Assignment problems

By J. S. Clowes and E. S. Page

Assignment problems and some methods of solution are described in an expository manner.

Examples of assignment problems arise when restrictions of the type "one man to one job" exist. If a number of vacant jobs has to be filled from a list of applicants each of whom may be qualified for more than one job, there may be no way of assigning all the men to jobs they can do, or there may be many ways, some of which are more satisfactory than others. The problem is to find the most satisfactory one. There are many different criteria by which the assignments could be judged; only two have received much treatment. A "value" or "cost" is supposed known for every man in every job. The two criteria are the sum of the values in an assignment of men to jobs, or one of the extremes (i.e. the greatest or least) of these values. Problems using the sum criterion are *classical* assignment problems while those using an extreme are *bottleneck* problems.

The simplest classical case is that in which all the values are zero or one. This "simple" assignment problem requires an assignment which places as many as possible of the men in jobs that they can do. The other classical cases will have a more general value matrix (Γ_{ij}), where Γ_{ij} is the value of assigning the ith man to the jth job. If, for example, Γ_{ij} is the expected value of orders obtained by the ith salesman if he is assigned to the jth sales area, we could want to maximize $\sum_{i=1}^{n} \Gamma_{ij}$ where the suffices j are all different and so contain the integers $1, 2, \ldots, n$ just once each. If the Γ_{ij} are the costs of travelling for man i to area j then the problem could be that of minimizing the total cost, $\sum_{i=1}^{n} \Gamma_{ij}$.

Let us suppose, picturesquely, that the Γ_{ij}, instead of being the costs of travel, are the times needed to get to duty stations and that the men are secret agents who all have to be in position before an operation can begin. Then the aim is to minimize the greatest of the Γ_{ij} for the men assigned. Here there is a bottleneck caused by the slowest of the agents reaching his post. Problems of this type so get their name; a corresponding one requires the maximization of the least of the Γ_{ij}.

The problems we consider are therefore as follows.

Given an $n \times n$ matrix (Γ_{ij}), we require to assign one element from each row and one from each column so that

$$\text{(1A)} \quad \sum_{i=1}^{n} \Gamma_{ij} \quad \text{is maximized} \qquad (1)$$

$$\text{(2A)} \quad \sum_{i=1}^{n} \Gamma_{ij} \quad \text{is minimized} \qquad (2)$$

$$\text{(1B)} \quad \underset{j}{\text{Min}}(\Gamma_{ij}) \quad \text{is maximized} \qquad (3)$$

$$\text{(2B)} \quad \underset{j}{\text{Max}}(\Gamma_{ij}) \quad \text{is minimized.} \qquad (4)$$

In every case the suffices, j, form some re-arrangement of the numbers $1, 2, \ldots, n$.

So far the formulation has supposed that the numbers of men and jobs are the same, i.e. the matrix (Γ_{ij}) is square. If this is not the case additional rows or columns can be added and filled with suitable numbers. For example, in problems 1A and 2A the new elements could be set equal to any constant; in problem 1B the constant may be any number greater than the largest element of (Γ_{ij}) and similarly for 2B. We suppose that these additions have been made where necessary.

Equivalent problems

Problems 1A and 2A can be easily recognized as special cases of transportation problems. If we let $x_{ij} = 1$ if the ith man is assigned to job j, and 0 otherwise then the x's must satisfy the following conditions stating the "one man, one job" principle.

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 1, 2, \ldots, n \qquad (5)$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 1, 2, \ldots, n \qquad (6)$$

$$x_{ij} \geqslant 0 \quad i, j = 1, 2, \ldots, n. \qquad (7)$$

These conditions coupled with the target for optimization (1) or (2) specify the transportation problem. Since routines for solving transportation problems exist for most computers, for many applications the search for a method of solution ends here. However, the conditions (5) and (6) are simpler than those in the general transportation case and alternative techniques have been devised to take advantage of them.

The bottleneck problems have been shown by one of us (Page, 1963) to be reducible to the corresponding classical case but, except for the smallest problems, the size of the numbers involved prevents the practical exploitation of the procedure. Consequently there is some interest in developing methods for assignment problems alone. Some methods appear to trade a quantity of arithmetic for an increase in the complexity of the logic, but the savings in operations are not always easy to assess.

Transforming assignment problems

Let us first consider problems (1A) and (2A). Since a maximization problem can be turned into a minimization one by changing the signs of all the Γ_{ij} we need only consider one of them, say the minimization problem, 2A. If all the values Γ_{ij} were increased by a constant to $\Gamma_{ij} + c$ the cost of any assignment would be increased by an amount nc, but the best assignment of men to jobs will still be the same for both value matrices. Similar comments apply to the bottleneck problems; here the optimum bottleneck value would be increased by c but the best assignment would be unchanged. Accordingly we can always make the entries Γ_{ij} positive, and we suppose that this has been done. Further, we can suppose in all practical problems that all the Γ_{ij} are integers, since their multiplication by a positive constant does not change an optimum assignment.

As we are trying in problem 2A to minimize a sum of positive elements it is clear that if we can include only zero elements of (Γ_{ij}) in our assignment we have succeeded in finding an optimum solution. It is unlikely that the matrix as given will contain such an assignment, but we can modify the matrix successively until it does. We can add a constant to any row or column of the (Γ_{ij}) without changing the optimum nature of an assignment, since each row and column is represented once and once only. In a practical example where the Γ_{ij} are distances between men and jobs, the problem of finding the optimum is clearly unchanged if one of the jobs is placed the same distance farther away from all the men. An attack can thus be made upon the problems 1A and 2A. Constants are subtracted from rows and columns in order to generate some zeros in the matrix. If an assignment can be completed from the positions of these zeros the solution has been found; if not, some further juggling is necessary—for example, constants can be added to a row and subtracted from a column to produce zeros in different positions.

Example

Suppose that the value matrix is as shown in (i).

$$\begin{array}{ccccc} 8 & 3 & 11 & 13 & 14 \\ 0 & 6 & 15 & 0 & 3 \\ 9 & 6 & 2 & 1 & 1 \\ 3 & 7 & 4 & 1 & 8 \\ 5 & 7 & 8 & 2 & 10 \end{array} \qquad \begin{array}{ccccc} 5 & 0 & 8 & 10 & 11 \\ 0 & 6 & 15 & 0 & 3 \\ 8 & 5 & 1 & 0 & 0 \\ 2 & 6 & 3 & 0 & 7 \\ 3 & 5 & 6 & 0 & 8 \end{array}$$

(i) (ii)

Then we can subtract the smallest element of each row from all the elements of that row and so reach (ii). A similar operation on the columns of (ii) produces (iii);

$$\begin{array}{ccccc} 5 & 0 & 7 & 10 & 11 \\ 0 & 6 & 14 & 0 & 3 \\ 8 & 5 & 0 & 0 & 0 \\ 2 & 6 & 2 & 0 & 7 \\ 3 & 5 & 5 & 0 & 8 \end{array} \qquad \begin{array}{ccccc} 5 & \boxed{0} & 7 & 12 & 11 \\ \boxed{0} & 6 & 14 & 2 & 3 \\ 8 & 5 & 0 & 2 & \boxed{0} \\ 0 & 4 & \boxed{0} & 0 & 5 \\ 1 & 3 & 3 & \boxed{0} & 6 \end{array}$$

(iii) (iv)

this matrix has many zeros but rows 4 and 5 have them only in column 4, and so there can be no zero assignment. We can now subtract 2 from rows 4 and 5 and add 2 to column 4. The selection of these rows and columns is discussed later. These operations leave the existing zeros in the important fourth and fifth rows unchanged and create some new ones—although others in column 4 are lost (iv). In this example it is readily apparent that a zero assignment exists—the boxed positions (man 1 to job 2, 2 to 1, 3 to 5, 4 to 3, 5 to 4). By referring to the original matrix the minimum value of the cost is $3 + 0 + 1 + 4 + 2 = 10$.

Recognition of an assignment

Sometimes it is easy to see whether a zero assignment exists. For example we can assign, one by one, zeros which are the only ones in rows and delete any other zeros in the same columns; sometimes this process completes the assignment, or will do so when the operations are performed on the remaining columns. But it can happen that there are alternatives left, and they can be very numerous and yet fail to include any zero assignments. How to decide whether or not a matrix possesses a zero assignment is one of the crucial problems (the other is which transformations to perform if the matrix does not as it stands).

To determine whether or not the matrix possesses a zero assignment the idea of a *cover* is introduced. A cover is a set of rows and columns ("lines" for short) which between them contain all the zero elements of the matrix. Thus in matrix (i), row 2 forms a cover. A theorem of König states, in effect, an $n \times n$ matrix possesses a zero assignment only if every cover contains at least n lines, i.e. no covers of less than n lines.

Many methods (some arising in other applications) have been proposed for finding a cover with less than n lines if such exists (Kuhn, 1956). The simplest (but most laborious) is probably that due to Porsching (1963).

Porsching's algorithm involves the examination of every submatrix consisting of $1, 2, \ldots, n$ rows of the value matrix Γ ($2^n - 1$ in all). If every such submatrix of k rows, $k = 1, \ldots, n$, contains at least k columns with zero entries then a zero assignment exists. But if a submatrix S of k rows contains $l < k$ columns with zero entries then these l columns together with the other $n - k$ rows of Γ not in S form a cover of $n - k + l = n - (k - l) < n$ lines.

In matrix (iii) the submatrix consisting of rows 4 and 5 contains zeros only in column 4 and this column together with rows 1, 2 and 3 form a cover of four lines (v).

$$\begin{array}{ccccc} 5 & 0 & 7 & 10 & 11 \\ 0 & 6 & 14 & 0 & 3 \\ 8 & 5 & 0 & 0 & 0 \\ 2 & 6 & 2 & 0 & 7 \\ 3 & 5 & 5 & 0 & 8 \end{array}$$

(v)

This algorithm checks to see if a zero assignment exists, but does not give one directly; additional operations are needed.

A superficially attractive idea for detecting a minimal cover of n lines is to replace the zero entries by suitably chosen, or even random, values, and all other entries by zero. If the determinant of this array, evaluated by standard methods, is non-zero at least one term is non-zero, and so a cover of n lines exists. Unfortunately practical application of this suggestion encounters snags but further work may remove them.

Generation of new zeros

When it is necessary to create new zeros the operations of subtracting a constant, α, from a row and adding α to a column are performed. This leaves the element of the matrix at the intersection unchanged but can create zeros elsewhere.

If we have found a cover of r rows and c columns with $r + c < n$ let m be the smallest matrix element outside the lines of the cover. For example, in (v) $m = 2$. We now subtract m from the $n - r$ rows *not* in the cover and add m to the c columns in the cover. This process when applied to (v) yields (iv), which has two additional zeros outside the cover in row 4 (at the places where the minimum m occurred).

In general, we get at least one zero not in the cover and all elements remain non-negative, since the only ones actually decreased are those not in the cover and these were all at least as big as m. If the transformed matrix still has no zero-assignment we can repeat the process. However, this cannot go on indefinitely. For, on subtracting m from the $n - r$ rows not in the cover we reduce the sum of the matrix elements by $n(n - r)m$, while the addition of n to the c columns in the cover increases it by ncm, a net reduction of $n(n - r + c)m = n(n - (r + c))m \geqslant 1$. Since we cannot go on for ever reducing the sum of the matrix elements by an integral quantity while still retaining non-negative elements the process must terminate after a finite number of steps with a matrix possessing a zero assignment.

Alternative methods

The technique described above, for deciding whether or not a matrix has a cover of less than n lines, has the disadvantage that it does not give the zero assignment when one exists, and also it requires a very large number of operations. Alternative methods have been developed by Kuhn (1955) and others (Kuhn, 1956) which get round both of these snags.

The principle of the methods is to try to build up an assignment and to improve what has been attained if it fails to meet all the conditions. One first starts making an assignment in some simple way—for example by considering each column in turn. If this fails to allocate just one man to each job some of the steps are retraced and an alternative chain of assignments is explored. A simple outline of one of these methods, the Hungarian, is given by Vajda (1961). These methods are likely to reach the optimum assignment in fewer steps than Porsching's but they pay the price by being more complicated.

Bottleneck problems

These problems (1B and 2B) are in most respects simpler than the classical ones. We can again consider the elements of the matrix to be positive integers in any practical case, since the addition of a constant to all the elements leaves an optimum assignment unchanged. In this case we can go further and suppose that the integers are consecutive commencing at any convenient place such as zero or unity, since it is only the order of the cost values which is relevant; some integers may, of course, be repeated. The transformations which could be carried out on rows and columns in the classical case are no longer available. If the problem is a maximization one (1B) then we can clearly search for an assignment in an $n \times m$ problem using first the largest elements, the next largest, and so on until at least n elements are included. If an assignment is detected no further steps are needed. If not, more equal elements are introduced stage by stage until an assignment exists.

Methods for the classical problem of building up an assignment can be modified for the bottleneck problems (Gross, 1959; Page, 1963).

Conclusion

Since computer programs for the transportation problem exist for most machines, isolated classical assignment problems of sizes that can be handled are likely to be performed using the more general approach. Typically assignments of 56×56 can be made on Pegasus and 128×128 on Mercury, without magnetic tapes, and larger programs will be possible on the newer machines. For those problems too large for the transportation approach yet still within the capacity of the machine, the task of writing special progiams can hardly be avoided. Naturally the regular and frequent appearance of even medium-sized assignment problems could make it worth while to tackle one of the more complicated but efficient algorithms; but these circumstances have so far been rare. The bottleneck problems present a different story; here they need one of the modified algorithms for the classical case, and if these have to be written, the little extra effort to cover both types of assignment could be faced.

References

GROSS, O. A. (1959). "The Bottleneck Assignment Problem," Rand Report P-1630.
KUHN, H. W. (1955). "The Hungarian Method for the Assignment Problem," *Nav. Res. Log. Quart.*, Vol. 2, p. 83.
KUHN, H. W. (1956). "Variants of the Hungarian Method for Assignment Problems," *Nav. Res. Log. Quart.*, Vol. 3, p. 253.
PAGE, E. S. (1963). "A note on Assignment Problems," *The Computer Journal*, Vol. 6, p. 241.
PORSCHING, T. A. (1963). "Matrix Assignments and an Associated Min Max Problem," *Math. Comp.*, Vol. 17, p. 81.
VAJDA, S. (1961). *Mathematical Programming*, Addison-Wesley: Reading, Mass.

Book Reviews

Contd. from p. 303

puter design. It is becoming increasingly important that means should be found for reducing the dependence of the performance of a system on that of its components. For various reasons, components may be inaccessible for replacement or repair, or the system must operate continuously, allowing no opportunity for maintenance. The idea of using some form of redundancy to achieve this end is by no means new, even in the computer field. It cannot be claimed that redundancy offers a technique where, by using enough equipment, one can achieve an arbitrarily high reliability. On the contrary, as is shown in one paper in this book, in certain by no means unlikely circumstances, by using enough equipment one can achieve an arbitrarily high unreliability. Nevertheless, there are fields of application where the various techniques may have a beneficial effect, and one of the achievements of this book is to begin to define these fields. One fact brought home by a reading of this book is that all too little is known of the statistical behaviour of large assemblies of components, how this behaviour is influenced by the properties of the components, and even the properties of the components themselves. If the publication of this book has the effect of stimulating research into these problems, it will have performed a great service to the designers and users of future generations of computers.

J. B. STRINGER.

Machine Independent Computer Programming, by MAURICE H. HALSTEAD, 1962; 267 pages. (Washington, D.C.: *Spartan Books*, $6.50.)

While we have increasing interest in and discussion of many varieties of mathematical and commercial source languages little has been published on the actual process of translation of a language to machine code, particularly in a manner that the student or non-specialist programmer can readily understand. Dr. Halstead has provided us with a primer of compiling technique, and we must be glad that he has been able to strip away the cabalistic overtones of the process so neatly.

The subject of this book is the Nelliac language and the minimum compiler required to compile itself in that language. Appendices illustrate actual compilers, expressed in the source language, to run on three different machines, and a Nelliac program for one of these that will accept absolute octal machine code as input and will therefrom produce Nelliac statements—a decompiler.

Nelliac is claimed to be a dialect of ALGOL 58; the latter was doubtless the inspiration of the Nelliac effort but the notation and format of the language differ so much that it is perhaps better considered a separate language. Remembering that the first rudimentary Nelliac compiler began running in February 1959 this difference is hardly a fault. Most of the notational flexibility of ALGOL in such matters as recursion, block structure, variable arrays, compound conditionals, formal parameters and functions has disappeared in Nelliac, deliberately, to leave a language suitable for single-pass compilation to absolute machine code. Unfortunately the resulting notation is unnecessarily difficult to read, and probably to write, accurately, particularly in the case of subroutines and conditional expressions. The required niceties of punctuation lack the redundancy required for easy comprehension.

The Nelliac language is more slanted towards data processing than either ALGOL or FORTRAN, and the user is more aware that he is using a computing machine. An example of the latter is the very useful implicit definition of the machine working store as an array in the language. This feature would seem necessary in a self-compiling compiler. The reader should note that the terms "function" and "procedure" have very different meanings in Nelliac from those more widely accepted through the influence of ALGOL.

The whole family of Nelliac compilers uses the technique of "generators", where the occurrence of given successions of symbols in the source text triggers the inclusion of predefined blocks of machine code. The book describes the exact coding for one computer; from the acquisition of individual characters of the source text from the input unit to the siting of the compiled object program in the working store of the computer, ready to run. Because of this ultimate transformation to absolute code the book also serves as a valuable handbook on single-pass assembly schemes.

The decompilation program included in the appendices completes the circle of source text to machine code to source text. Moreover, it is shown that programs originally written in machine code can be decompiled successfully, and it is claimed that this process can be a powerful aid in "debugging" machine code programs. The fact that any meaningful sequence of machine instructions can be converted mechanically to a meaningful sequence of Nelliac text on at least one machine would seem to call for some studies of the properties of those machine/language pairs for which this relationship holds.

It is unfortunate that whole-hearted recommendation of the content of this book must be tempered by concern at the low quality of the typesetting. Not only are there misprints, and footnotes that have wandered from their proper page, but here and there whole lines would appear to have been omitted by the printer.

H. D. BAECKER.