A FEASIBILITY ALGORITHM FOR ONE-WAY SUBSTITUTION IN PROCESS ANALYSIS

Kenneth J. Arrow
Selmer M. Johnson

P-941

Revised September 12, 1957

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA

CLEARINGHOUSE FOR FEDERAL SCIENTIFIC AND TECHNICAL INFORMATION CFSTI
DOCUMENT MANAGEMENT BRANCH 410.11

LIMITATIONS IN REPRODUCTION QUALITY

ACCESSION # AD 605092

☒ 1 WE REGRET THAT LEGIBILITY OF THIS DOCUMENT IS IN PART UNSATISFACTORY. REPRODUCTION HAS BEEN MADE FROM BEST AVAILABLE COPY.

☐ 2 A PORTION OF THE ORIGINAL DOCUMENT CONTAINS FINE DETAIL WHICH MAY MAKE READING OF PHOTOCOPY DIFFICULT.

☐ 3 THE ORIGINAL DOCUMENT CONTAINS COLOR, BUT DISTRIBUTION COPIES ARE AVAILABLE IN BLACK-AND-WHITE REPRODUCTION ONLY.

☐ 4 THE INITIAL DISTRIBUTION COPIES CONTAIN COLOR WHICH WILL BE SHOWN IN BLACK-AND-WHITE WHEN IT IS NECESSARY TO REPRINT.

☐ 5 LIMITED SUPPLY ON HAND: WHEN EXHAUSTED, DOCUMENT WILL BE AVAILABLE IN MICROFICHE ONLY.

☐ 6 LIMITED SUPPLY ON HAND: WHEN EXHAUSTED DOCUMENT WILL NOT BE AVAILABLE.

☐ 7 DOCUMENT IS AVAILABLE IN MICROFICHE ONLY.

☐ 8 DOCUMENT AVAILABLE ON LOAN FROM CFSTI (TT DOCUMENTS ONLY).

☐ 9

PROCESSOR: GL

TSL 107 (3-53)

SUMMARY

Suppose that we have a certain number of machines, not all of the same capability, with which to do a certain number of tasks, not all of the same difficulty. On the assumption that it is better to use a more capable machine for a more difficult task, an algorithm is given for the most efficient assignment of machines to tasks. It is shown that the algorithm solves an equivalent linear programming problem.

A FEASIBILITY ALGORITHM FOR ONE-WAY SUBSTITUTION IN PROCESS ANALYSIS*

1. INTRODUCTION

We suppose, as is done in some simple forms of process analysis (see Refs. 1 and 2), that we have a certain number of machines of m different types with which to do a certain number of tasks of n different types. Each machine is capable of performing tasks of one or more types, and conversely each task can be done by machines of one or more types. We further assume that the machines can be numbered in decreasing order of capability, so that any machine can perform any task that can be performed by a later machine; similarly, it is assumed that the tasks can be numbered in decreasing order of difficulty, so that, if a given task can be performed by a given machine, all later tasks can be performed by that machine. Finally, we assume that, in a certain sense, it is more efficient to do a task by a machine that is as close as possible to being the first machine in the series. That is, if task 1 is more difficult than task 2, then it is better to use a more capable machine to do task 1 (if the machine is capable of performing task 1) and use a less capable machine to do task 2.

Suppose we are given a certain amount of work of each type to do and a certain quantity of machines of each type.

---

*We are indebted to Harry Markowitz for cooperation in conjecturing the algorithm discussed here.

We wish to test whether or not the given task program is feasible.

2. THE ALGORITHM

The following algorithm is proposed: First, use the most capable machine for the most difficult task. If the whole desired quantity of the task can be done by the available quantity of this machine, apply the left-over amount of the most capable machine to the next most difficult task. Continue until the stock of machines of the most capable type is exhausted and then continue with machines of the second most capable type. This process is continued either until all the machine stocks capable of performing a certain task have been exhausted and some of this task remains, in which case the task program is infeasible, or until all the tasks have been assigned without using more of any machine than is available, in which case the program is feasible.

The second statement is obvious; the difficult point is to prove the criterion for infeasibility, since it might seem at first glance that other assignments of machines to tasks than the one considered might show a program to be feasible when the method of assigning the most difficult tasks to the most capable machines would not indicate this. In fact — as we shall see — under the assumptions made, this possibility cannot arise. Moreover, the algorithm will be shown to solve an equivalent linear programming problem.

3. MATHEMATICAL FORMULATION

Let a_{ij} be the amount of task j performed by a unit input of machine i. The ordering of machines and tasks by capability and difficulty, respectively, can be expressed as follows:

(1) if $a_{ij} > 0$ and $i' < i$, then $a_{i'j} > 0$;

(2) if $a_{ij} > 0$ and $j' > j$, then $a_{ij'} > 0$.

We are further assuming that if $i < i'$ and $j < j'$, and if machine i' is capable of handling task j at all, then it is better to devote machine i to task j and use the amount of released stock of machine i' for task j'. That is, the output of task j' for a unit input of machine i is less than the output achievable by devoting the unit of machine i to task j and applying the amount of machine i' that would have been needed to perform the same amount of task j to the performance of task j'. By definition, it requires $a_{ij}/a_{i'j}$ units of machine i' to perform the same amount of task j as one unit of machine i; hence, the assumption can be expressed as follows:

(3) if $i < i'$, $j < j'$, and $a_{i'j} > 0$, then

$a_{ij'} < (a_{ij}/a_{i'j})a_{i'j'}$; that is, $(a_{ij'}/a_{ij}) < (a_{i'j'}/a_{i'j})$,

since by (1) the inequality $a_{i'j} > 0$ implies $a_{ij} > 0$.

Let

M_i = the number of units of the i–th type machine ($i = 1, 2, \ldots, m$),

T_j = the number of units of the j–th type task to be performed ($j = 1, 2, \ldots, n$),

x_{ij} = the number of units of the i–th type machine assigned to the j–th type task,

$\bar{X}_r = (\bar{x}_{ij})_r$ = the set of x_{ij}'s defined according to the algorithm described in the introduction when applied to the first r tasks.

Thus

$$\bar{x}_{11} = \min\left(M_1, \frac{T_1}{a_{11}}\right).$$

If $\bar{x}_{11} = M_1$, then $\bar{x}_{1j} = 0$ for $j > 1$, and

$$\bar{x}_{21} = \min\left(M_2, T_1 - \frac{a_{11}M_1}{a_{21}}\right).$$

If $\bar{x}_{11} = \frac{T_1}{a_{11}}$, then $\bar{x}_{ij} = 0$ for $i > 1$, and

$$\bar{x}_{12} = \min\left(M_1 - \frac{T_1}{a_{11}}, \frac{T_2}{a_{12}}\right),$$

and so on.

4. THEOREM AND PROOF

We shall establish the following result.

Theorem. If any allocation procedure satisfies a given program of machines and tasks, then the algorithm does also,

and in addition it maximizes the amount of work done on the last task type while meeting the other task requirements.

We present a proof by induction.

Let F_r be the set of x_{ij}'s satisfying the conditions

$$x_{ij} \geq 0, \tag{4}$$

$$\sum_{j=1}^{n} x_{ij} \leq M_i \qquad (i = 1, \ldots, m), \tag{5}$$

$$\sum_{i=1}^{m} a_{ij} x_{ij} \geq T_j \qquad (j = 1, \ldots, r). \tag{6}$$

We are seeking to prove that if F_n is nonnull, then it contains $\bar{X}_n$.

Let P_{r+1} be the linear programming problem of maximizing

$$\sum_{i=1}^{m} a_{i,r+1} x_{i,r+1} \tag{7}$$

among the elements of F_r — that is, maximizing the work done on the $(r+1)$-st type task while satisfying the requirements on the tasks of the first r types. We want to prove the following two statements:

(8) if $\bar{X}_r$ belongs to F_r, then $\bar{X}_{r+1}$ solves P_{r+1};

(9) if $\bar{X}_{r+1}$ solves P_{r+1}, then it belongs to F_{r+1}.

The two statements together imply that $\bar{X}_{r+1}$ belongs to F_{r+1}

if $\bar{X}_r$ belongs to F_r. Now $\bar{X}_0$ clearly belongs to F_0; hence induction establishes the result that $\bar{X}_r$ belongs to F_r for all $r \leq n$, and therefore $\bar{X}_n$ belongs to F_n.

The proof of (9) is obvious. Since F_n is nonnull, there is a set of x_{ij}'s belonging to F_r for which (7) takes on a value at least as great as T_{r+1} (since an element of F_n belongs to F_r for all r). From (8), then, (7) must take on at least as great a value when $\bar{X}_{r+1}$ is substituted for the x_{ij}'s. Hence $\bar{X}_{r+1}$ belongs to F_{r+1}.

To prove (8), we assume that $\bar{X}_r$ is in F_r. Construct $\bar{X}_{r+1}$ from $\bar{X}_r$ by diverting any possible excess machine time from the r-th type task to the $(r+1)$-st type task according to the algorithm. It is easy to see that the set $\bar{X}_{r+1}$ forms a basis for the linear programming problem P_{r+1}. By construction there are $m+r$ basic variables > 0 that form a "path" of steps to the right and downward through the (m by $r+1$) matrix of (x_{ij}) in P_{r+1} starting at $\bar{x}_{11}$ and ending at $\bar{x}_{m,r+1}$. (If an M_i and a T_j are exhausted simultaneously, we choose $\bar{x}_{i+1,j} = 0$ for the basis.)

Let p_i and q_j be the dual variables associated with the inequalities (5) and (6), respectively. Then the duality conditions require that

$$(10) \qquad a_{ij}q_j - p_i \leq 0 \qquad (i = 1, \ldots, m;\ j = 1, \ldots, r),$$

$$(11) \qquad a_{i,r+1} - p_i \leq 0 \qquad (i = 1, \ldots, m).$$

Inequalities (10) and (11) can be combined in the form

$$(12) \quad a_{ij}q_j - p_i \leq 0 \quad (i = 1, \ldots, m;\ j = 1, \ldots, r+1), \quad q_{r+1} = 1.$$

We define the q_j's and p_i's by requiring equality for the basic variables in (12); i.e., we require

$$(13) \quad a_{ij}q_j - p_i = 0 \quad \text{for } i, j \text{ corresponding to } \bar{x}_{ij} > 0.$$

We shall show that (12) holds also for every nonbasic variable such as $x_{i_0 j_0}$, say; i.e., we shall show that

$$(14) \qquad \frac{p_{i_0}/q_{i_0}}{a_{i_0, j_0}} \geq 1.$$

Now $x_{i_0 j_0}$ can be expressed in terms of a subset of the basic variables, say $\bar{x}_{i_1 j_0}$, $\bar{x}_{i_1 j_1}$, $\bar{x}_{i_2 j_1}$, $\bar{x}_{i_2 j_2}$, ..., $x_{i_0 j_k}$, where this set combined with $x_{i_0 j_0}$ forms the corners of a closed loop in the (x_{ij}) matrix.

To simplify the notation, let

$$(u, v) = a_{i_u, j_v}.$$

Then from (13) we have

$$(u, u-1) = p_{i_u}/q_{j_{u-1}},$$

$$(u, u) = p_{i_u}/q_{j_u},$$

$$(0, k) = p_{i_0} q_{j_k}.$$

Therefore, we have

$$(u, u-1)/(u, u) = q_{j_u}/q_{j_{u-1}}.$$

Hence, the ratio in (14) can be written as

$$\frac{p_{i_0}/q_{i_0}}{(0, 0)} = \frac{(0, k)}{(0, 0)} \prod_{u=1}^{k} \frac{(u, u-1)}{(u, u)}$$

$$= \frac{(0, k)}{(0, 0)} \prod_{u=1}^{k} \frac{(u, u-1)}{(u, u)} \prod_{u=1}^{k} \frac{(u, 0)}{(u, 0)}$$

$$= \left[\frac{(0, k)}{(k, k)} \frac{(k, 0)}{(0, 0)}\right] \prod_{u=0}^{k-1} \left[\frac{(u+1, u)(u, 0)}{(u, u)(u+1, 0)}\right] \geq 1,$$

since by (3) the value of each heavy bracket group is ≥ 1.

Thus, (12) is verified for every x_{ij}, and $\bar{X}_{r+1}$ is an optimal linear programming solution to P_{r+1}. That is, (8) holds for each r, and the theorem is proved.

REFERENCES

1. Markowitz, H., Process Analysis of the Metal Working Industries, The RAND Corporation, Research Memorandum RM-1085, May 12, 1953.

2. _____, "Process Analysis of the U. S. Technology," Econometrica, Vol. 22 (1954), p. 515 (abstract).