An Algorithm for the Allocation Problem*

T. E. EASTERFIELD

Industrial Operations Unit, Department of Scientific and Industrial Research

This paper* sets out a procedure for solving allocation problems, on different lines from procedures based on linear programming.

IN "A Combinatorial Algorithm"[1] the author posed and solved the following problem: let us suppose we are given a square $(n \times n)$ array of real numbers $\{a_{ij}\}$ $(i,j = 1, \ldots, n)$, and let it be required to find, among all sums $\sum_{i=1}^{n} a_{ir_i}$ in which r_i is the ith member of a permutation of the numbers $1, \ldots, n$, that subset whose members have the least value. (For brevity, this subset will be called the set of minimum sums.) It will be seen that this is, in fact, the allocation problem. If there are n activities that must be carried out at n locations, and a_{ij} is the cost of carrying out activity i at location j, then the set of minimum sums gives us the set of cheapest ways of allocating the activities to the locations.

(2) Let us call the least elements in each row the *row-minima*. It is obvious that *if* the row-minima are so disposed that we can choose one from each row, no two being in the same column, then such a set is a minimum sum; and, moreover, that in this case only such a set of row-minima can be a minimum sum. Thus, for example, in the array

$$\begin{matrix} 1 & 1 & 3 \\ 4 & 1 & 1 \\ 1 & 5 & 1 \end{matrix}$$

the sums $(a_{11}+a_{22}+a_{33}), (a_{12}+a_{23}+a_{31})$ consist entirely of 1's. Any other sum of elements, no two of which lie in the same row or column, will contain elements greater than 1, but none less, and therefore will not be in our required set.

(3) In general, however, this condition will not be satisfied. It is, however, possible to derive from our original array $\{a_{ij}\}$ another, $\{\hat{a}_{ij}\}$, with the properties:

(i) Σa_{ir_i} belongs to the required set of minimum sums if and only if $\Sigma \hat{a}_{ir}$ belongs to the set of minimum sums of the array $\{\hat{a}_{ij}\}$;

* The basis of this paper is one published by me in 1946, which gave an algorithm for allocation problems. The circumstances of the time did not allow the results to be used then. I am indebted to Dr. Vajda for the suggestion that the main results should be published in a place and under a title where they are more likely to be found by readers to whom they might be of interest. At the same time I have taken the opportunity to fill a gap and make a minor correction to the original paper.

The algorithm does not seem to lead to less laborious computation than the better known methods; on the other hand it seems to me closer in spirit to the essentially combinatorial nature of the problem.—T. E. Easterfield.

(ii) it is possible to choose from among the row-minima of $\{\hat{a}_{ij}\}$ a set of n no two of which lie in the same row or column;

(iii) any row-minimum of $\{\hat{a}_{ij}\}$ lies in at least one minimum sum.

In fact, our derived array $\{\hat{a}_{ij}\}$ will enable us to find all permutations r_i such that Σa_{ir_i} is a minimum sum of the original array.

(4) $\{\hat{a}_{ij}\}$ is formed from $\{a_{ij}\}$ by a series of steps, in each of which a constant quantity is subtracted from every element of one or more columns. Such a step will alter the pattern of row-minima; but since each column contributes one term to any sum, all sums will be altered by the same amount, and the permutations giving rise to minimum sums will be unchanged.

(5) We wish to find a pattern of row-minima such that we can choose n of them, one lying in each row and one in each column. It is, however, easier to work with another condition on the pattern that may be shown to be equivalent to this. Let us define a set of $r(\leqslant n)$ columns as being *adjusted* if each subset of $s(\leqslant r)$ columns in it contains row-minima from *at least* s rows. If an adjusted set of r columns contains row-minima from just r rows, the set will be called *exactly adjusted.*

(6) The paper proves as theorem I: A necessary and sufficient condition for it to be possible to choose one row-minimum from each column of a set of columns in such a way that no two of these lie in the same row is that the set be adjusted. Further, if the set contains no exactly adjusted subset, we may include any one of its row-minima, arbitrarily chosen, in such a choice.

(7) It is then proved that the following algorithm leads to the whole array being adjusted:

Let us represent the ith column of the array by (i), the set of the ith, jth and kth columns by (ijk) (where $i<j<k$), and so on; and let us order all sets of columns as follows: (1) (2) (12) (3) (13) (23) (4) (14) (24) (124) (34) (134) (234) (1234) .... (Formally $(a_1 a_2 \ldots a_r)$ precedes $(b_1 b_2 \ldots b_s)$ if $a_r < b_s$, *or* if $a_r = b_s$ and $(a_1 \ldots a_{r-1})$ precedes $(b_1 \ldots b_{s-1})$. The null set is taken as preceding every other.) We then adjust each set in order.

(8) First, if (1) contains no row-minimum, we subtract from every element of (1) the least number that will make it contain one. Next, treat (2) likewise. Now if (12) contains row-minima in one row only (through their both being in the same row), subtract from every element of both columns the least number required to introduce a row-minimum of another row into (12). Now (12) contains row-minima in two rows, hence it is adjusted. Go on similarly, subtracting from every element of the set reached the least number required to adjust it. The proof in the paper shows that with the given ordering of the sets of columns, the adjustment of any set remains unspoiled by all later steps. Thus we come eventually to a state in which the whole array is adjusted.

(9) Finally, if any subset of columns is exactly adjusted, the subtraction of a very small quantity from every element in the subset will remove from all other columns row-minima in the rows in which the subset has row minima,

but no other row-minima. If this procedure is applied to every exactly adjusted subset left in the final array, we end with a pattern of row-minima such that every row-minimum is in an exactly adjusted set of columns that has no exactly adjusted subset, and hence by theorem I, will appear in at least one minimum sum.

(10) The proof that this algorithm does in fact give the desired answer makes it clear that the ordering of all subsets of columns described in paragraph 7 above is not the only possible one. The essential condition on the ordering is that, if $(a_1 \dots a_r)$ precedes $(b_1 \dots b_s)$, the adjustment of the latter will not spoil that of the former if we have already ensured the adjustment of one of the sets of columns containing every column in each set bar one. (For example, adjusting (145) will not spoil the adjustment of (123) provided that one of (1234) (1235) (1345) and (2345) precedes (145).) Further, any set must be preceded by all its proper subsets. (In the paper, only the latter condition was given.)

(11) For a large array, the number of steps involved is very large. If the job is being done by hand, however, a little practice will allow the worker to take short cuts in choosing sets of columns which will most quickly lead to the final stage.

(12) As an example we give the working on a 6×6 array:

9	22	58	11	19	27
43	78	72	50	63	48
41	28	91	37	45	33
74	42	27	49	39	32
36	11	57	22	25	18
3	56	53	31	17	28

(Note: We have underlined all row-minima in this and the successive derived arrays.)

Here all sets of columns up to (123) are adjusted. Subtract 2 from each element of (4) and obtain

9	22	58	9	19	27
43	78	72	48	63	48
41	28	91	35	45	33
74	42	27	47	39	32
36	11	57	20	25	18
3	56	53	29	17	28

All sets of columns up to (1234) are adjusted. Subtract 10 from each element of (5) and obtain

9	22	58	9	9	27
43	78	72	48	53	48
41	28	91	35	35	33
74	42	27	47	29	32
36	11	57	20	15	18
3	56	53	29	7	28

(15), (25), (125), (35), (135), (235) and (1235) are adjusted; (45) is not. Subtract 2 from each element of each of these columns and obtain

9	22	58	7	7	27
43	78	72	46	51	48
41	28	91	33	33	33
74	42	27	45	27	32
36	11	57	18	13	18
3	56	53	27	5	28

(145), (245), (1245) are adjusted; (345) is not. Subtract 2 from each element of all three columns and obtain

9	22	56	5	5	27
43	78	70	44	49	48
41	28	89	31	31	33
74	42	25	43	25	32
36	11	55	16	11	18
3	56	51	25	3	28

Every set up to (12345) is now adjusted; (6) is not. Subtract 5 from each element of (6) and obtain

9	22	56	5	5	22
43	78	70	44	49	43
41	28	89	31	31	28
74	42	25	43	25	27
36	11	55	16	11	13
3	56	51	25	3	23

The whole set is now adjusted. We may apply the process of paragraph 9 which we may exhibit by subtracting 1 from each element of columns 3 and 4 and obtain

9	22	55	4	5	22
43	78	69	43	49	43
41	28	88	30	31	28
74	42	24	42	25	27
36	11	54	15	11	13
3	56	50	24	3	23

The sets of columns (3), (4) and (1256) are all exactly adjusted but contain no exactly adjusted subset.

It will be found that there are just two minimum sums,

$$a_{21}+a_{52}+a_{43}+a_{14}+a_{65}+a_{36} \quad \text{and} \quad a_{61}+a_{32}+a_{43}+a_{14}+a_{55}+a_{26},$$

which are, in the original,

$$43+11+27+11+17+33=142 \quad \text{and} \quad 3+28+27+11+25+48=142.$$

(13) It will be seen that although each row-minimum left in the final array appears in at least one minimum sum, only a limited number of combinations of row-minima is possible. The following process allows all possible combinations to be found.

(14) We may suppose that the final array has no exactly adjusted subsets, for if not we can treat any exactly adjusted subset independently of the rest of the array.

(15) If we choose any particular row-minimum, we know that it will occur in at least one minimum sum. Moreover, the minimum sums in which it occurs are those of the array formed by suppressing the row and the column in which it lies. This array must obviously be adjusted; but it may now contain an exactly adjusted subset, necessitating the suppression of some row minima by the method of paragraph 9 and the reduction of the array to a set of exactly adjusted sets, none of which contains an exactly adjusted subset.

(16) For a small array, the identification of exactly adjusted subsets can be done easily by inspection. If a mechanical procedure is needed, then for any set of r columns let us define its *excess* as $(s-r)$, where s is the number of rows in which it contains row-minima. An exactly adjusted set of columns has excess zero. Note the excess for each set of columns, at the end of the calculation of paragraphs 8 and 9. (It would probably save no time to note the excess of each column treated during this calculation, since the excess of a set of columns that has been adjusted may change as later sets of columns are dealt

with. We know that it cannot fall below zero; but it may rise, and later it may also fall.)

(17) When a row and a column are removed, as in paragraph 15, any other column will lose a row-minimum if it contained one in the suppressed row, but not otherwise. Thus for any set of columns in the remaining array, the excess will fall by one if *any* column of the set had a row-minimum in the suppressed row, but otherwise be unchanged. It is now easy to see which sets of columns are exactly adjusted, which row-minima must be suppressed, as in paragraph 9, and thus where to start from for the next step.

(18) All minimum sums may therefore be found by the following process:

Break up the array found by the procedure of paragraphs 8 and 9 into exactly adjusted sets of columns without exactly adjusted subsets. Deal with each of these separately.

Take the first row-minimum in column 1 of the first one of these sets, delete the row and column in which it occurs, and break up the remaining array similarly into minimal exactly adjusted sets of columns.

Take the first row-minimum in column 1 of the first of these, and proceed as before.

When this procedure comes to a stop, go back to the last column from which the first row-minimum was taken, and take the next, and proceed as before.

Proceed similarly, going back to the most recent stage at which an alternative choice was possible, until at last all possibilities have been exhausted.

(19) Thus in the example, the columns 3 and 4 are each exactly adjusted. If we delete these and the corresponding rows, and write in only those elements where row-minima occur in the remaining array, we get

$$\begin{array}{cccc} a_{21} & . & . & a_{26} \\ . & a_{32} & . & a_{36} \\ . & a_{52} & a_{55} & . \\ a_{61} & . & a_{65} & . \end{array}$$

(It will be seen that we can work here with the pattern formed by the row-minima only; the numbers they stand for no longer affect the working.)

If we choose a_{21}, the first row and first column must be deleted; a_{36} is now the sole row-minimum in the last column, so a_{32} must be suppressed; a_{52} is the only remaining row-minimum in the first remaining column, so a_{55} must be suppressed. Thus the only minimum sum containing a_{21} is

$$a_{14}+a_{21}+a_{36}+a_{43}+a_{52}+a_{65}.$$

The only point to which we can go back to make a different choice is the beginning: instead, we can choose a_{61}. If we delete column 1 and row 6, the only row-minimum in column 5 is a_{55}; hence a_{52} must be suppressed, and, in

consequence, a_{36}. Thus the only minimum sum containing a_{61} is

$$a_{14}+a_{26}+a_{32}+a_{43}+a_{55}+a_{61}.$$

Since there are no other row-minima in column 1, there are no other minimum sums.

ACKNOWLEDGEMENT

This paper is published with the permission of the Department of Scientific and Industrial Research. Certain passages have been taken verbatim from the original paper; we wish to thank the London Mathematical Society for these.

REFERENCE

[1] T. E. EASTERFIELD, "A Combinatorial Algorithm", *J. London Math. Soc.*, 1946, **21**, 219.