PERIODICA POLYTECHNICA SER. HUM. and SOC. SCI. VOL. 3, NO. 1, PP. 71–80 (1995)

ZERO-ONE ALLOCATION OF SUBSTITUTABLE RESOURCES

Béla PATAKI

Department of Industrial Management and Business Economics
Technical University of Budapest
Phone: (36-1) 204-1111/24-32
Fax: (36-1) 166-5208

Received: July 5, 1994

Abstract

The author found a major limitation of the original zero-one programming method: it can't take into account that some resources are substitutable regarding some particular activities, i. e. the desired result is achievable by using any of them. Substitutability can be multi-level when the higher level resources are broader categories, containing every kind of resources which can be substituted regarding one or more activities. Usually substitutability is not a general feature but refers only to some particular activities. This article:

- defines this kind of substitutability,
- describes a tree-structured model of this special kind of hierarchical substitutability,
- introduces a measure of substitutability, called 'substitutability coefficients'.

The author modified the original mathematical formulas:

- for calculating the constraints of higher level resources in the hierarchy, and
- for determining the feasible solutions (a set of inequalities),

taking the hierarchical substitutability of resources into account.

Keywords: zero-one programming, resource allocation, substitutability.

1. Introduction

Different types of mathematical programming are well known in management science and are widely used in managerial practice for optimizing different sorts of objective functions subject to different sorts of constraints.

'Resource allocation problems are concerned with the allocation of limited resources among competing activities so as to optimize some objective. In certain applications there is a single limited resource, e.g., money, whereas in others, hundreds or thousands of resources must be allocated prudently. (...) Careful allocation of limited resources is, therefore, needed on different levels, starting from strategic, long-term planning down to weekly, or daily, production scheduling.' (KLEIN and LUSS, 1991).

One of these methods is zero-one programming. The author realized a major limitation of the original method when he tried to apply it to real resource allocation problems and developed a new algorithm to eliminate this limitation.

2. Zero-one Programming

The following notation is used:

i = index for resources
j = index for activities
k = index for sets of activities (solutions)
r = number of resources ($i = 1 \dots r$)
a = number of activities ($j = 1 \dots a$)
c_i = amount available of resource i (constraint i)
s_i = substitutability coefficient of resource i
d_{ij} = amount of resource i needed for activity j (demand ij)
w_j = weight, representing relative importance, associated with activity j (objective function coefficient j)
x_{kj} = value of activity j in solution k

The allocation problem in zero-one programming is given by the equations:

$$\text{maximize: } W = \sum_{j=1}^{a} w_j x_{kj}$$

$$\text{subject to: } \sum_{j=1}^{a} d_{ij} x_{kj} \leq c_i \text{ for every } i$$

The x_{kj}-s can assume only the values of zero and one. It means that we either fully implement an activity (a course of action) or we don't implement it at all, there are no other choices, partial implementation is not possible. The available amounts of our resources are limited and we can implement only such combinations of our activities which have total consumptions (the sums of the activities' demands) lower than these limitations.

2.1. Theoretical Solutions

The first step in solving this resource allocation problem is to determine all sets of activities not regarding our constraints (theoretical solutions). Since every activity can assume two values, and we have a activities, the number of theoretical solutions is 2^a. For example, if we have 3 possible activities, the number of theoretical solutions is $2^3 = 8$, as *Fig. 1* shows. 'Zero' means that we don't implement that particular course of action,

$$\begin{array}{c|cccc}
 & 1 & 2 & 3 & \rightarrow j \\
\hline
1 & 1 & 1 & 1 & \\
2 & 1 & 1 & 0 & \\
3 & 1 & 0 & 1 & \\
4 & 1 & 0 & 0 & \\
5 & 0 & 1 & 1 & \\
6 & 0 & 1 & 0 & \\
7 & 0 & 0 & 1 & \\
8 & 0 & 0 & 0 & \\
\downarrow k & & & &
\end{array}$$

Fig. 1. Theoretical solutions with three activities

'one' means that we implement it. For example: solution 5 means that we implement activity 2 and 3 but don't implement activity 1.

2.2. Feasible Solutions

The second step is to screen out every not feasible solution. Our problem in 0/1 programming is that solution 1 (implementing all activities), which would obviously be the most desirable solution, is not feasible because it needs more resources than we have. Solution 8 (doing nothing) is always feasible but a real-life problem solver would not be very happy with it. If we checked the feasibility of every activity one-by-one during data input then it is trivial that solutions 4, 6, and 7 are also feasible. The constraints become the inequalities:

$$\sum_{j=1}^{a} d_{ij} x_{kj} \leq c_i \text{ for every } i.$$

If there is only one c_i resource limit lower than the overall consumption of a particular solution then solution k is not feasible.

2.3. Optimal solutions

The third step is to choose the best feasible solution or solutions. At this point, it is necessary to make a distinction between the two basic versions of 0/1 programming depending on whether the objective function is given on a quantitative or on an ordinal scale.

The above mentioned types of scales of measurement refer to the classical types defined by STEVENS (1946, 1951). This classification has become widely accepted and has been used without any change up to now. '... measurement, in the broadest sense, is defined as the assignment of numerals to objects or events according to rules. The fact that numerals can be assigned under different rules leads to different kinds of scales and different kinds of measurement.' (STEVENS, 1946).

Nominal scale numerals are used only as labels or type numbers, so words, letters, colours, etc. would serve as well. Two types of nominal assignments are sometimes distinguished: one with unit classes of one member each, and another with more than one member in each class. This scale has the so-called 'substitution (or permutation) group' structure because it remains invariant under any $x' = f(x)$ transformation where $f(x)$ means any one-to-one substitution.

Ordinal scale arises from the operation of rank-ordering. This scale has the structure of an 'isotonic (or order-preserving) group' since any $x' = f(x)$ transformation will leave this scale form invariant where $f(x)$ means any monotonic increasing function. Rankings don't express anything about differences or ratios and it is a very frequent source of serious computational errors in everyday life, even in the practice of professionals without sufficient mathematical background. Ordinal numbers are very often added up, subtracted, multiplied, or divided with each other as if they showed anything more than just the relative rank-order of data. These mathematical operations are invalid with rankings and the results of these invalid operations are meaningless. The results would be correct only if the successive intervals on the scale were equal in size but we haven't got any such kind of information when we only have ordinal numerals (rankings). This fact is very important when we distinguish the two main types of 0/1 programming.

On an *interval scale* the zero point is a matter of convention or convenience. This scale has a 'general linear group' structure because it remains invariant under any $x' = ax + b$ transformation where $a \neq 0$. Ratios of interval scale numerals are meaningless because the zero point is arbitrary, but dividing differences of interval scale numerals is valid because in this case constant b, which defines the zero point, disappears during the subtractions.

Ratio scale has a natural zero point. It has the structure of a 'similarity group' since any $x' = ax$ transformation will leave this scale type invariant where $a \neq 0$. All kinds of mathematical operations are applicable to ratio scales. Interval and ratio scales together are often called quantitative scales.

The more conventional and better known version of 0/1 programming is the one with a quantitative objective function (see e.g. DANNENBRING and STARR, 1981). In this case every activity has a quantitative w_j objective function coefficient (e.g. money) which expresses how favourable or important it is for us to implement. It is very easy to find the optimal solution by simply adding up the w_j weights of the activities within every solution k:

$$W_k = \sum_{j=1}^{a} w_j x_{kj}.$$

The optimal solution k is the one with the maximal W_k value. It is possible to have more than one optimal solutions with the same W_k values but it is not very frequent in practice.

Optimizing is a bit more difficult with ordinal scale objective function coefficients (BARTEE, 1971). In this case we know only the preference order of our activities, so the w_j values are just rankings. We don't know the real quantitative relations, and therefore we cannot add up the ranking numbers as discussed above.

If we assign the rankings to our activities and mark the most preferred activity with $A1$ and the second best with $A2$, etc., and indicate preference relations with arrows pointing from the more preferred to the less preferred, then we can draw a preference graph of our solutions as on *Fig. 2/a.* (We could also use a simplified graph, like *Fig. 2/b*, by omitting the redundant arrows showing preference relations which are obvious. For instance, arrow $A1 \rightarrow A3$ is redundant because the preference relations are transitive, so arrows $A1 \rightarrow A2$ and $A2 \rightarrow A3$ show the relation between $A1$ and $A3$ as well.)

The important thing in our example is the preference relation between solution $A1$ and solution $A2+A3$. We cannot choose between them because our objective function is ordinal. We can't decide which solution would be better for us: the most preferred activity alone or the second and third best ones together, because we only know rankings. Solutions $A1$ and $A2+A3$ are not comparable on an ordinal scale. Such solutions are called *alternate optimal solutions.*

What is of particular interest is that the number of non-comparable solutions increases exponentially as the number of activities increases. *Table 1* shows this trend as BARTEE (1971) calculated it. The results show

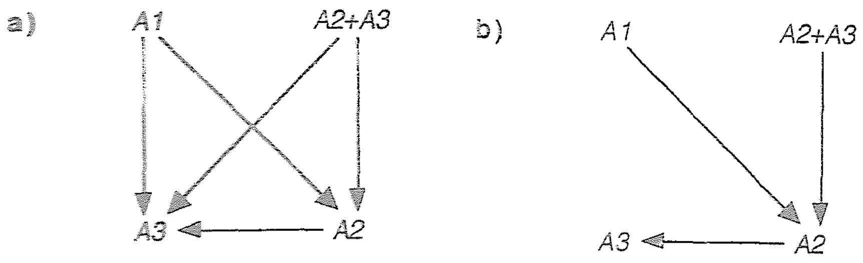

Fig. 2. Preference graph of feasible solutions on ordinal scale

an exponential decrease in the degree of isomorphism between the model and the real problem if someone makes pseudo comparisons using rankings as if they were real numbers on a quantitative (interval or ratio) scale.

Table 1
The problem of non-comparable solutions

Number of activities	Number of solutions	Number of paired comparisons	Not comparable No.	%
2	4	6	0	0
3	8	28	1	3.6
4	16	120	10	8.3
5	32	496	66	13.3
6	64	2.016	364	18.1
7	128	8.128	1.821	22.4

There are various but basically similar algorithms for the computation of this kind of comparison in ordinal 0/1 programming (GERZSENYI, 1980; PATAKI, 1989/a; CSEGÉNY, 1992) but these computer algorithms are beyond the scope of this article.

3. The Problem of Substitutability

During the assessment of consumptions and constraints a special problem may arise: what resource categories should we use? Let us see an example. We have some large and some small lorries for transporting different sorts of goods. Some of the goods can be transported with both types of lorries, but some other sorts (because of their weights or measures) are transportable only with the large ones. What resource categories should we use in this case, '*large lorry capacity*' and '*small lorry capacity*' separately or just '*lorry capacity*' alone? (A huge number of similar examples could be given with skilled workers of different qualifications, different types of manufacturing equipment, different raw materials, etc.)

Some activities could make it necessary to use the finer resource categories. But how to define those activities' demands which can be done with more than just one type of resource? If we define such a demand in one of the finer categories then this amount could be enough in one or more solutions containing this activity for not satisfying one or more resource constraint inequalities. If we defined this demand in another category or in some random combination of more than one category then we would get a different result when we solved the inequalities. If we used the broader category for defining every activity's demands, then we would get more feasible solutions but during the implementation we would find that not all of them are feasible in reality, because we can't use just any type of resource for any activity. So we have to enable our model to use finer and broader resource categories at the same time.

We will use the following definition of substitutability (PATAKI, 1989/a,b):

> *Two or more resources are substitutable regarding a particular activity if the desired result is achievable by using any of them.*

It is important that there can be several usages of the word 'substitutability' with different meanings in management science (see e.g. KLEIN AND LUSS, 1991). In our case it means free choice of different resources for the same purpose.

Substitutability can be multi-level, for instance, if we have open and close lorries, cooled and not cooled closed ones, and different sizes of every type. *Fig. 3* shows such a multi-level substitutability structure. The nodes represent resource categories and the links represent substitutability relations. First-level categories (the leaves of trees) represent the finest differentiation between our resources (the 'real' resources, if you like). Second-level categories contain those first-level ones which are substitutable for at least one activity. In general: level n categories represent those categories on level $n-1$ which are substitutable for at least one activity. In *Fig. 3* resource 1 is not substitutable with any other resource for any activity. Resources 2, 3 and 4 are substitutable for at least one activity, but aren't substitutable with any other ones. Resources 5 and 6 are substitutable for one or more activities, and both are substitutable with resource 7 for one or more other activities, but these three resources are not substitutable with others for any activity.

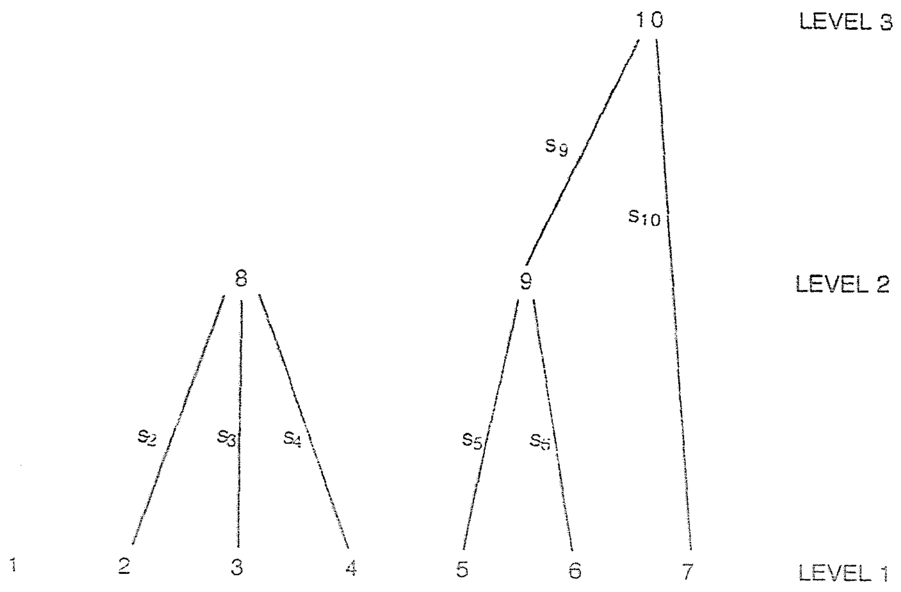

Fig. 3. Hierarchical substitutability of resources

The measure of substitutability can be characterised by *substitutability coefficients* (s_i-s). These coefficients can have different physical dimensions or can be numerals without dimensions. If, for example, we measure the available amounts of our transport capacities in 'lorry-kilometers', for a certain period of time, then it's suitable to use our lorries' load-bearing capacities as substitutability coefficients. If we measure the available amounts in 'lorry-ton-kilometers', then substitutability coefficients have no dimensions. In this case every s_i is equal to 1. But we can also use one of the lorries' load-bearing capacity as a unit of measurement, and in this case the substitutability coefficients are ratio numbers, again without dimension.

The constraints of first-level resources are input data, the higher-level constraints are calculated from these, stepping from bottom upwards, level by level. If we use i' index to sign those resource categories which are $n-1$ level sub-categories of resource category i on level n, then constraint i can be calculated as follows:

$$k_i = \sum_{i'} h_{i'} k_{i'}.$$

We can define any activities' demands with any possible combination of our resource categories regardless of their position in the hierarchical structure. Considering our example in *Fig. 3*: if activity j needs resource 5 and/or resource 6, then d_{5j} and d_{6j} mean those demands which can be met exclusively with resource 5 and with resource 6, respectively, while d_{9j} means those demands which can be met with either resource 5 or resource 6. In practice, demands like d_{5j} and d_{6j} refer to different tasks within the activities which need special expertise, equipment, etc., while d_{9j} refers to such tasks which can be done by workers with different qualifications and skills or by more than just one type of equipment (lorries, computers, manufacturing machines, etc.). All these demands are input data, there are no calculations needed here.

Feasibility calculations of solutions are also done from bottom upwards, level by level. On level 1 there is no change in the method, but on every other level we must take into account that all the demands defined on the lower levels, in narrower resource categories, have to be added to the demand defined in the broader resource category when examining the feasibility of solution k:

$$\sum_{j=1}^{a}(d_{ij} + \sum_{i'} s_{i'} d_{i'j})x_{kj} \leq c_i \quad \text{for every } i.$$

In other words and in another mathematical formula: the amount available of resource i must be reduced by the amount defined on lower levels in narrower resource categories:

$$\sum_{j=1}^{a} d_{ij}x_{kj} \leq c_i - \sum_{j=1}^{a}\left(\sum_{i'} s_{i'} d_{i'j}\right)x_{kj} \quad \text{for every } i.$$

The two wordings and the two formulas are equivalent.

4. Final Remarks

A computer software for 0/1 programming with substitutable resources was at first developed for testing the new algorithm and for demonstration purposes only. (PATAKI, 1989/a). Since then a professional, user-friendly version has been developed in the author's department for general purposes. (CSEGÉNY, 1992). A complex decision support system (DSS) is being developed, containing the substitution algorithm described above. This DSS development project is supported by the National Institute of Technological Development (OMFB).

References

1. BARTEE, E. M. (1971): Problem Solving with Ordinal Measurement. *Management Science,* Vol. 17, No. 10, pp. B622–B633.
2. CSEGÉNY, Z. (1992): A 0/1 programozás számítógépes programja és menedzsment alkalmazási lehetőségei (Computer Software and Possible Management Applications of 0/1 Programming). MSc dissertation, Technical University of Budapest, 1992 (In Hungarian.)
3. DANNENBRING, D. G. – STARR, M. K. (1981): Management Science – an Introduction. McGraw–Hill, pp. 395–401, 411–418.
4. GERZSENYI, J. (1980): Az ordinális programozás alkalmazástechnikája és számítógépes programja 1–2 (Application and Computer Software of Ordinal Programming). *Információ – Elektronika,* Vol. XV, No. 3, pp. 165–169, No. 5, pp. 294–298. (In Hungarian.)
5. KLEIN, R. S. – LUSS, H. (1991): Minimax Resource Allocation with Tree Structured Substitutable Resources. *Operations Research,* Vol. 39, No. 2, pp. 285–295.
6. PATAKI, B. (1989/a): Az ordinális programozás továbbfejlesztése és számítógépes programja (Some Developments of Ordinal Programming and it's Computer Software). *Doctoral thesis,* Technical University of Budapest, 1989. (In Hungarian.)
7. PATAKI, B. (1989/b): Korszerűsített ordinális programozás (Advanced Ordinal Programming). *Ipar–Gazdaság,* Vol. XLI, No. 10, pp. 29–32. (In Hungarian.)
8. STEVENS, S. S. (1946): On the Theory of Scales of Measurement. *Science,* Vol. 103, No. 2684, pp. 677–680.
9. STEVENS, S. S. (1951): Mathematics, Measurement, and Psychophysics. In: Handbook of Experimental Psychology, John Wiley & Sons, pp. 1–49.