63-3-3

10

PACIFIC MISSILE RANGE

POINT MUGU, CALIFORNIA

Technical Memorandum No. PMR-TM-63-6

A MATHEMATICAL FORMULATION AND SOLUTION OF A MULTIPLE-RESOURCE SCHEDULING PROBLEM

By
R. V. Skarda, Jr.
Operations Research Group

1 May 1963

Approved by:
L. A. LEAKE
Head, Operations Research Group

Dr. G. W. BRAUN
Chief Scientist

HEADQUARTERS PACIFIC MISSILE RANGE
POINT MUGU CALIFORNIA

TABLE OF CONTENTS

SUMMARY

An algorithm is proposed for determining "optimal" ways of scheduling multiple operations, each of which must be serviced by a number of resources simultaneously. The objective is to allocate the resources on a time-scale such that all the desired operations may be completed in the minimum time possible. By means of a selective searching procedure, the algorithm in effect considers all possible allocations but in much less time than direct enumeration would entail. The result is an "optimal" schedule or time sequence for operations. The basic inputs to the algorithm are each operation's time requirements on each type of resource and a statement of the resources which the servicing system has available for allocation. The procedure is deterministic in the sense that the operational time requirements, which are in truth probabilistic, are assumed to be known exactly.

The algorithm was developed as a possible means for scheduling operations on the Pacific Missile Range. In that context, the many operations and even greater number of resource types which must be considered necessitate the use of a large-scale digital computer to execute the algorithm.

INTRODUCTION

This memorandum describes a formulation of an operation scheduling problem, involving service by multiple resources, and provides an algorithm for finding "optimal" schedules. The work described herein is part of a broad study to develop effective procedures for handling the increasingly complex task of scheduling operations on the Pacific Missile Range. Two other approaches[1, 2] have been made; one[1] considers a different objective function, and the other[2] partly motivated the approach of this paper. The second is limited, however, in that it treats only a *sequencing* problem and requires as an input, sets of feasible operations (i.e., collections of operations which, because of the multiple resources, may be executed simultaneously). The algorithm given here generates optimal feasible subsets of the set of all operations and in addition may be generalized to subsume the *sequencing* problem and hence provide optimal *schedules.*

INITIAL PROBLEM FORMULATION

Definitions

A *resource* is any commodity used by the range to support certain missile testing operations. This term includes portions of air and water space, segments of the radio-frequency spectrum, and various instruments, such as radar, real-time computers, and optical gear. A resource can be measured in terms of discrete units (as instruments are) or in arbitrary fractions of units (as segments of air space can be). Although time itself satisfies this definition of a resource, it will not be treated as such in the following.

Conversely, an *operation* is considered abstractly as a requested assortment of resources of various descriptions and quantities.

A *resource type* is a collection of resources, all fitting some particular description (kind of commodity, location, etc.), which can be used interchangeably on certain missile tests. One resource type may differ from another in containing a different kind of commodity. Or, certain resource types which contain the same kind of commodity may differ because of their locations. For example, four resource types might be

1. Radar on San Nicolas Island
2. Radar at Point Mugu
3. Radar either at Point Mugu or on San Nicolas Island
4. Electronic computer

It is only necessary to consider resource types which contain commodities of limited supply and which can thus affect scheduling considerations.

For the following definitions, k represents the number of resource types under consideration.

The *support system* is the totality of the resource types available to the range. It is represented mathematically by a k-dimensional *system vector*, where the dimensions correspond to different resource types (taken in any preassigned order) and where the coefficient for each

[1]Pacific Missile Range. Generalized Range Scheduling Problem, by R. J. Frisbie. Point Mugu, Calif., PMR, 20 Oct 1961. (PMR Technical Memorandum No. PMR-TM-61-18) UNCLASSIFIED.

[2]Pacific Missile Range. An Approach to the Sequencing of Range Operations, by C. E. Wisler. Point Mugu, Calif., PMR, 29 Apr 1963. (PMR Technical Memorandum No. PMR-TM-63-4) UNCLASSIFIED.

dimension represents the number of units available for scheduling of that particular resource type.[3] (A unit of a resource might be a single instrument or a communication channel.)

For example, the system vector, starting with the resource types in the example above, and in the same order, would have the dimensions

(3, 4, 7, 1, . . .),

if there were three radar installations on San Nicolas Island and four radar installations at Point Mugu, with one electronic computer. The following dimensions represent other resource types.[4]

Each operation is characterized by a similar *k*-dimensional *operation vector* representing the resource types in the same order that the system vector does, where each coefficient represents that operation's minimum requirement for the corresponding resource type. For example, an operation simply requiring one radar either on San Nicolas Island or at Point Mugu would have the operation vector

(0, 0, 1, 0, . . .).

On the other hand, an operation requiring one radar on San Nicolas Island and two radars at Point Mugu would have the operation vector

(1, 2, 3, 0, . . .)

since the requirements imply that three units are also required of the third resource type (radars that are either on San Nicolas Island or at Point Mugu).

In treating resources of varying degrees of specificness in this manner, and in designating certain resource types to represent likely resource requests, it is possible for the algorithm to control many possibly complicated conflicts of resources while representing the resource requirements for an operation with an abstract vector.

The sum of two k-dimensional vectors is defined in the traditional mathematical sense; i.e., a *k*-dimensional vector, each component of which is the sum of the corresponding components in the original two vectors. The sum of a finite number of *k*-dimensional vectors is defined similarly; for example,

$$\begin{aligned} & (0, 1, 2, 3, \ldots) \\ \text{plus } & (1, 0, 0, 0, \ldots) \\ \text{plus } & (1, 0, 1, 0, \ldots) \\ \text{equals } & (2, 1, 3, 3, \ldots). \end{aligned}$$

The relation $\leq$ is said to hold between two vectors if and only if it holds for all their respective components:

$$\begin{aligned} \text{for example,} \quad & (0, 1, 2, 3, \ldots) \leq (1, 2, 2, 3, \ldots) \\ & (0, 6, 8, 1, \ldots) \leq (0, 7, 8, 1, \ldots) \end{aligned}$$

[3] It is an underlying assumption that these coefficients are all non-negative.

[4] By altering the system vector to correspond to possible changes in the resource strength of the range, it is possible for the scheduling procedure to adjust for day-to-day breakdowns in equipment or to schedule experimentally with hypothetical additions of equipment as the range grows.

but it is not true that

(0, 6, 0, 0, . . .) $\leq$ (5, 5, 5, 5, . . .).

Given any two arbitrary vectors, the relation $\leq$ does not necessarily hold in either direction.

The term *subset* will refer to a subset of the total set of operations under consideration. For each subset, its *subset vector* is the sum of the vectors of the individual operations belonging to that particular subset, each operation counting no more than once for each subset. The *index* of a subset is the number of different operations it contains. For examples of these definitions, see tables 1, 2 and 3.

A subset is called *feasible* if and only if its (subset) vector $\leq$ the system vector. Feasibility is equivalent to the possibility for the operations in the subset to be scheduled simultaneously without conflicts in resource requirements.

Two subsets are called *disjoint* if they have no operation in common. Similarly, subsets in a collection are disjoint if no two of the subsets have an operation in common. For example, the following subsets are disjoint:

$$\begin{bmatrix}\text{Operation A}\\ \text{Operation C}\\ \text{Operation F}\\ \text{Operation H}\end{bmatrix} \quad \begin{bmatrix}\text{Operation B}\\ \text{Operation E}\end{bmatrix} \quad \begin{bmatrix}\text{Operation D}\\ \text{Operation G}\\ \text{Operation J}\end{bmatrix}$$

Table 1. Initial Data for the Example, Operation and System Vectors

Operation Vectors								
(0,	0,	1,	3,	0,	60,	0,	0.0,	1)
(0,	3,	2,	10,	0,	70,	1,	2.0,	0)
(0,	1,	4,	0,	15,	130,	0,	2.6,	1)
(1,	1,	3,	0,	16,	140,	0,	1.0,	2)
(1,	0,	6,	12,	0,	125,	0,	0.0,	0)
(0,	4,	5,	0,	12,	190,	0,	0.0,	0)
(0,	0,	0,	10,	8,	0,	1,	0.0,	2)
(0,	5,	2,	0,	9,	30,	0,	0.0,	0)
(0,	0,	2,	3,	0,	40,	1,	0.5,	0)
(0,	1,	4,	0,	6,	0,	0,	0.8,	1)
(1,	0,	1,	3,	0,	80,	0,	1.4,	0)
(0,	2,	0,	4,	5,	0,	1,	1.5,	0)
(0,	4,	1,	2,	26,	15,	0,	2.6,	1)
System Vector								
(1,	6,	10,	15,	25,	220,	1,	3.0,	3)

Table 2. Subsets of Index 1

	Feasible Subsets of Index 1
Operation 1	(0, 0, 1, 3, 0, 60, 0, 0.0, 1)
Operation 2	(0, 3, 2, 10, 0, 70, 1, 2.0, 0)
Operation 3	(0, 1, 4, 0, 15, 130, 0, 2.6, 1)
Operation 4	(1, 1, 3, 0, 16, 140, 0, 1.0, 2)
Operation 5	(1, 0, 6, 12, 0, 125, 0, 0.0, 0)
Operation 6	(0, 4, 5, 0, 12, 190, 0, 0.0, 0)
Operation 7	(0, 0, 0, 10, 8, 0, 1, 0.0, 2)
Operation 8	(0, 5, 2, 0, 9, 30, 0, 0.0, 0)
Operation 9	(0, 0, 2, 3, 0, 40, 1, 0.5, 0)
Operation 10	(0, 1, 4, 0, 6, 0, 0, 0.8, 1)
Operation 11	(1, 0, 1, 3, 0, 80, 0, 1.4, 0)
Operation 12	(0, 2, 0, 4, 5, 0, 1, 1.5, 0)
Subset of Index 1 Tested and Discarded	
	(0, 4, 1, 2, 26, 15, 0, 2.6, 1)

Assumptions

The following assumptions are made principally to simplify the description of the algorithm:

1. The duration of each operation is one unit of time (e.g., 15 minutes or 1 hour).
2. Each operation can be scheduled for any time.
3. Each unit of a resource is used for no more than one operation at any time.
4. The desired objective for this problem is to schedule *all* of the given operations into as few units of time as possible, with some (feasible) subsets of operations scheduled simultaneously. This is tantamount to dividing the total set of operations into as few (disjoint) feasible subsets as possible.
5. The scheduling process is to be finished before the first operation is executed.
6. The system vector is constant throughout the time to be scheduled.

In the generalized problem formulation, assumptions 1, 2, 3, and 6 are superseded by less restrictive assumptions.

This algorithm does not concern itself with the relative economic or strategic values of these operations.

THE ALGORITHM

Part I: Feasible Subsets

This part of the algorithm determines all of the feasible subsets and lists them according to index.[5]

[5]The division of the algorithm into two parts is justified by the tendency for part I to contain most of the arithmetic operations and for part II to contain most of the logical operations. Also, part I is the extent to which the algorithm considers the system and operation vectors.

Table 3. Subsets of Index 2

Feasible Subsets of Index 2										Subsets of Index 2 Tested and Discarded									
[1, 2]	(0,	3,	3,	13,	0,	130,	1,	2.0,	1)	[1, 6]	(0,	4,	6,	13,	12,	250,	0,	0.0,	1)
[1, 3]	(0,	1,	5,	3,	15,	190,	0,	2.6,	2)	[2, 3]	(0,	4,	6,	10,	15,	200,	1,	4.6,	1)
[1, 4]	(1,	1,	4,	3,	16,	200,	0,	1.0,	3)	[2, 5]	(1,	3,	8,	22,	0,	195,	1,	2.0,	0)
[1, 5]	(1,	0,	7,	15,	0,	185,	0,	0.0,	1)	[2, 6]	(0,	7,	5,	10,	12,	260,	1,	2.0,	0)
[1, 7]	(0,	0,	1,	13,	8,	60,	1,	0.0,	3)	[2, 7]	(0,	3,	2,	20,	8,	70,	2,	2.0,	2)
[1, 8]	(0,	5,	3,	3,	9,	90,	0,	0.0,	1)	[2, 8]	(0,	8,	4,	10,	9,	100,	1,	2.0,	0)
[1, 9]	(0,	0,	3,	6,	0,	100,	1,	0.5,	1)	[2, 9]	(0,	3,	4,	13,	0,	110,	2,	2.5,	0)
[1, 10]	(0,	1,	5,	3,	6,	60,	0,	0.8,	2)	[2, 11]	(1,	3,	3,	13,	0,	150,	1,	3.4,	0)
[1, 11]	(1,	0,	2,	6,	0,	140,	0,	1.4,	1)	[2, 12]	(0,	5,	2,	14,	5,	70,	2,	3.5,	0)
[1, 12]	(0,	2,	1,	7,	5,	60,	1,	1.5,	1)	[3, 4]	(1,	2,	7,	0,	31,	270,	0,	3.6,	3)
[2, 4]	(1,	4,	5,	10,	16,	210,	1,	3.0,	2)	[3, 5]	(1,	1,	10,	12,	15,	255,	0,	2.6,	1)
[2, 10]	(0,	4,	6,	10,	6,	70,	1,	2.8,	1)	[3, 6]	(0,	5,	9,	0,	27,	320,	0,	2.6,	1)
[3, 7]	(0,	1,	4,	10,	23,	130,	1,	2.6,	3)	[3, 9]	(0,	1,	6,	3,	15,	170,	1,	3.1,	1)
[3, 8]	(0,	6,	6,	0,	24,	160,	0,	2.6,	1)	[3, 10]	(0,	2,	8,	0,	21,	130,	0,	3.4,	2)
[4, 8]	(1,	6,	5,	0,	25,	170,	0,	1.0,	2)	[3, 11]	(1,	1,	5,	3,	15,	210,	0,	4.0,	1)
[4, 9]	(1,	1,	5,	3,	16,	180,	1,	1.5,	2)	[3, 12]	(0,	3,	4,	4,	20,	130,	1,	4.1,	1)
[4, 10]	(1,	2,	7,	0,	22,	140,	0,	1.8,	3)	[4, 5]	[illegible]	1,	9,	12,	16,	265,	0,	1.0,	2)
[4, 12]	(1,	3,	3,	4,	21,	140,	1,	2.5,	2)	[4, 6]	(1,	5,	8,	0,	28,	330,	0,	1.0,	2)
[5, 8]	(1,	5,	8,	12,	9,	155,	0,	0.0,	0)	[4, 7]	(1,	1,	3,	10,	24,	140,	1,	1.0,	4)
[5, 9]	(1,	0,	8,	15,	0,	165,	0,	0.5,	0)	[4, 11]	(2,	1,	4,	3,	16,	220,	0,	2.4,	2)
[5, 10]	(1,	1,	10,	12,	6,	125,	0,	0.8,	1)	[5, 6]	(1,	4,	11,	12,	12,	315,	0,	0.0,	0)
[6, 7]	(0,	4,	5,	10,	20,	190,	1,	0.0,	2)	[5, 7]	(1,	0,	6,	22,	8,	125,	1,	0.0,	2)
[6, 10]	(0,	5,	9,	0,	18,	190,	0,	0.8,	1)	[5, 11]	(2,	0,	7,	15,	0,	205,	0,	1.4,	0)
[6, 12]	(0,	6,	5,	4,	17,	190,	1,	1.5,	0)	[5, 12]	(1,	2,	6,	16,	5,	125,	1,	1.5,	0)
[7, 8]	(0,	5,	2,	10,	17,	30,	1,	0.0,	2)	[6, 8]	(0,	9,	7,	0,	21,	220,	0,	0.0,	0)
[7, 10]	(0,	1,	4,	10,	14,	0,	1,	0.8,	3)	[6, 9]	(0,	4,	7,	3,	12,	230,	1,	0.5,	0)
[7, 11]	(1,	0,	1,	13,	8,	80,	1,	1.4,	2)	[6, 11]	(1,	4,	6,	3,	12,	270,	0,	1.4,	0)
[8, 9]	(0,	5,	4,	3,	9,	70,	1,	0.5,	0)	[7, 9]	(0,	0,	2,	13,	8,	40,	2,	0.5,	2)
[8, 10]	(0,	6,	6,	0,	15,	30,	0,	0.8,	1)	[7, 12]	(0,	2,	0,	14,	13,	0,	2,	1.5,	2)
[8, 11]	(1,	5,	3,	3,	9,	110,	0,	1.4,	0)	[8, 12]	(0,	7,	2,	4,	14,	30,	1,	1.5,	0)
[9, 10]	(0,	1,	6,	3,	6,	40,	1,	1.3,	1)	[9, 12]	(0,	2,	2,	7,	5,	40,	2,	2.0,	0)
[9, 11]	(1,	0,	3,	6,	0,	120,	1,	1.9,	0)										
[10, 11]	(1,	1,	5,	3,	6,	80,	0,	2.2,	1)										
[10, 12]	(0,	3,	4,	4,	11,	0,	1,	2.3,	1)										
[11, 12]	(1,	2,	1,	7,	5,	80,	1,	2.9,	0)										

The main principle of this part of the algorithm is that *a subset can be feasible only if all the subsets of it are feasible.* For example, the subset consisting of operations A, B, and C (which is denoted [A, B, C] for brevity), might be feasible only if the subsets [A], [B], [C], [A, B], [A, C], and [B, C] are all feasible. This is similar in obviousness and spirit to the "Principle of Optimality" on which Richard Bellman bases the art of dynamic programming.[6] However, since the customary functional notation of dynamic programming lends itself awkwardly to this problem, it is not used here.

[6]Bellman, Richard, Dynamic Programming, Princeton, Princeton U. Press, 1957, p. 83.

Procedure

(To illustrate the following steps, an example is carried out in various tables throughout the description of the algorithm. The raw data--the operation and system vectors--appear in table 1.)

First, all of the feasible subsets of index 1 are selected by comparing the operation vectors individually with the system vector, and by discarding all operation vectors which are not $\leq$ the system vector (figure 1). Each discarded operation vector represents an operation which cannot possibly be scheduled on the given system and which is discarded.[7] The feasible operations are numbered in any specific sequence 1, . . . N, where N is the number of different operations. The order of each operation is defined as its number in this sequence. (The order is not significant to the solution of this part of the algorithm. It is used only to eliminate redundancy in the following.)

Then all of the feasible subsets of index 1 (individually schedulable operations) are listed in order with their corresponding (operation) vectors. (In table 2, one operation is discarded because of an overflow in the fifth resource type.)

To find all of the feasible subsets of index 2, each feasible subset of index 1--starting with operations 1, 2, and proceeding up to N - 1--is added vectorwise to each feasible subset of index 1 and of higher order, and is compared with the system vector for compatibility.

In the example, the following calculation occurs:

The operation vector for operation 1 is (0, 0, 1, 3, 0, 60, 0, 0.0, 1), and for operation 2 is (0, 3, 2, 10, 0, 70, 1, 2.0, 0). The (combined) vector for subset [1, 2] is then (0, 3, 3, 13, 0, 130, 1, 2.0, 1) which $\leq$ the system vector (1, 6, 10, 15, 25, 220, 1, 3.0, 3), and so the subset [1, 2] is feasible. The process is repeated for [1, 3], . . . , [1, 4], [2, 3], . . . , and [11, 12] so that each pair of operations is tested for feasibility once and only once. The pairs that are not feasible are discarded and the feasible sets of index 2 are listed with their subset vectors in order of the lower ordered operation of each subset and then in order of the remaining operation. (See table 3.)

The subsets of a given index are listed most easily in order of lowness of the lowest ordered operation of each subset, then the lowness of the second lowest operation, etc. (Note the order in table 4.) This ordering is automatic if each list is read from the "top" and if each subset is added at the "bottom" of its list when it is found feasible.

The procedure for finding feasible subsets becomes more complicated for higher indices. This complication, which reduces very effectively the length of time required for this process, is based on the main principle, stated above, which requires in particular that a subset S of index H might be feasible, and hence should be tested for feasibility, only if its subsets of index H-1 are themselves feasible. These subsets are designated as follows:

S_1 = S without its highest ordered operation

S_2 = S without its second-highest ordered operation,

[7] Such an operation indicates an inconsistency in the data which may require further human judgment and editing before the algorithm is resumed or restarted. In this case, a special printout from the computer--and possibly a halt to the program--might be appropriate.

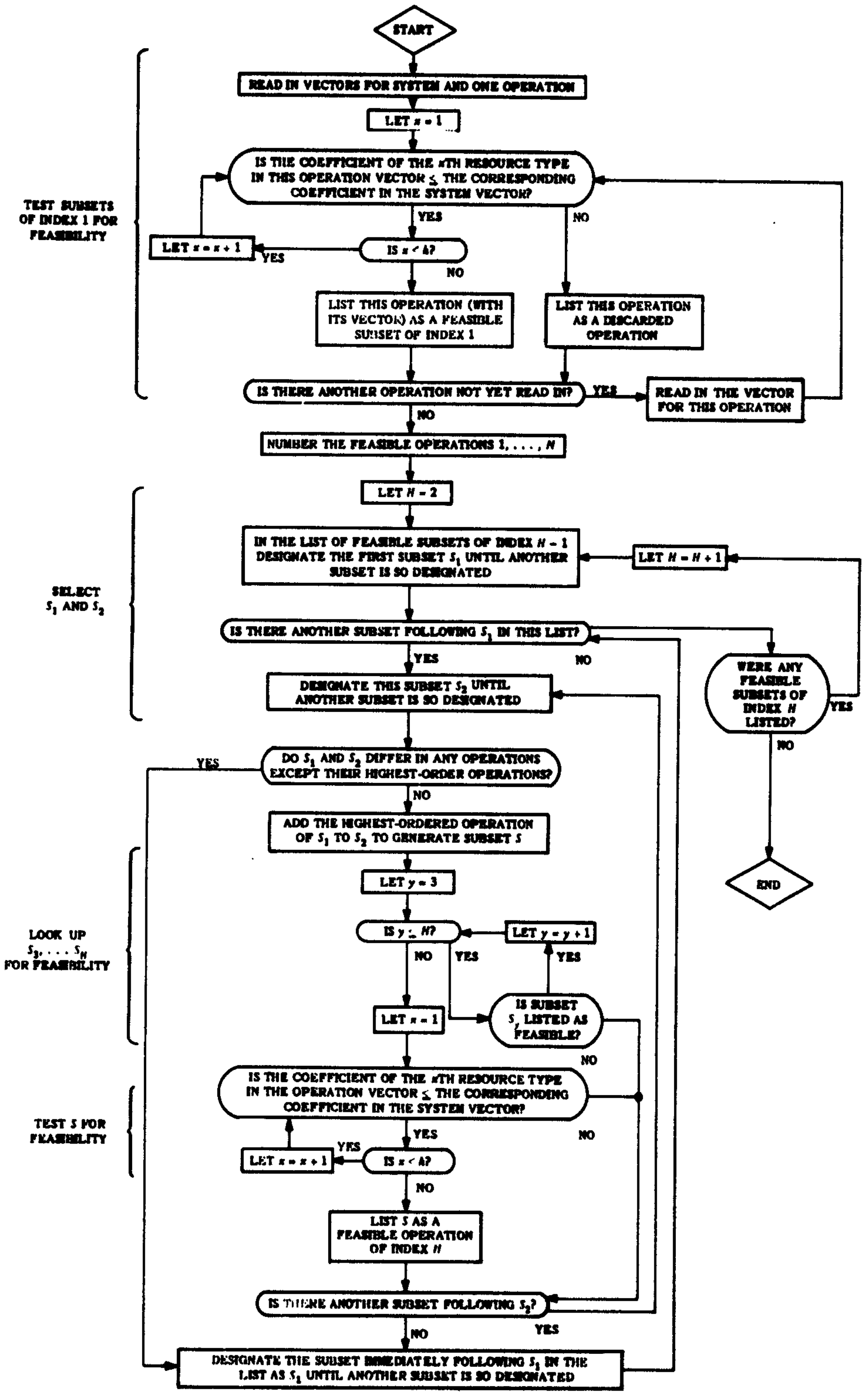
START
READ IN VECTORS FOR SYSTEM AND ONE OPERATION
LET x = 1
IS THE COEFFICIENT OF THE xTH RESOURCE TYPE IN THIS OPERATION VECTOR ≤ THE CORRESPONDING COEFFICIENT IN THE SYSTEM VECTOR?
YES
NO
LET x = x + 1
YES
IS x < k?
NO
LIST THIS OPERATION (WITH ITS VECTOR) AS A FEASIBLE SUBSET OF INDEX 1
LIST THIS OPERATION AS A DISCARDED OPERATION
IS THERE ANOTHER OPERATION NOT YET READ IN?
YES
READ IN THE VECTOR FOR THIS OPERATION
NO
TEST SUBSETS OF INDEX 1 FOR FEASIBILITY
NUMBER THE FEASIBLE OPERATIONS 1, . . . , N
LET H = 2
IN THE LIST OF FEASIBLE SUBSETS OF INDEX H - 1 DESIGNATE THE FIRST SUBSET S1 UNTIL ANOTHER SUBSET IS SO DESIGNATED
LET H = H + 1
SELECT S1 AND S2
IS THERE ANOTHER SUBSET FOLLOWING S1 IN THIS LIST?
YES
NO
DESIGNATE THIS SUBSET S2 UNTIL ANOTHER SUBSET IS SO DESIGNATED
WERE ANY FEASIBLE SUBSETS OF INDEX H LISTED?
YES
NO
END
YES
DO S1 AND S2 DIFFER IN ANY OPERATIONS EXCEPT THEIR HIGHEST-ORDER OPERATIONS?
NO
ADD THE HIGHEST-ORDERED OPERATION OF S1 TO S2 TO GENERATE SUBSET S
LET y = 3
LOOK UP S3, . . . SH FOR FEASIBILITY
IS y ≤ H?
LET y = y + 1
NO
YES
YES
LET x = 1
IS SUBSET Sy LISTED AS FEASIBLE?
NO
IS THE COEFFICIENT OF THE xTH RESOURCE TYPE IN THE OPERATION VECTOR ≤ THE CORRESPONDING COEFFICIENT IN THE SYSTEM VECTOR?
TEST S FOR FEASIBILITY
YES
NO
YES
LET x = x + 1
IS x < k?
NO
LIST S AS A FEASIBLE OPERATION OF INDEX H
IS THERE ANOTHER SUBSET FOLLOWING S2?
NO
YES
DESIGNATE THE SUBSET IMMEDIATELY FOLLOWING S1 IN THE LIST AS S1 UNTIL ANOTHER SUBSET IS SO DESIGNATED

Figure 1. Flow Chart for Part I of the Algorithm.

Table 4. Subsets of Indexes 3, 4, and 5

Feasible Subsets of Index 3		Subsets of Index 3 Tested and Discarded	
[1, 2, 10]	(0, 4, 7, 13, 6, 130, 1, 2.8, 2)	[1, 2, 4]	(1, 4, 6, 13, 16, 270, 1, 3.0, 3)
[1, 3, 8]	(0, 6, 7, 3, 24, 220, 0, 2.6, 2)	[1, 3, 7]	(0, 1, 5, 13, 23, 190, 1, 2.6, 4)
[1, 4, 12]	(1, 3, 4, 7, 21, 200, 1, 2.5, 3)	[1, 4, 8]	(1, 6, 6, 3, 25, 230, 0, 1.0, 3)
[1, 5, 8]	(1, 5, 9, 15, 9, 215, 0, 0.0, 1)	[1, 4, 9]	(1, 1, 6, 6, 16, 240, 1, 1.5, 3)
[1, 7, 8]	(0, 5, 3, 13, 17, 90, 1, 0.0, 3)	[1, 4, 10]	(0, 2, 8, 3, 22, 200, 0, 1.8, 4)
[1, 8, 9]	(0, 5, 5, 6, 9, 130, 1, 0.5, 1)	[1, 5, 9]	(1, 0, 9, 18, 0, 225, 1, 0.5, 1)
[1, 8, 10]	(0, 6, 7, 3, 15, 90, 0, 0.8, 2)	[1, 5, 10]	(1, 1, 11, 15, 6, 185, 0, 0.8, 2)
[1, 8, 11]	(1, 5, 4, 6, 9, 170, 0, 1.4, 1)	[1, 7, 10]	(0, 1, 5, 13, 14, 60, 1, 0.8, 4)
[1, 9, 10]	(0, 1, 7, 6, 6, 100, 1, 1.3, 2)	[1, 7, 11]	(1, 0, 2, 16, 8, 140, 1, 1.4, 3)
[1, 9, 11]	(1, 0, 4, 9, 0, 180, 1, 1.9, 1)	[2, 4, 10]	(1, 5, 9, 10, 22, 210, 1, 3.8, 8)
[1, 10, 11]	(1, 1, 6, 6, 6, 140, 0, 2.2, 2)	[3, 7, 8]	(0, 6, 6, 10, 32, 160, 1, 2.6, 3)
[1, 10, 12]	(0, 3, 5, 7, 11, 60, 1, 2.3, 2)	[4, 8, 10]	(1, 7, 9, 0, 31, 170, 0, 1.8, 3)
[1, 11, 12]	(1, 2, 2, 10, 5, 140, 1, 2.9, 1)	[4, 10, 12]	(1, 4, 7, 4, 27, 140, 1, 3.3, 3)
[4, 8, 9]	(1, 6, 7, 3, 25, 210, 1, 1.5, 2)	[5, 8, 10]	(1, 6, 12, 12, 15, 155, 0, 0.8, 1)
[4, 9, 10]	(1, 2, 9, 3, 22, 180, 1, 2.3, 3)	[5, 9, 10]	(1, 1, 12, 15, 6, 165, 1, 1.3, 1)
[5, 8, 9]	(1, 5, 10, 15, 9, 195, 1, 0.5, 0)	[6, 7, 10]	(0, 5, 9, 10, 26, 190, 1, 0.8, 3)
[7, 8, 10]	(0, 6, 6, 10, 23, 30, 1, 0.8, 3)	[6, 10, 12]	(0, 7, 9, 4, 23, 190, 1, 2.3, 1)
[7, 8, 11]	(1, 5, 3, 13, 17, 110, 1, 1.4, 2)	[10, 11, 12]	(1, 3, 5, 7, 11, 80, 1, 3.7, 1)
[7, 10, 11]	(1, 1, 5, 13, 14, 80, 1, 2.2, 3)		
[8, 9, 10]	(0, 6, 8, 3, 15, 70, 1, 1.3, 1)		
[8, 9, 11]	(1, 5, 5, 6, 9, 150, 1, 1.9, 0)		
[8, 10, 11]	(1, 6, 7, 3, 15, 110, 0, 2.2, 1)		
[9, 10, 11]	(1, 1, 7, 6, 6, 120, 1, 2.7, 1)		
Feasible Subsets of Index 4		**No Subsets Tested of Index 4 Were Discarded**	
[1, 8, 9, 10]	(0, 6, 9, 6, 15, 130, 1, 1.3, 2)		
[1, 8, 9, 11]	(1, 5, 6, 9, 9, 210, 1, 1.9, 1)		
[1, 8, 10, 11]	(1, 6, 8, 6, 15, 170, 0, 2.2, 2)		
[1, 9, 10, 11]	(1, 1, 8, 9, 6, 180, 1, 2.7, 2)		
[7, 8, 10, 11]	(1, 6, 7, 13, 23, 110, 1, 2.2, 3)		
[8, 9, 10, 11]	(1, 6, 9, 6, 15, 150, 1, 2.7, 1)		
Feasible Subset of Index 5		**No Subsets Tested of Index 5 Were Discarded**	
[1, 8, 9, 10, 11]	(1, 6, 10, 9, 15, 210, 1, 2.7, 2)		

S_3 = S without its third-highest ordered operation,

. . . .

S_H = S without its lowest ordered operation.

For example, if S = [1, 8, 9, 10, 11], then S_1 = [1, 8, 9, 10],

S_2 = [1, 8, 9, 11], S_3 = [1, 8, 10, 11], S_4 = [1, 9, 10, 11]

and S_5 = [8, 9, 10, 11].

Before any detailed description of this procedure, some comments about these subsets are in order.

1. Because of the system described above of ordering subsets of a given index, S_1, S_2, S_3, . . . , and S_H appear in the list of feasible subsets of index H-1 in the same order, although not necessarily consecutively.
2. S_1 and S_2 differ only in their highest ordered operations.

3. S actually consists of subset S_1 plus the highest ordered operation of subset S_2.
4. The definitions imply that any two subsets of index H-1 which satisfy 2 determine a unique S of index H in the manner of 3. These subsets are S_1 and S_2.

Each subset of index H that might possibly be feasible must then be derivable, in the manner of 3, from two unique feasible subsets of index H-1.

In brief, the procedure involves selecting all possible combinations of S_1 and S_2 from the list (of feasible subsets of index H-1); determining S, $S_3, \ldots S_H$ from the definitions and comments above; and then--if $S_3, \ldots S_H$ are all listed as feasible--testing S directly for feasibility.

In detail, consider the first subset in the list as a possible subset S_1 for some subset S. If it is such, S_2 is another subset in the list. Because of comments 1 and 2 and the ordering of this list as described above, all subsets in the list which can possibly function as S_2, with respect to this S_1, must follow S_1 consecutively in the list, in a group.

For $H = 3$, in the example, select $S_1 = [1, 2]$.
The only possible subsets that can be S_2 are
[1, 3], [1, 4], [1, 5], [1, 7], [1, 8], [1, 9], [1, 10],
[1, 11], and [1, 12].

The first of these possible subsets (if any) is selected and considered temporarily as S_2.

The corresponding S is determined from S_1 and S_2, and then $S_3, \ldots S_H$ are determined from S and looked up in the list to see if they are feasible subsets of index H-1. If $S_3, \ldots S_H$ are all listed, then subset S is tested directly (by resource types) to determine its feasibility, and if it is feasible, it is listed as a feasible subset of index H-1.

Returning to the example, where $S_1 = [1, 2]$ and S_2
is taken as [1, 3], the subset S is [1, 2, 3] and S_3
is [2, 3]. Since S_3 is feasible, S itself is tested
and listed as a feasible subset of order 3. (This
procedure is outlined in detail in figure 1.)

If some S_i is missing from the list, then it is not feasible and by the main principle neither is S feasible, so there is no need to look up any more S_i's nor to test S itself for feasibility. At this point, the next of the possible subsets (if any) described in the last paragraph is selected and considered temporarily as S_2 instead. The process described in this paragraph is repeated with all the possible S_2's for the S_1 under consideration.

After all S_2's have been considered, or if there are no subsets at all which can possibly function as S_2 with respect to this S_1, the process in the last two paragraphs is repeated with the subset following S_1 in the list of feasible subsets of index H-1 considered as a possible S_1. This procedure tests and lists each feasible subset S in its proper order. The restrictions involving the ordering of various operations eliminate calculations for redundant permutations of subsets. The example shows, as might be expected, that very few nonfeasible subsets ever progress far enough in this process, before being discarded, to be tested directly for feasibility. The most time-consuming part of the procedure is thus frequently avoided.

In the same way that this procedure derives the list of feasible subsets of index H from the list of index H-1, it is repeated to derive the list of index $H + 1$ from the list of index H; and so on, until some index K is reached such that there exists at least one feasible subset of index K but none of index $K + 1$. K represents the maximum number of operations that can be scheduled simultaneously. At this point the feasible subsets have all been determined, and the algorithm proceeds to part II.

Part II: Selection of Schedules

This part of the algorithm subdivides the set of operations into a minimum number of disjoint feasible subsets. Since each feasible subset corresponds to a subset of operations that can run simultaneously, this minimum number of subsets represents a minimum total scheduled time for all the operations. By this means the algorithm finds one or more optimal schedules.

In the example of part I, some possible subdivisions (which are not necessarily optimal) are:

[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12],
[1, 2] [3, 7] [4, 8] [5, 9] [6, 10] [11, 12],
[1, 8, 9, 11] [2, 4] [3, 7] [5, 10] [6, 12], and
[1, 8, 9, 10, 11] [3, 8] [4, 12] [5] [6] [7].

The only information necessary for this part is the output of part I, the number of operations, N, and the listing, by index, of all feasible subsets. (See table 5, for the example).

Table 5. Feasible Subsets of Indexes 1 Through 5

Feasible Subsets of Index 1											
[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]	[12]
Feasible Subsets of Index 2											
[1, 2]	[1, 3]	[1, 4]	[1, 5]	[1, 7]	[1, 8]	[1, 9]	[1, 10]	[1, 11]			
[1, 12]	[2, 4]	[2, 10]	[3, 7]	[3, 8]	[4, 8]	[4, 9]	[4, 10]	[4, 12]	[5, 8]		
[5, 9]	[5, 10]	[6, 7]	[6, 10]	[6, 12]	[7, 8]	[7, 10]	[7, 11]	[8, 9]			
[8, 10]	[8, 11]	[9, 10]	[9, 11]	[10, 11]	[10, 12]	[11, 12]					
Feasible Subsets of Index 3											
[1, 2, 10]	[1, 3, 8]	[1, 4, 12]	[1, 5, 8]	[1, 7, 8]	[1, 8, 9]	[1, 8, 10]					
[1, 8, 11]	[1, 9, 10]	[1, 9, 11]	[1, 10, 11]	[1, 10, 12]	[1, 11, 12]	[4, 8, 9]	[4, 9, 10]				
[5, 8, 9]	[7, 8, 10]	[7, 8, 11]	[7, 10, 11]	[8, 9, 10]	[8, 9, 11]	[8, 10, 11]	[9, 10, 11]				
Feasible Subsets of Index 4											
[1, 8, 9, 10]	[1, 8, 9, 11]	[1, 8, 10, 11]	[1, 9, 10, 11]	[7, 8, 10, 11]	[8, 9, 10, 11]						
Feasible Subset of Index 5											
[1, 8, 9, 10, 11]											

Procedure

With some number L_1, known not to be greater than the minimum number of subsets in a solution, a survey is made to determine whether there exists a collection of L_1 disjoint feasible subsets which include each schedulable operation.

Since each of the N operations in such a collection must appear once and only once (the subsets being disjoint), the sum of the indices of the L_1 subsets must be N. Therefore, none of the desired collections is missed if this survey is limited to collections of L_1 disjoint feasible subsets whose indices add up to N.

If no such collection is found for L_1, the survey is repeated for $L_2 = L_1 + 1$, $L_3 = L_2 + 1$, etc., in place of L_1 until at least one collection is found for some L--designated as L_{optimum}. Each collection represents an optimal schedule with the operations scheduled simultaneously in subsets and with the subsets scheduled in any order. (The order might depend on additional constraints not mentioned in the problem formulation above. See the discussion of this in the generalization below.) L_{optimum} is the optimum time for the scheduling of these operations, in terms of the unit of time that each of these operations is supposed to last.

Before discussing any further details of this procedure, it is necessary to define a special classification of the collections of subsets under consideration. Each of these collections can be classified according to how many of its subsets have index H, for each value of H in the range $1 \leq H \leq K$, K being the index of the largest feasible subset found in part I. This classification is defined as a *configuration* and it is represented mathematically by a K-dimensional *configuration vector* $X_a = (x_{a1}, x_{a2}, \ldots x_{aK})$ where each x_{aH} is the number of subsets in the collection which have index H. (These vectors should not be confused with those in part I. The only property in common between the two concepts is that each represents a string of numbers to be handled in a certain way.)

The configuration vectors for the subdivisions, or collections, cited above are

(12, 0, 0, 0, 0),
(0, 6, 0, 0, 0),
(0, 4, 0, 1, 0), and
(3, 2, 0, 0, 1), respectively.

Since $L = \sum_{i=1}^{K} x_{a,i}$ and $N = \sum_{i=1}^{K} i \cdot x_{a,i}$ for any configuration vector X_a, a configuration vector can generate another configuration vector with the same L and N under the following type of transformation:

For any i and j such that $2 \leq i \leq j \leq K - 1$
and such that x_{ai} and x_{aj} are greater than 0,
(1) subtract 1 each from $x_{a,i}$ and $x_{a,j}$ if $i < j$, or subtract 2
if $i = j$ and if $x_{a,i} \geq 2$; and (2) add 1 to $x_{a,i-1}$ and $x_{a,j+1}$.

Given any L and N, there is a configuration vector which can generate all of the other configuration vectors by a sequence of transformations of this type. (A few experiments with this process will make this last statement obvious to the reader.) This vector, which is referred to as the *primary configuration vector* is determined numerically as follows:

$$x_{a,\left[\frac{N}{L}\right]} = L - N + L \cdot \left[\frac{N}{L}\right]$$

$$x_{a,\left[\frac{N}{L}\right]+1} = N - L \cdot \left[\frac{N}{L}\right]$$

and all other x's = 0.[8]

The primary configuration vector represents a configuration in which all of its subsets are "compressed" into one or two adjacent indices, so that no other configuration vector can possibly generate this vector. On the other hand, the last configuration vector in each hierarchy has at

[8] $[u]$ is the highest integer $\leq u$.

most one nonzero coefficient, excluding the first and last coefficients, and this coefficient is not greater than 1, so it is not possible for this vector to generate another vector. The configuration vectors for L_1, L_2, L_3, and L_4 in the example are shown, with the generative relationships among them, in figure 2.

The following details of this procedure are illustrated in the accompanying flow chart (figure 3)[9].

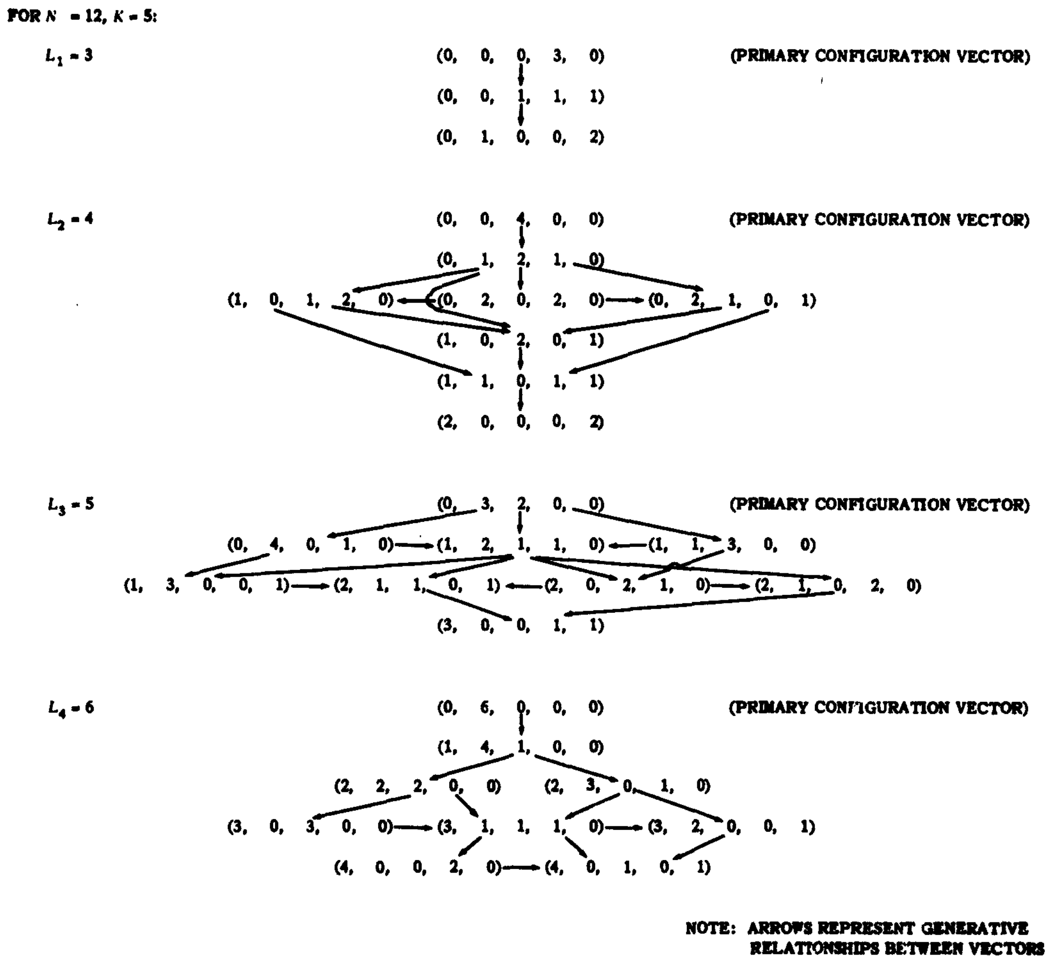

Figure 2. Configuration Vectors in the Example.

[9]Since the above exposition was written, it has been learned by the author that a very fundamental concept underlying this algorithm is known as the partition function of additive number theory. An alternate generative process for configuration vectors due to Hindenburg appears in *History of the Theory of Numbers, Volume II, Diophantine Analysis*, p. 106, by L. E. Dickson, 1920. Carnegie Institution. Washington, D. C.

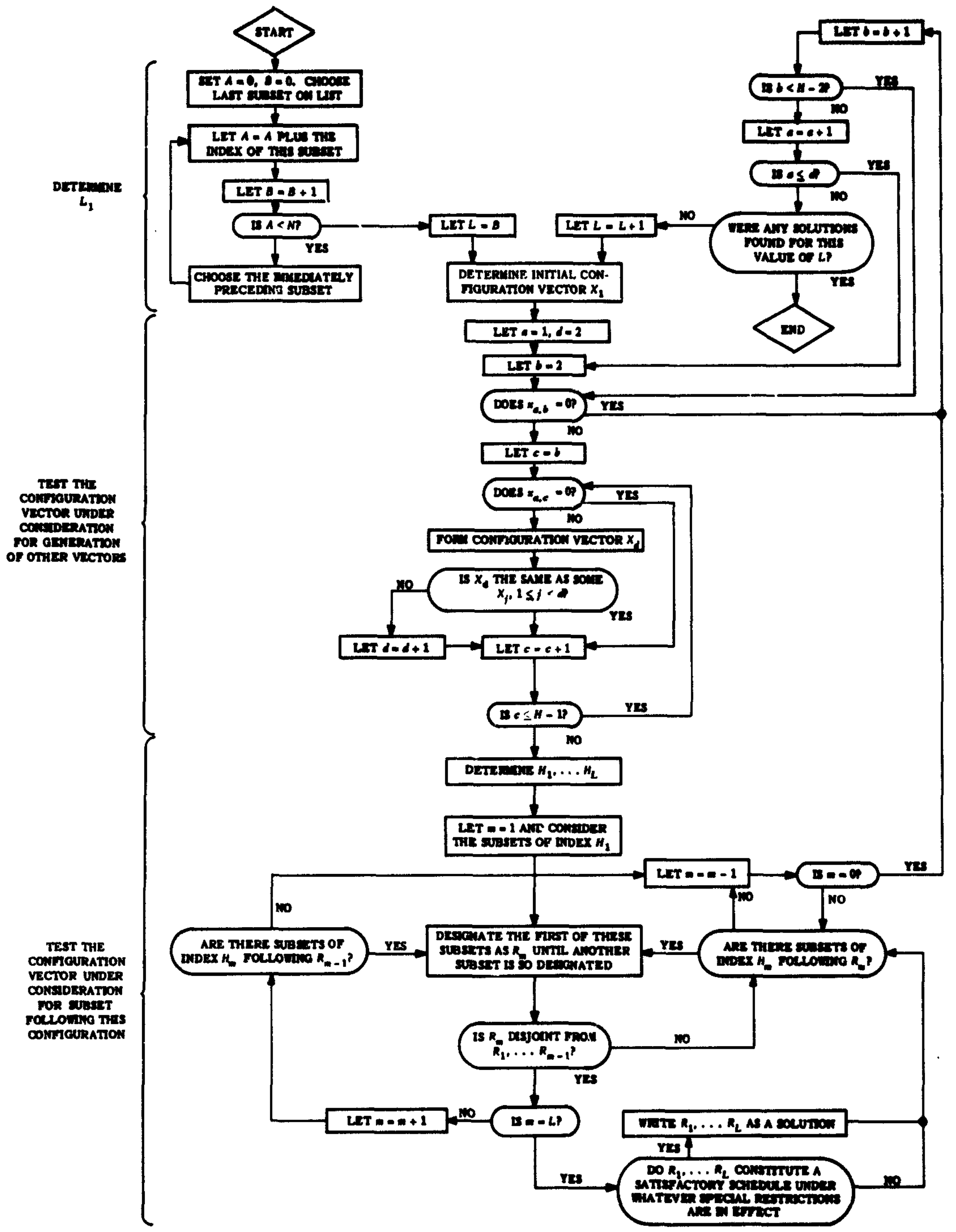
START
SET $A = 0$, $B = 0$. CHOOSE LAST SUBSET ON LIST
LET $A = A$ PLUS THE INDEX OF THIS SUBSET
LET $B = B + 1$
IS $A < N$?
YES
CHOOSE THE IMMEDIATELY PRECEDING SUBSET
DETERMINE L_1
LET $L = B$
LET $L = L + 1$
NO
WERE ANY SOLUTIONS FOUND FOR THIS VALUE OF L?
YES
END
LET $b = b + 1$
IS $b < H - 2$?
YES
NO
LET $a = a + 1$
IS $a \leq d$?
YES
NO
DETERMINE INITIAL CONFIGURATION VECTOR X_1
LET $a = 1$, $d = 2$
LET $b = 2$
DOES $x_{a,b} = 0$?
YES
NO
LET $c = b$
DOES $x_{a,c} = 0$?
YES
NO
FORM CONFIGURATION VECTOR X_d
IS X_d THE SAME AS SOME X_j, $1 \leq j < d$?
NO
YES
LET $d = d + 1$
LET $c = c + 1$
IS $c \leq H - 1$?
YES
NO
TEST THE CONFIGURATION VECTOR UNDER CONSIDERATION FOR GENERATION OF OTHER VECTORS
DETERMINE $H_1, \ldots H_L$
LET $m = 1$ AND CONSIDER THE SUBSETS OF INDEX H_1
LET $m = m - 1$
IS $m = 0$?
YES
NO
NO
ARE THERE SUBSETS OF INDEX H_m FOLLOWING R_{m-1}?
YES
NO
DESIGNATE THE FIRST OF THESE SUBSETS AS R_m UNTIL ANOTHER SUBSET IS SO DESIGNATED
YES
ARE THERE SUBSETS OF INDEX H_m FOLLOWING R_m?
IS R_m DISJOINT FROM $R_1, \ldots R_{m-1}$?
NO
YES
LET $m = m + 1$
NO
IS $m = L$?
YES
WRITE $R_1, \ldots R_L$ AS A SOLUTION
YES
DO $R_1, \ldots R_L$ CONSTITUTE A SATISFACTORY SCHEDULE UNDER WHATEVER SPECIAL RESTRICTIONS ARE IN EFFECT
NO
TEST THE CONFIGURATION VECTOR UNDER CONSIDERATION FOR SUBSET FOLLOWING THIS CONFIGURATION

Figure 3. Flow Chart for Part II of the Algorithm.

Let L_1 be the smallest number of feasible subsets (disjoint or not), whose indices can add up at least to N. Its value can be determined simply by adding the indices of the highest indexed subsets, one at a time, until their cumulative index total is at least N. The number of subsets whose indices are added is an appropriate value for L_1. (Since these particular subsets are not necessarily disjoint, this does not imply that a solution has been found.)

Taking the subsets in the example in order of highness of indices, one at a time, L_1 is calculated as follows:

Subsets	Indices	Cumulative Index Total
[1, 8, 9, 10, 11]	5	5
[8, 9, 10, 11]	4	9
[7, 8, 10, 11]	4	13

Since only three such subsets are required to make the cumulative index total at least 12, L_1 is taken as 3.

Given L_1, K, and N, the primary configuration vector is determined as above. This configuration vector, X_1, is the first of a small list of configuration vectors to be computed.

Determine for the primary configuration vector, all vectors which this vector generates directly. These vectors are listed after X_1. Each vector on the list is used as a generator and any new vectors generated are added to the list. When every vector has been used as a generator, the list is complete.

In the example, where $L = L_2$, when configuration vector (0, 0, 4, 0, 0) is under consideration, only the vector (0, 1, 2, 1, 0) is generated and added to the list. When the latter vector is under consideration, vectors (1, 0, 1, 2, 0), (0, 2, 0, 2, 0), (0, 2, 1, 0, 1), and (1, 0, 2, 0, 1) are added--and so on, until all of the configuration vectors are generated and listed for L_2.

Next, the configuration vector under consideration is tested to see if there is any collection of disjoint feasible subsets satisfying that particular configuration.

For this configuration, let H_1 be the index of the largest indexed subset, H_2 for the next largest, etc.

For example. if the configuration vector is (0, 1, 2, 1, 0);
$H_1 = 4$, $H_2 = 3$, $H_3 = 3$, and $H_4 = 2$.
Similarly, for the configuration vector (0, 0, 4, 0, 0),
$H_1 = H_2 = H_3 = H_4 = 3$.

Now, considering the feasible subsets in a single list (with the subsets of higher indices listed earlier) a systematic survey is made by trying to find the necessary number of disjoint subsets of each index, the higher indices first. This procedure appears in detail at the bottom of figure 3.

For some unsatisfactory configurations, this procedure is very short since it determines very quickly that the specified number of disjoint subsets of a certain index may be more than can be provided.

The following remarks follow directly, for the example, from table 5:

No two feasible subsets of index 4 or 5 are disjoint, hence there can be at most one subset of index 4 or of index 5, but not both in any configuration tested. This restriction rules out the following configurations of figure 2:

(0, 0, 0, 3, 0), (1, 0, 1, 2, 0), (2, 0, 0, 0, 2),
(0, 0, 1, 1, 1), (0, 2, 0, 2, 0), (2, 1, 0, 2, 0),
(0, 1, 0, 0, 2), (1, 1, 0, 1, 1), and (3, 0, 0, 1, 1).

If there is a feasible subset of index 4 in a configuration, there should be no more than one disjoint feasible subset in that configuration of index 3. This restriction eliminates

(0, 1, 2, 1, 0) and
(2, 0, 2, 1, 0) additionally.

If there is a feasible subset of index 5 in a configuration, there should be no subsets of index 3.

This further eliminates

(0, 2, 1, 0, 1),
(1, 0, 2, 0, 1), and
(2, 1, 1, 0, 1).

Since no group of four subsets of index 3 are disjoint, vector (0, 0, 4, 0, 0) is also eliminated.

All of the configuration vectors for $L_1 = 3$ and $L_2 = 4$ are eliminated. So it is necessary to examine all of the remaining configuration vectors for $L_3 = 5$.

<u>(0, 3, 2, 0, 0)</u>

If [1, 2, 10] and [4, 8, 9] are considered as subsets of index 3 in this configuration, the only subsets of index 2 disjoint to these subsets are [3, 7], [6, 7], [6, 12], [7, 11], and [11, 12].
No three of the latter subsets are disjoint.
If [1, 2, 10] and [5, 8, 9] are the subsets of index 3, instead, the possible subsets of index 2 are [3, 7], [4, 12], [6, 7], [6, 12], [7, 11], and [11, 12].
For each of the following combinations of two disjoint subsets of index 3 there is no group of three subsets of index 2, disjoint to that combination, which are disjoint to each other:

[1, 2, 10]	[7, 8, 10]	[1, 5, 8]	[4, 9, 10]
[1, 2, 10]	[8, 9, 11]	[1, 5, 8]	[7, 10, 11]
[1, 3, 8]	[4, 9, 10]	[1, 7, 8]	[4, 9, 10]
[1, 3, 8]	[9, 10, 11]	[1, 7, 8]	[9, 10, 11]
[1, 4, 12]	[5, 8, 9]	[1, 8, 9]	[7, 10, 11]
[1, 4, 12]	[7, 8, 10]	[1, 8, 11]	[4, 9, 10]
[1, 4, 12]	[7, 8, 11]	[1, 9, 10]	[7, 8, 11]
[1, 4, 12]	[7, 10, 11]	[1, 9, 11]	[7, 8, 10]
[1, 4, 12]	[8, 9, 10]	[1, 10, 11]	[4, 8, 9]
[1, 4, 12]	[8, 10, 11]	[1, 10, 12]	[4, 8, 9]
[1, 4, 12]	[9, 10, 11]	[1, 10, 12]	[5, 8, 9] (Cont'd)

[1, 10, 12] [7, 8, 11]
[1, 10, 12] [8, 9, 11]
[1, 11, 12] [4, 8, 9]
[1, 11, 12] [4, 9, 10]
[1, 11, 12] [5, 8, 9]
[1, 11, 12] [7, 8, 10]
[1, 11, 12] [8, 9, 10]
[4, 8, 9] [7, 10, 11]
[4, 9, 10] [7, 8, 11]

However, the following solutions do exist:

[1, 3, 8] [7, 10, 11] [2, 4] [5, 9] [6, 12],
[1, 5, 8] [9, 10, 11] [2, 4] [3, 7] [6, 12],
[1, 10, 11] [5, 8, 9] [2, 4] [3, 7] [6, 12], and
[5, 8, 9] [7, 10, 11] [1, 3] [2, 4] [6, 12].

(1, 2, 1, 1, 0)

Considering [8, 9, 10, 11] as the subset of index 4 in this collection, the only disjoint subset of index 3 is [1, 4, 12] and the only subsets of index 2 disjoint to each of these subsets are [3, 7] and [6, 7]. But [3, 7] and [6, 7] are not disjoint. The only other subset of index 4 disjoint from some subset of index 3 is [7, 8, 10, 11], and the only disjoint subsets of index 3 is [1, 4, 12] again. The only disjoint subset of index 2 is [5, 9], so there is no solution with this configuration.

(1, 1, 3, 0, 0)

The only group of three disjoint subsets of index 3 is [1, 4, 12], [5, 8, 9], and [7, 10, 11]. But there is no disjoint subset of index 2.

(1, 3, 0, 0, 1)

The only subset of index 5 is [1, 8, 9, 10, 11]. The subsets of index 2 disjoint to this subset are [2, 4], [3, 7], [4, 12], [6, 7], and [6, 12]. Also, the only disjoint group of three of these subsets is [2, 4], [3, 7], and [6, 12]. Therefore, the only solution of this configuration is [1, 8, 9, 10, 11] [2, 4] [3, 7] [6, 12] [5].

(0, 4, 0, 1, 0)

If [1, 8, 9, 10] is considered as the subset of index 4, there are no four disjoint subsets of index 2. However, the following solutions do exist with the other possible subsets of index 4:

[1, 8, 9, 11] [2, 4] [3, 7] [5, 10] [6, 12],
[1, 8, 10, 11] [2, 4] [3, 7] [5, 9] [6, 12],
[1, 9, 10, 11] [2, 4] [3, 7] [5, 8] [6, 12],
[7, 8, 10, 11] [1, 3] [2, 4] [5, 9] [6, 12], and
[8, 9, 10, 11] [1, 5] [2, 4] [3, 7] [6, 12].

GENERAL FORMULATION OF THE SCHEDULING PROBLEM

The assumptions in the initial formulation restrict the algorithm too much indeed for most scheduling problems. The following modifications eliminate or mitigate some of the restrictions to make the algorithm versatile enough for most, if not all, situations.

Modification 1: Special Time Assignments

It was assumed initially that each operation could be scheduled for any time (assumption 2). The starting times for some operations are actually limited to some short span of time and it is folly to consider "feasible" subsets which have two operations with no such time in common. To eliminate such subsets from even being considered as feasible, it is sufficient to invent an additional resource type for each pair of operations with no scheduling time in common, and to choose 1 to be the corresponding coefficient for the system vector and each of the vectors in the pair. As each "optimal solution" is found in part II of the algorithm, an attempt should be made to schedule each subset at a time satisfactory to all of its operations. If this is found to be impossible, the optimal solution under consideration should no longer be considered as such and should not be listed. The algorithm should resume as if it were still searching for a solution. If such a schedule is found, the optimal solution should be listed in a form suitable for reading as a schedule.

Modification 2: Operations of Longer Duration

Most of the operations that last longer than one unit of time generally require and monopolize different resources for different segments of time. Some resources are required in advance of execution for special adjustments. On the other hand, certain resources are tied up afterwards for mop-up jobs, such as collection of data (see figures 4 and 5).

Unfortunately, this kind of operation violates assumption 1 in the initial problem formulation. A lighter restriction is necessary, so the following assumption is substituted for the former one:

7. An operation may last more than one unit of time, but its resource requirements must remain constant within time segments which are integer multiples of the selected unit of time.

Figure 4 shows such an operation, where the number of units of each resource changes from one unit of time to the next.

The procedure for this modification is mainly to split each long operation into smaller operations of one unit of time in duration, each smaller operation representing the original operation during one particular unit of time.

Of course, since these smaller operations must run at different times, modification 1 should be applied to those operations as a group.

For example, in figure 4, the total duration of operation A is four units of time, so four smaller vectors

(1, 0, 6, 12, 0, 125, 0, 0.0, 0),
(0, 3, 2, 10, 0, 70, 1, 2.0, 0),
(0, 0, 1, 3, 0, 60, 0, 0.0, 1), and
(0, 0, 0, 10, 8, 0, 1, 0.0, 2)

should be substituted for it.

By a contrivance, these vectors are identical with the vectors for operations 5, 2, 1, and 7, respectively.

In figure 5, operation B is shown to be equivalent to the sequence of operations 8, 10, and 6--in that order.

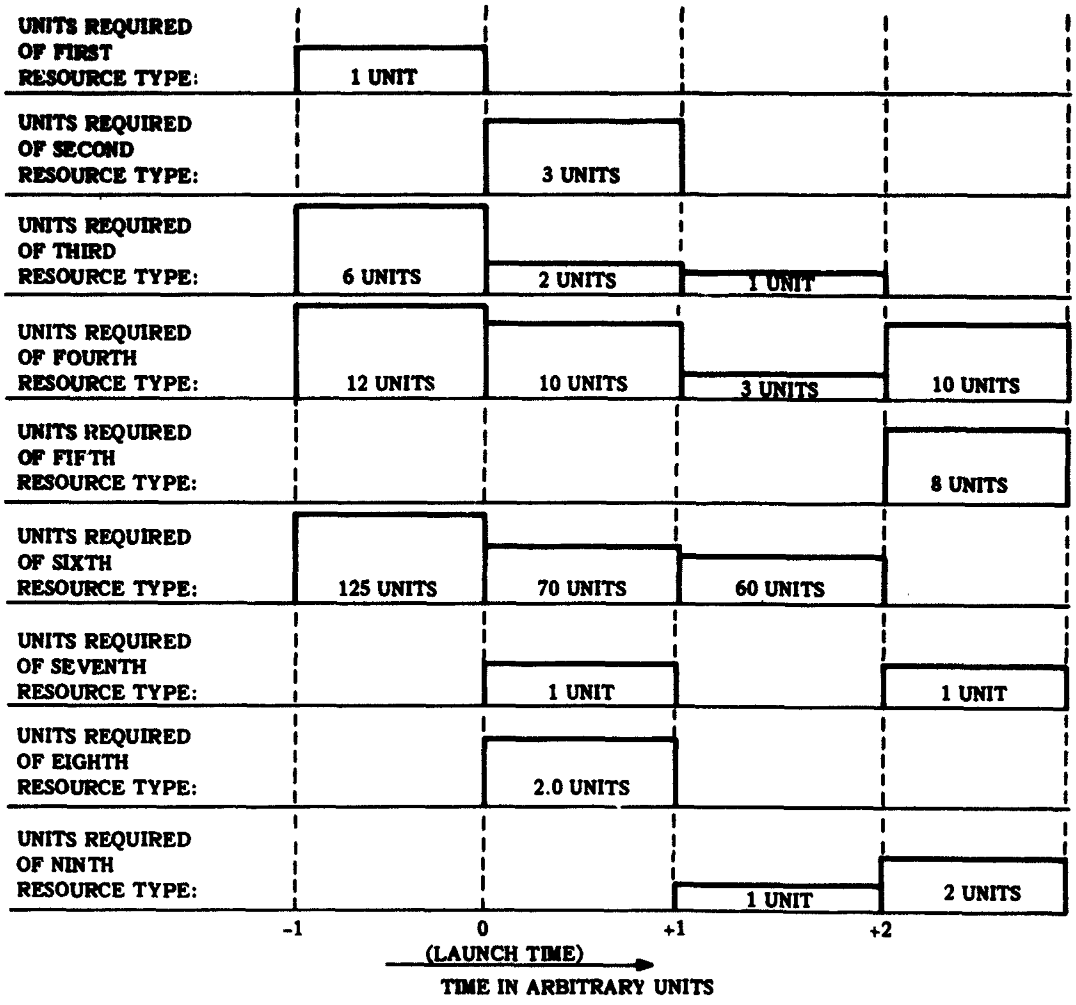

Figure 4. Example of an Operation With Fluctuating Resource Requirements That the Algorithm Can Schedule Under Modification 2 (Operation A).

The problem of scheduling operations A, B, 3, 4, 9, 11, and 12 into a segment of time as short as possible is equivalent to the example included in the original exposition of the algorithm above except for the following constraint.

Operations 5, 2, 1, and 7 must follow in that consecutive order and 8, 10, and 6 must also. Each "optimal solution" found in the algorithm should be tested for this constraint when it is tested for the constraint of modification 1.

The following solution from the example satisfies this constraint. The subsets are taken in a satisfactory order.

[5, 9] [2, 4] [1, 3, 8] [7, 10, 11] [6, 12]

This is the only optimal solution to this problem.*

*NOTE: Further consideration of modification 2 has been given to the problem since the departure of the author. The de-composition of operations into smaller-length operations satisfying assumption 1 may cause an excessive computer load in part I of the algorithm.

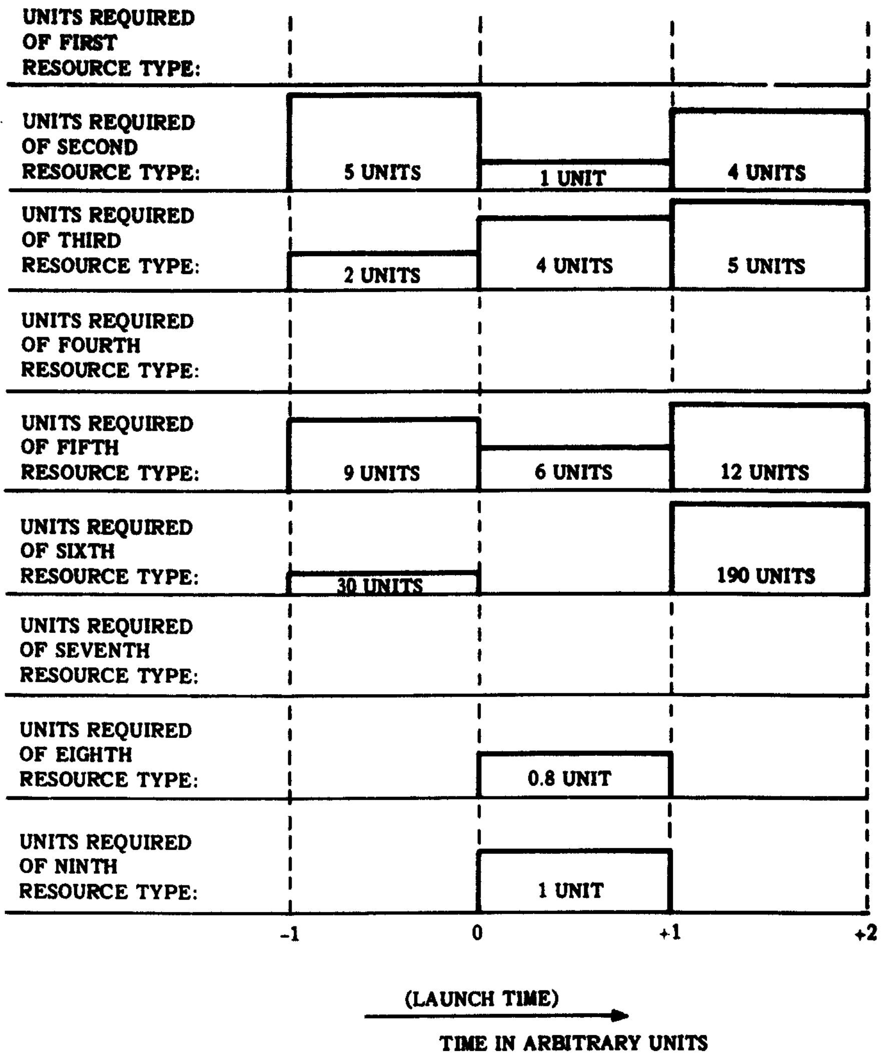

Figure 5. Example of an Operation With Fluctuating Resource Requirements That the Algorithm Can Schedule Under Modification 3 (Operation B).

Modification 3: Change in the System

Assumption 6 in the initi⁻l problem formulation is restrictive in a situation where the operation of certain resources may be preempted at certain times of the day by maintenance or other nonoperating tasks. Nonoperating activities can be accounted for simply by adding artificial "operations" which must be scheduled at the appropriate times and which require whatever resources are in question.