary, the optimum plan is seen to hold over a given region in a plane, and this region is bounded by linear segments. It is the object of the first stage of a price mapping program to calculate, not only the optimum plan, but also the coefficients which determine these boundary lines of the optimum region, for two selected variables, together with the co-ordinates of their points of intersection.

The second stage of a price mapping program consists of a seeking process to determine the boundaries of the regions in which all the "next best" plans hold. These regions surround the optimum region. Later repetitions of this second stage of the program substitute every one of these "next best" plans in turn for the optimum plan, and use these new plans to seek further solutions, which, in their turn, are valid over regions neighbouring those regions already found. In this way a complete price map is built up, showing every possible plan and the region over which it is valid. It can be proved rigorously that this seeking process is a finite one and that it must come to an end sometime.

The normal simplex process can be thought of, in economic terms, as that of calculating the maximum profit on a given investment of money and other resources, such as labour, land and materials. The solution consists of a matrix, the optimum plan; the elements of the first column vector in this matrix are the amounts of the various resources required to produce this best possible plan.

Such models are very crude, even when the resulting matrix exceeds 100 by 100 in size. As well as the practical difficulties of performing the calculations on such large matrices, there are also difficulties in the actual measurement of the observed coefficients, namely the resources. These models can, at best, provide only a coarse outline picture of this best possible plan. The picture could be made much more detailed if we could measure subjective elements, such as managerial preferences and workers' prejudices in money terms. But this would introduce still more elements into our matrix. Instead, it is often more acceptable to the person who wants the answers, to be given a price map showing the region in which the best possible plan holds, and also all those other regions in which the "next best" plans hold, plans resulting from the relaxation of every one of the boundary restrictions in turn.

The outlines of the simplex technique are well known, but little has been written about the computational details of a really efficient price mapping program for a fast electronic computer. It is the object of this paper to fill this gap.

Two difficulties were encountered in practice. First it is necessary to design the process of price mapping in such a way that the calculation comes to an end sooner rather than later. Most early programs (see Cran, 1961) were aimed primarily at satisfying this requirement, particularly those designed for small or slow computers. However, it was soon found that most of the methods (see Heady and Candler, 1960) proposed in the standard text books omitted a few of the possible "next best" solutions altogether.

Mr. E. Evans, now at Aberystwyth, worked out part of one of these price maps by using a desk calculator, so that the fault in the seeking process, which was then in use, could be traced and corrected. The final outcome was the present program, developed for use on the Cambridge University electronic computer, EDSAC 2. We shall assume in this paper that the first part of the problem, that of collecting the data and of setting it up in a form suitable for the application of a price mapping program, has already been carried out by the economists, and that only the calculations remain to be worked out.

Calculating procedure

The data consists of a column of arbitrary indices, the I column, which can be used to identify the rows of the data, a row of indices, the J row, which identifies the columns of the data, the initial cost row C_0, two variable cost rows C_1, and C_2, and the main data, with the activity level k_i (or supply) standing first in every row.

The program has three sections and each section is organized into a number of subroutines.

Section I

This first section of the program reads and checks the data. It can be entered at a number of points. The primary entry reads and stores the I column, the J row, and the main data, which can occur either in rows or

in columns. This original data is stored on magnetic tape, so that once read in, it can be varied slightly by the input of short correction tapes, and a number of price maps formed from several similar sets of data. The program then checks that all the items in the first, the K column, of the data, are in fact $\geqslant 0$, and then goes on to the second stage of the calculation.

The data is now set up in the main store of the machine thus:

	K	C_1	C_2	J_1	J_2	...		J_n
C_0	0	0	0	0	0	c_{03}	c_{04}	$\ldots c_{0n}$
C_1	0	1	0	−1	0	0	0	$\ldots 0$
C_2	0	0	1	0	−1	0	0	$\ldots 0$
I_1	k_1	0	0	a_{11}	a_{12}	a_{13}	a_{14}	$\ldots a_{1n}$
I_2	k_2	0	0	a_{21}	a_{22}	a_{23}	a_{24}	$\ldots a_{2n}$
.....								
I_m	k_m	0	0	a_{m1}	a_{m2}	a_{m3}	a_{m4}	$\ldots a_{mn}$

The two dummy rows, C_1 and C_2, introduced here, will contain, later, the elements of the two restricting vectors which we are going to relax. If our price map had more than two dimensions, we would, of course, have to introduce more than two variable cost rows here, one for each added dimension. Such multi-dimensional mapping problems are still outside the range of all but the fastest electronic computers. The C_1 and C_2 columns are used by the program for internal checking purposes.

Section II

This section maximizes

$$c_{00} \equiv \sum_{j=1}^{n} c_{0j} z_j,$$

subject to the constraints

$$\sum_{j=1}^{n} a_{ij} z_j \leqslant k_i$$

for $i = 1, 2, 3, \ldots, m$.

The program enters a subroutine 10, which seeks the most negative element c_s along the first row of the main data, excluding the first element c_{00}. If such a negative c_s is found, the program enters another subroutine 13. This seeks down the K column and the sth column. If $k_i = 0$ or if $a_{is} = 0$, it ignores this value of i; otherwise it forms the ratios

$$\theta_i = k_i/a_{is}, \text{ for } k_i \neq 0, \text{ and } a_{is} \neq 0.$$

It seeks the smallest positive of these ratios θ_r and selects a_{rs} as the pivot element. If no such θ_r can be found, the program reports that the maximum is infinite and the rest of the calculation is impossible, but if such a θ_r is found then the program goes on to enter subroutine 11. This subroutine divides the elements a_{rj} along the pivot row r, by a_{rs}, the pivot, so that

$$a_{rj} \to a_{rj}/a_{rs}, \text{ for } j \neq s.$$

It then replaces the pivot a_{rs} by its inverse $1/a_{rs}$, and it transforms the rest of the elements in the pivot column so that

$$a_{is} \to -a_{is}/a_{rs}.$$

It transforms all the remaining elements a_{ij} of the entire data matrix so that

$$a_{ij} \to a_{ij} - a_{is}a_{rj}/a_{rs} \text{ for } i \neq r, j = s,$$

and it interchanges the coefficients in the I column and J row, so that

$$I_s \to J_r \text{ and } J_r \to I_s.$$

The program seeks along the C_0 row for any element c_{0j} such that

$$|c_{0j}| < \sum_{j=1}^{n} \{|c_{0j}|/n\}10^{-5}.$$

It replaces any such arbitrarily small elements by zeros. This is a very necessary precaution as these small elements might give rise to infinite cycling later on in the calculation.

This is the standard simplex process. The program prints I_s and J_r at every cycle, and repeats the cycle from the entry to subroutine 10 onwards, until no negative element c_s can be found by subroutine 10. The simplex process is then finished, and the first plan P_0, consisting of the entire matrix, with its attendent I column and J row, is now in the main store. The program stores, in a separate sequence, the vector $\{l_0, m_0, n_0\}$ which consists of the first three elements of the first column of plan P_0. This vector C_0 identifies the given plan P_0 completely and uniquely, since no two regions of any price map can overlap. It is called, by economists, the *return to plan* vector, and by mathematicians, the *identifying* vector of the plan. The program stores the entire plan P_0 on magnetic tape, and enters the next subroutine, 21, to print the values of i, j and the corner point

$$(l_im_j - l_jm_i)/(m_in_j - m_jn_i), (n_il_j - n_jl_i)/(m_in_j - m_jn_i)$$

of this plan. In this first case, $i = C_1$, $j = C_2$ and the point printed is the origin (0, 0), the first corner of the price map. The program then prints the elements of the I column and the first or K column in pairs

$$I_i, k_i \text{ for } i = 0, 1, 2, \ldots, m + 2,$$

and it then sets up various counts ready to enter the third section of the program. If an optional stop is introduced into the program at this point, the same program can be used to solve standard linear programming problems, as the solution P_0 which has now been printed, is in fact, the standard simplex solution.

Section III

This section deals in a comprehensive way with the difficult problem of relaxing every possible restriction on every known plan in turn, to find every possible new plan. Again, the program is organized, as in the previous

sections, as a short main program which controls entry to a number of subroutines.

At the first entry to Section III, one plan P_0 is in the main store, together with one "return to plan" vector, or C vector, $\{c_{00}, c_{10}, c_{20}\}$, and the count over all known plans stands at one, since only plan P_0 is as yet known. In the first place, we test the C_1 and C_2 rows of the current plan P_t to find all possible restrictive vectors which limit the plan P_t. Thus, if the elements of the row C_1 are $c_{11}, c_{12}, c_{13}, \ldots, c_{1,n+2}$, and the elements of the row C_2 are $c_{21}, c_{22}, c_{23}, \ldots, c_{2,n+2}$, we have to form every possible determinant

$$A_{ij} \equiv \begin{vmatrix} c_{1i} & c_{1j} \\ c_{2i} & c_{2j} \end{vmatrix}.$$

For a vector $\{c_{0i}, c_{1i}, c_{2i}\}$ to be a restrictive vector limiting the plan P_t, it is necessary that $A_{ij} \neq 0$, for the selected i and for some value of j, and also it is necessary that the determinant

$$B_{ijk} \equiv \begin{vmatrix} c_{0i} & c_{0j} & c_{0k} \\ c_{1i} & c_{1j} & c_{1k} \\ c_{2i} & c_{2j} & c_{2k} \end{vmatrix}$$

should be $\geqslant 0$ for all possible values of $k = 1, 2, \ldots, n+2$. So we have to form and test A_{ij}, counting over $i = 1, 2, 3, \ldots, n+2$, and $j = i, i+1, i+2, \ldots, n+2$, until we find the first pair of values for i and j such that $A_{ij} \neq 0$. Then, with this pair of values for i and j, we have to form every possible B_{ijk} and test that $B_{ijk} \geqslant 0$, for every value of k.

When such a pair of values for i and j has been found, we print the pair of restrictive vectors

$$\{c_{0i}, c_{1i}, c_{2i}\} \text{ and } \{c_{0j}, c_{1j}, c_{2j}\}$$

and store them in the sequence R_t of restrictive vectors on the plan P_t. Then we continue our counting, seeking and testing over all the remaining values of i and j until both the i and j counts are exhausted. The sequence R now contains, in r pairs, a record of all the restrictive vectors on the plan P_t.

We are now in a position to go on to generate further pairs of plans, $P_t, P_{t+1}, \ldots$, by relaxing each of the r pairs of restrictions in turn. Each plan P_t is completely identified by the first three elements in its first column, that is, its C vector $\{c_{0t}, c_{1t}, c_{2t}\}$. This vector is checked against the sequence of stored vectors which defines the plans already found. If P_t has already been found it is ignored, but if P_t is a new plan it is stored in its entirety, on the end of the sequence of known plans already on the magnetic tape, and its C vector is stored on the end of the sequence of identifying vectors.

In this way, we can relax every one of the $2r$ restrictions on plan P_t in turn, and when all possible new plans have been found and tested, and the r count is exhausted, we can advance the t count to $t+1$, and return to the start of Section III to continue our search for new plans from a new starting plan P_{t+1}.

Finally, the t count too will become exhausted as the number of known plans increases, and the likelihood of finding a completely new plan decreases. However, if this point is likely to take a long time to reach, the region of the search can be limited by the introduction of four boundary conditions into the initial data. These consist of four columns which have all their items zero except the first three. These three items take the values

$$(0, 1, 0)\ (0, 0, 1)\ (A, 1, 0)\ (B, 0, 1) \quad \text{where}$$

$(0, 0)$ $(0, A)$ (B, A) $(B, 0)$ are the corner points of the rectangular region in which we are interested.

The present main store of EDSAC 2 gives a capacity of $m, n \leqslant 100$, and a complete price map of this size takes about 6 minutes to complete the linear programming stage, and then about 30 seconds to generate every new plan. The seeking process, outlined here, tests every possible combination of plans, and although sometimes it will print a known plan a second time, in a heavily disguised form, it cannot omit any possible plan.

An example

The following gives in full the example worked out by Mr. Evans.

Initial data, I_0

	K	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8
x_1	2·10	1	2	0·25	0·5	0·25	0·25	0	0·5
x_2	1·80	5	0·25	0	0	0	0	0	0
x_3	1·55	1	0·25	0	0	1	1	0	0·5
x_4	1·95	0	0	0·5	0·5	0·5	0	0·5	1·25
x_5	2·45	3	2·5	0·5	0·25	0	0	0·5	0·25
x_6	2·05	0	3·5	0	0	0	0	0	0
x_7	0·50	1	0	0	0	0	0	0	0
x_8	0·50	0	1	0	0	0	0	0	0
x_9	1·50	0	0	(1)	0	0	0	0	0
x_{10}	2·25	0	0	1	1	0	0	0	0
x_{11}	0·50	0	0	0	0	1	1	0	0
x_{12}	3·00	1	1	1	1	1	1	1	1
C_1	0	−1	0	0	0	0	0	0	0
C_2	0	0	−1	0	0	0	0	0	0
C_0	0	0	0	−0·32	−0·26	−0·15	−0·10	−0·18	−0·10

Here the numerically greatest negative item in the C_0 row is in column A_3, $-0{\cdot}32$, and the smallest positive ratio $K/A \neq 0$ is $x_9/A_3 = 1{\cdot}5$. Hence the pivot is at the junction of the x_9 row and the A_3 column, $a_{93} = 1$.

The first iteration leads to $A_3 \leftrightarrow x_9$. The second iteration leads to $A_4 \leftrightarrow x_{10}$. The third iteration gives $A_7 \leftrightarrow x_{12}$.

I_3									
x_1	1·35	1	2	0·25	−0·5	0·25	0·25	0	0·5
x_2	1·8	5	0·25	0	0	0	0	0	0
x_3	1·55	1	0·25	0	0	1	1	0	0·5
x_4	0·45	−0·5	−0·5	0	0	0	−0·5	−0·5	0·75
x_5	1·1375	2·5	2	−0·25	−0·25	−0·5	−0·5	−0·5	−0·25
x_6	2·05	0	3·5	0	0	0	0	0	0
x_7	0·5	1	0	0	0	0	0	0	0
x_8	0·5	0	1	0	0	0	0	0	0
A_3	1·5	0	0	1	0	0	0	0	0
A_4	0·75	0	0	−1	1	0	0	0	0
x_{11}	0·5	0	0	0	0	1	1	0	0
A_7	0·75	1	1	0	−1	1	1	1	1
C_1	0	−1	0	0	0	0	0	0	0
C_2	0	0	−1	0	0	0	0	0	0
C_0	0·81	0·18	0·18	0·06	0·08	0·03	0·08	0·18	0·08

There is now no negative item in the cost row, C_0, so this is the best possible solution, plan P_{00}, identified by the vector

$$\{l, m, n\} = \{0{\cdot}81, 0, 0\}.$$

This plan is valid under the two restrictions

$$-x + 0{\cdot}18 \geqslant 0, \text{ and } -y + 0{\cdot}18 \geqslant 0, \text{ and we}$$

can now proceed to seek those plans that are the next best, which arise from the relaxation of either one of these two restrictions. We can remove such a restriction by carrying out one simplex iteration process, as above, with the column defined by the restrictive vector in place of the column with the greatest negative item in the cost row.

If we relax the first restriction, we find a new plan, $A_1 \leftrightarrow x_2$.

P_{01}									
x_1	0·99	−0·2	1·95	0·25	−0·5	0·25	0·25	0	0·5
A_1	0·36	0·2	0·05	0	0	0	0	0	0
x_3	1·19	−0·2	0·2	0	0	1	1	0	0·5
x_4	0·63	0·1	−0·475	0	0	0	−0·5	−0·5	0·75
x_5	0·2375	−0·5	1·875	−0·25	0·25	−0·5	−0·5	−0·5	−0·25
x_6	2·05	0	0·35	0	0	0	0	0	0
x_7	0·14	−0·2	−0·05	0	0	0	0	0	0
x_8	0·5	0	1	0	0	0	0	0	0
A_3	1·5	0	0	1	0	0	0	0	0
A_4	0·75	0	0	−1	1	0	0	0	0
x_{11}	0·5	0	0	0	0	1	1	0	0
A_7	0·39	−0·2	0·95	0	−1	1	1	1	1
C_1	0·36	0·2	0·05	0	0	0	0	0	0
C_2	0	0	−1	0	0	0	0	0	0
C_0	0·7452	−0·036	0·171	0·06	0·08	0·03	0·08	0·18	0·08

This plan is valid under the restrictions

$$0{\cdot}05\,x - y + 0{\cdot}171 \geqslant 0, \text{ and } 0{\cdot}2\,x - 0{\cdot}036 \geqslant 0.$$

It is identified by the vector $\{0{\cdot}745, 0{\cdot}36, 0\}$. Further plans found by Evans are

$$P_{012} = \{0{\cdot}724, 0{\cdot}354, 0{\cdot}127\}$$
$$P_{02} = \{0{\cdot}720, 0, 0{\cdot}5\}$$

$$P_{021} = \{0{\cdot}710, 0{\cdot}055, 0{\cdot}5\}$$

and $P_{0211} = \{0{\cdot}699, 0{\cdot}087, 0{\cdot}5\}$.

These calculations took several days with a hand calculating machine. The computer generated all these known plans in about three minutes, and, in fact, one further plan was found by EDSAC 2:

$$P_3 = \{0{\cdot}707, 0{\cdot}035, 0{\cdot}31\}.$$

References

CRAN, J. (1961). "Calculating optimum farm programs," *M.Sc. Thesis*, Cambridge.

HEADY, E. O., and CANDLER, W. (1960). *Linear programming methods*, Chap. 8, Iowa State Press.