

PROCEDURES FOR LOT SIZE DETERMINATION IN
TWO LEVEL PRODUCTION SYSTEMS

A THESIS

Presented to

The Faculty of the Division of Graduate Studies

By

Ricardo R. Lopez Canales

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

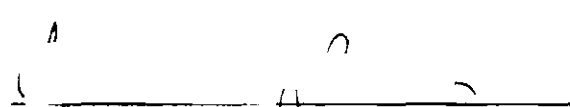
in the School of Industrial and Systems Engineering

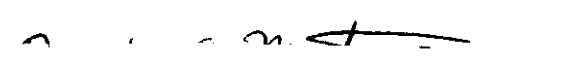
Georgia Institute of Technology

December, 1977

PROCEDURES FOR LOT SIZE DETERMINATION IN
TWO LEVEL PRODUCTION SYSTEMS

Approved:


L. A. Johnson, Chairman


D. C. Montgomery


L. F. McGinnis

Date approved by Chairman Nov. 29, 1977

ACKNOWLEDGEMENTS

I wish to express my deepest appreciation to my Thesis Advisor, Dr. L. A. Johnson, a constant source of patience, understanding and encouragement. His guidance and enthusiasm were indispensable in the completion of this work.

Distinct thanks are also due to Dr. D. C. Montgomery and Dr. L. F. McGinnis for participating in my Thesis Committee and also providing constructive criticisms in the preparation of the final draft.

I wish to also thank Dr. David E. Fyffe for his help throughout my graduate studies.

Finally I dedicate this Thesis to my parents, whose unyielding support and encouragement have been the reason for the achievement of goals through my years of education.

TABLE OF CONTENTS

	Page
ACKNOWLEDGEMENTS	iii
LIST OF TABLES	v
LIST OF ILLUSTRATIONS	vi
Chapter	
I. INTRODUCTION	1
Description of the Problem	
Objectives of the Research	
Scope of the Research	
Organization of the Material	
II. STATIONARY DEMAND CASE	10
Series Structure	
Assembly Structure	
Arborescence Structure	
Acyclic Structure	
III. TIME VARYING DEMAND CASE	68
Series Structure	
Assembly Structure	
Arborescence Structure	
Acyclic Structure	
IV. CONCLUSIONS AND RECOMMENDATIONS	121
Results of the Investigation	
Recommendations	
APPENDICES	127
BIBLIOGRAPHY	183

LIST OF TABLES

Table	Page
1. Results For Assembly Structure	38
2. Data For Assembly Problems	40
3. Heuristic Results	44
4. Problems For Arborescence Structure	53
5-A. Results For Arborescence Structure	55
5-B. Detailed Results For Modified Myopic Policies	57
6. Data And Results For Acyclic Structure	64
7. Problems For Series Structure	78
8. Production Schedule In Series Structure, Time Varying Demand	80
9. Problems For Assembly Structure, Time Varying Demand . . .	88
10. Production Schedules For Assembly Structure, Time Varying Demand	91
11. Problems For Arborescence Structure, Time Varying Demand .	100
12. Production Schedule In Arborescence Structure, Time Varying Demand	103
13. Problems For Acyclic Structure, Time Varying Demand . . .	114
14. Production Schedule For Acyclic Structure, Time Varying Demand	117

LIST OF ILLUSTRATIONS

Figure	Page
1. General Structure	2
2. Series Structure	3
3. Assembly Structure	4
4. Arborescence Structure	5
5. Acyclic Structure	6
6. A Two-Level Serial System	11
7. Inventories Under Integer Multiple Policy With Simultaneous Production and Consumption	14
8. Inventories Under Integer Multiple Policy With Nonsimultaneous Production and Consumption	16
9. Inventories Under Integer Divisor Policy With Simultaneous Production and Consumption	18
10. Inventories Under Integer Divisor Policy With Nonsimultaneous Production and Consumption	20
11. Two-Level Assembly Structure	24
12. Series Structure	25
13. Arborescence Structure In Two Levels	42
14. Acyclic Structure	59
15. Inventory Model	60
16. Acyclic Problem	62
17. Dynamic Series Structure	69
18. Network Representation	73
19. Dynamic Assembly Structure	81
20. Dynamic Arborescence Structure	90

LIST OF ILLUSTRATIONS (continued)

Figure		Page
21.	Dynamic Acyclic Structure	107
22.	Acyclic Problem	113
A-1	$S_1/S_2 = .1$ Simultaneous Production and Consumption	129
A-2	$S_1/S_2 = .20$ Simultaneous Production and Consumption	130
A-3	$S_1/S_2 = .50$ Simultaneous Production and Consumption	131
A-4	$S_1/S_2 = 1$ Simultaneous Production and Consumption	132
A-5	$S_1/S_2 = 2$ Simultaneous Production and Consumption	133
A-6	$S_1/S_2 = 5$ Simultaneous Production and Consumption	134
A-7	$S_1/S_2 = 10$ Simultaneous Production and Consumption	135
A-8	$S_1/S_2 = .10$ Nonsimultaneous Production and Consumption	136
A-9	$S_1/S_2 = .20$ Nonsimultaneous Production and Consumption	137
A-10	$S_1/S_2 = .50$ Nonsimultaneous Production and Consumption	138
A-11	$S_1/S_2 = 1$ Nonsimultaneous Production and Consumption	139
A-12	$S_1/S_2 = 2$ Nonsimultaneous Production and Consumption	140
A-13	$S_1/S_2 = 5$ Nonsimultaneous Production and Consumption	141
A-14	$S_1/S_2 = 10$ Nonsimultaneous Production and Consumption	142

CHAPTER I

INTRODUCTION

Description of the Problem

The classical economic lot size model determines the order quantity that minimizes the sum of set up costs and inventory carrying costs for a single production or procurement operation. The analysis leading to the development of the model assumes that the acquisition of the item stocked is from an external source and that the requirements satisfied by withdrawal from inventory are imposed by a second external source. Although such models are widely used for controlling inventories of finished goods, in-process items, and purchased materials, these assumptions are never true for a production system. For any given inventory stage in a production system requiring lot size decisions in the resupply process, there is at least one predecessor stage or successor stage whose replenishment decisions are also under control. To obtain lot size policies that have some claim to being optimal, the interdependences among related lot size decisions must be explicitly considered. The purpose of this research was to investigate the determination of economic lot size policies in multistage production systems.

Before giving a more complete description of the problem, it is convenient to define the terms stage and level.

A stage is a point in the production process where material remains for a period of time, and where the material may or may not receive processing that increases its value.

A level indicates the specific position of a stage in the multi-stage structure, according to the number of successor levels in series following it. Figure 1 illustrates the concept, where Stage 5 produces the finished product. Note that a Level 1 stage must supply requirements imposed by external sources.

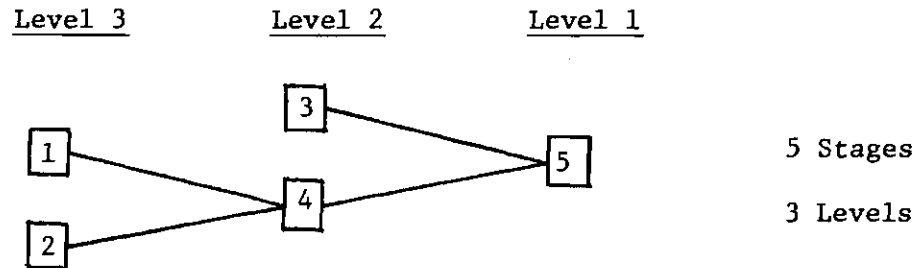


Figure 1. General Structure

The stages may have many different structural arrays and this fact serves to classify the kind of system under study. Multistage systems have been classified into four principal types for investigation: series structure, assembly structure, arborescence structure, and acyclic structure.

Series Structure

This is the case where each stage may have at most one stage as immediate predecessor or immediate successor. The demand at each stage is determined by the requirements of the next stage, except for the final stage, which must satisfy the external product demand. The first stage obtains its input from an external supplier, who responds to the ordering policy used by that stage.

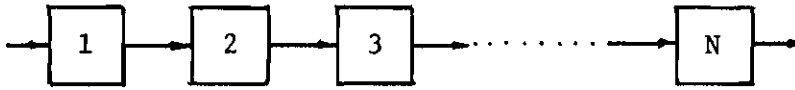


Figure 2. Series Structure

In the series structure, the number of stages is exactly equal to the number of levels (see Figure 2).

This structure is commonly found in industry when the process starts with a unique raw material; and it is processed, operation by operation, until it is transformed into a final product. (For example, many metallic parts require a sequence of pressed operations that are executed in lots which are moved from stage to stage in sequence, representing exactly the series structure described above.)

Assembly Structure

Each stage in a multi-stage assembly system may have any number of immediate predecessors, but has at most one immediate successor. For example in the assembly network of Figure 1, stage 4 has two immediate predecessors (1, 2) and one immediate successor (5). Figure 3 gives a more understandable presentation of the assembly structure.

As in the series case, the demand at any stage will be determined by the successor's demand, except for the final stage which must satisfy external demand.

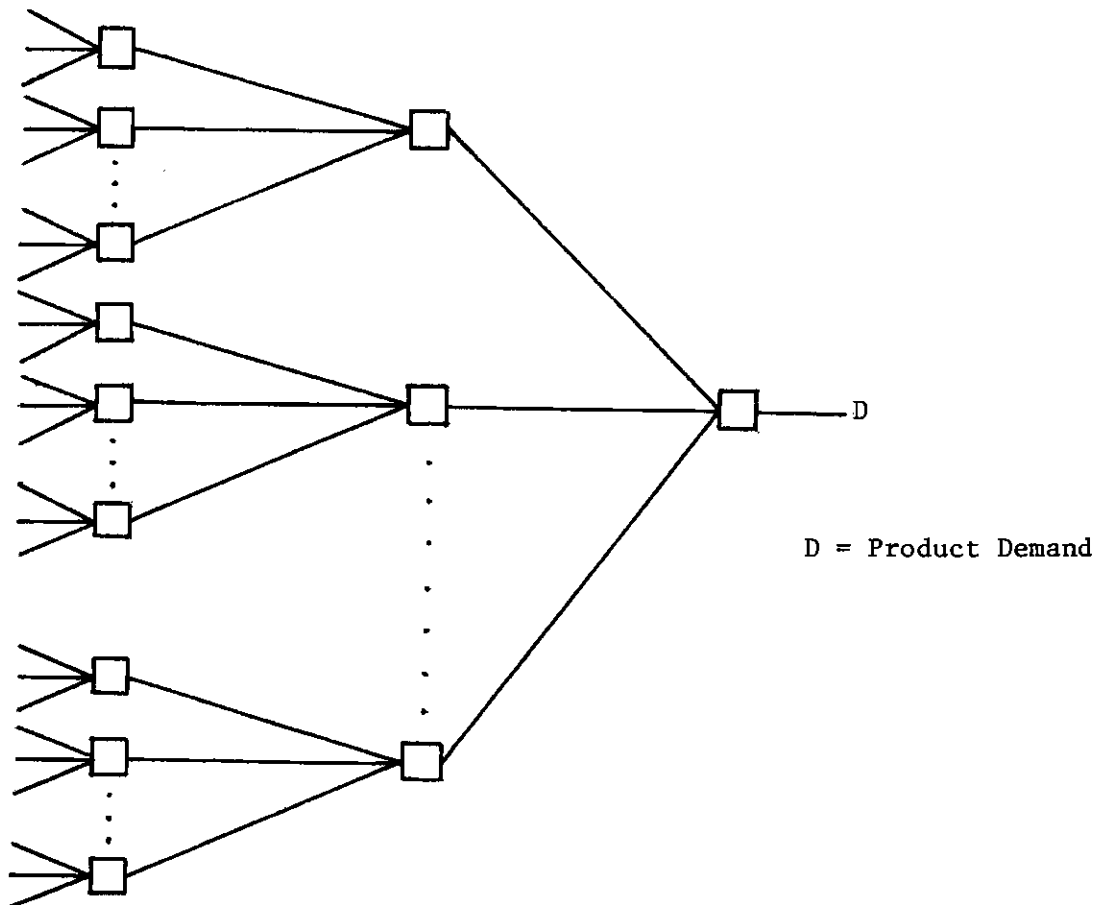


Figure 3. Assembly Structure

This kind of structure is, as its name implies, found in the production systems where products are assembled from components. It may be called a pure assembly system, since it is assumed that a component is used in exactly one higher level item. Examples of this particular structure are found in the electric and electronic industries, the automobile industry, the furniture industry, and many others.

It is interesting to observe that the organigrams in any kind of

organization follow the same pattern as the assembly structure, where the sub-assemblies are the managers and their predecessors are their subordinates. Now considering the information as material, it is possible to find an optimal cycle time for reporting information to the next level and the amount of the information, based on the transmission information cost and the omission of information cost.

Arborescence Structure

In the arborescence structure, a stage may have any number of successors, but at most one predecessor (Figure 4). The demand at a stage will be determined by the sum of the immediate successors' demands with the exception of first level stages, whose requirements are dictated by external sources.

Typical examples of this case occur in the chemistry industry, where a common raw material is transformed into different sub-products or finished products. A situation where a purchased material is packaged into a variety of package types is a classical example of a simple two-level arborescence system.

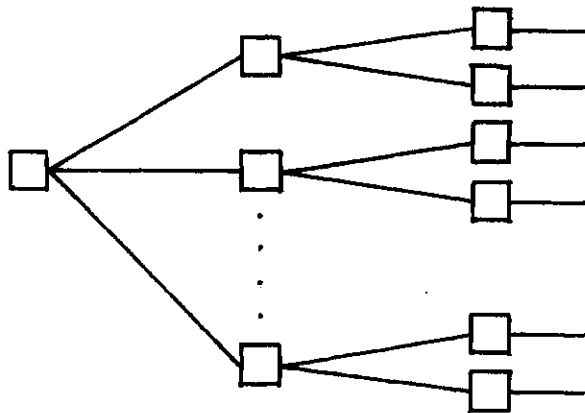


Figure 4. Arborescence Structure

Acyclic Structure

The acyclic structure can be considered as the most general, with the assembly and arborescence structures being special cases. (The serial system is a special case of both the assembly and the arborescence structures.) Here any stage (except at the first and last levels) may have any number of successors or predecessors. A first level stage will have no successors, but it may have any number of predecessors. Its demand will be the corresponding product demand. A stage at the lowest level will have no predecessors, but may have any number of successors. In order to understand the system, consider the three level acyclic structure shown in Figure 5. It could represent a large electric company, where it might be possible to identify the third level as purchased component inventory, with stages for transistors, circuit boards, tubes, wiring, etc. Then the second level would represent principal products like televisions, radios, refrigerators, and washing machines. Finally the first level stages would represent the points where all the products will be distributed or sold to the public.

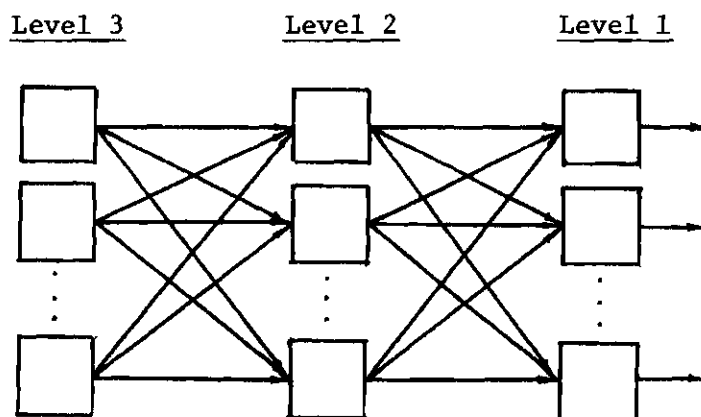


Figure 5. Acyclic Structure

Objectives of the Research

The purpose of this research is to gain a better understanding of multi-level systems and how the interrelations between stages affect the determination of optimal lot sizes.

Since the classical approach has emphasized its attention to the one-level production system, the logical next step would be the study of two-level production systems. Consequently this research has been directed toward organization and consolidation of existing relevant methodology and analysis of important cases not yet treated in order to provide a comprehensive analysis of the two-level system. In addition to it being a logical extension of single stage economic lot size analysis, this research is conducted on two-level systems because there are a variety of production-inventory systems that consist of only two levels, and the results should be of wide interest. Finally, the solution methodology developed for the two-level case may lend insight and contribute to heuristic approaches to the solution of general multi-level systems.

Scope of the Research

In studying multi-level systems, one encounters a variety of properties that characterize a given situation: the nature of external demand, the type of costs, the number of items, the structure of the system, and so on. Specifically the problems that have been considered by this research are: (1) a single item produced by each stage of a two-level system, with stationary external demand and infinite planning horizon; and (2) a single item produced by each stage of a two-level system, with time-varying external demand and finite planning horizon. For each of

these situations, the four types of structures described at the beginning of this chapter (series, assembly, arborescence, and acyclic) will be analyzed. It is desirable to emphasize that the term "single item" refers to a unique item at each stage of the structure.

Costs are assumed to vary among stages. The production or procurement cost at a stage is assumed to consist of a fixed set up cost for each lot produced plus a constant variable cost for each unit in the lot. This is the usual fixed charge production cost model of the form

$$C(Q) = \begin{cases} S + cQ, & Q > 0 \\ 0, & Q = 0 \end{cases}$$

where S is the set up cost, Q is the lot size, and c is the unit variable cost. The inventory holding costs are assumed to be proportional to the average inventory level in the infinite horizon case and proportional to end-of-period inventory in the periodic review, finite horizon case.

Only deterministic demand situations are considered. In all cases, it is assumed that no shortages are allowed; however, for many of the static demand models it is relatively easy to extend the results obtained to permit backlogging.

The case of multiple items competing for production resources at a given production stage was not considered.

The systems analyzed in this research are multi-level production systems where requirements on a given stage become exactly the requirements imposed on each immediate predecessor stage. This is in contrast with a multi-level distribution inventory system where the requirements

equal the sum of the requirements on its immediate predecessor stages.

Organization of the Material

The first section (Chapter II) of this thesis is devoted to the study of the stationary demand case, while the second section (Chapter III) treats the time-varying demand case. Within each section, the four basic system structures are analyzed in sequence. For each situation, the approach is to describe the problem, review the literature, present the analysis and one or more solution algorithms, give results from running test problems, and discuss these results. The final section (Chapter IV) contains conclusions and recommendations. Listings of all computer programs used in the study are given in the Appendix.

CHAPTER II

STATIONARY DEMAND CASE

Series StructureProblem Definition

The problem as it was described in the previous chapter, consists of two stages, or facilities, where the product item produced at the first stage is required in the production of the item at the second stage. It may be assumed, without loss of generality, that one unit of the first stage item is required in producing one unit of the second item. The requirements to be satisfied by first level production are assumed to be known and constant at a rate of D units per unit time. The problem is to determine the production policy to follow at each stage to minimize the relevant costs while satisfying the requirements.

The production policy adopted affects two types of costs: production cost and inventory holding cost. It is assumed that production costs have the fixed charge structure, where the cost of producing a lot of Q_n units at stage n is given by

$$C_n(Q_n) = \begin{cases} S_n + C_n Q_n, & \text{if } Q_n > 0 \\ 0, & \text{if } Q_n = 0 \end{cases}$$

The inventory holding costs per unit time at stage n are assumed to be proportional to the average inventory level, \bar{I}_n ; that is,

$$H_n(\bar{I}_n) = h_n \bar{I}_n$$

The objective may be stated as minimization of the total average cost per unit time for production and inventory.

A better understanding of the system is achieved by considering the first stage as a purchasing function to provide a material or component for use in the second, or production, stage. The procurement rate at the purchasing stage is infinite since it is assumed that the entire lot is delivered at one time. The average demand per unit time on the purchasing stage will be equal to the demand rate acting on the second stage, or finished goods, inventory. However, while the demand on the finished goods inventory occurs at a constant rate, the demand on the material inventory is discontinuous in time and depends on the production program at stage 2. It is assumed that there is a finite time required for the withdrawal of a unit of material from inventory until it is converted through production into a finished item. This interval is called the production lead time and is denoted by T_2 .

Figure 6 shows the system that has been described.

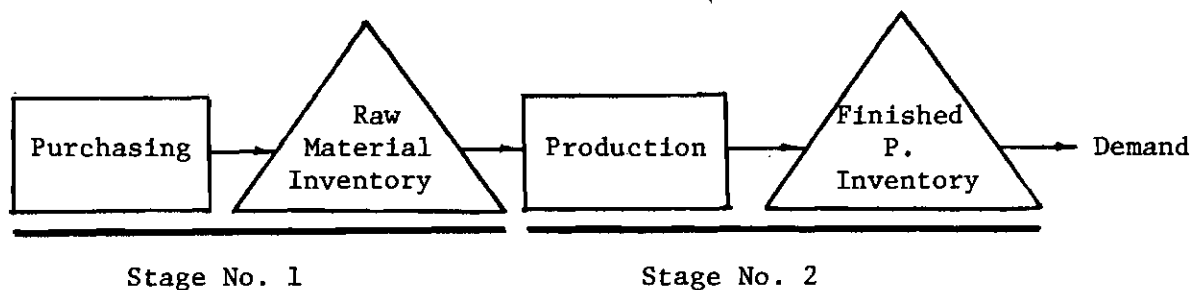


Figure 6. A Two-Level Serial System

The following notation will be needed:

- Q_1 = Purchase lot size
 Q_2 = Production lot size
 P_1 = Procurement rate (infinite)
 P_2 = Production rate (finite)
 D = Demand rate
 CT_n = Cycle time at stage n , $n = 1, 2$
 T_2 = Production lead time at stage 2
 t_2 = Time to produce a lot at stage 2
 h_n = Inventory cost per unit per unit time at stage n , $n = 1, 2$
 S_n = Fixed cost per lot, $n = 1, 2$
 T = Total cost
 \bar{U}_2 = Average unavailable inventory at stage 2
 \bar{W}_2 = Average work-in-process at the production stage
 \bar{V}_n = Average available inventory at stage n , $n = 1, 2$
 \bar{I}_n = Average total inventory at stage n , $n = 1, 2$

The models that will be analyzed do not allow shortages. Also, the production and consumption of a lot of finished product may or may not be simultaneous. The latter situation results in the creation of an unavailable inventory, consisting of units that have completed production, but that will not be available for satisfying demand until the entire lot is completed.

Two policy structures will be considered. The first requires that lots be produced or purchased at equally spaced time intervals. This is called a cyclical policy. The second requires a cyclical policy at the production stage, but permits material to be carried in inventory only during the time production takes place.

The Cyclical Policy Solution

For a cyclical policy to function efficiently without shortages, it appears rational that the cycle time at the first stage should be an integer multiple of the cycle time at the second stage; this implies

$$Q_1 = k_1 Q_2 \text{ where } k_1 = 1, 2, \dots$$

Many writers have assumed this policy in their derivation of multi-stage algorithms. Crowston, Wagner and Williams [5] prove for pure assembly systems (of which this is a special case) that an optimal set of lot sizes exists such that the lot size at each facility is a positive integer multiple of the lot size at its successor facility. Taha and Skeith [25] allow non-instantaneous production, delay between stages, and back-orders for the product of the final stage. Johnson and Montgomery [11] obtain average inventory equations for each stage as a function of the lot sizes. Those equations will be used in the present analysis.

The Simultaneous Production and Consumption Case

Figure 7 illustrates the integer multiple policy for the case $Q_1 = 3Q_2$ and where simultaneous production and consumption of the production lot occurs.

The following results for average inventory levels are obtained from Johnson and Montgomery [3, p. 157]:

$$\bar{I}_1 = \bar{V}_1 = Q_2/2(k_1 - 1 + D/P_2)$$

$$\bar{I}_2 = \bar{W}_2 + \bar{V}_2 = T_2 D + Q_2/2(1 - D/P_2)$$

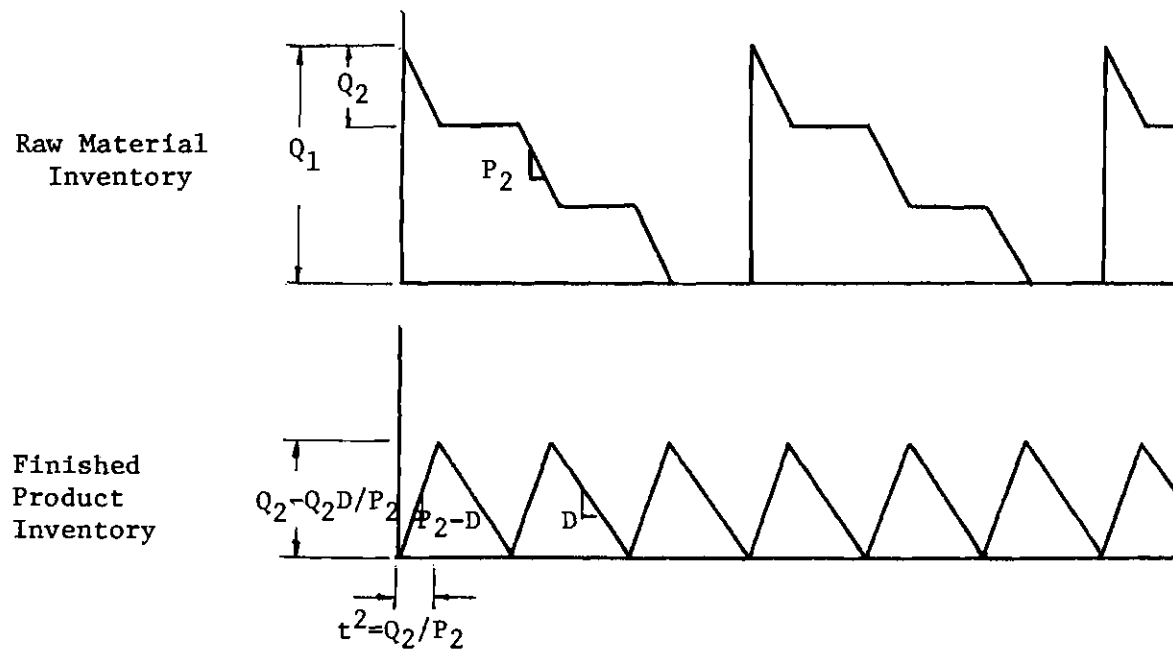


Figure 7. Inventories Under Integer Multiple Policy With Simultaneous Production and Consumption

The average cost per unit time would be

$$\bar{T}(k_1, Q_2) = \frac{S_1 D}{k_1 Q_2} + \frac{S_2 D}{Q_2} + h_1 \frac{Q_2}{2} \left[k_1 - 1 + \frac{D}{P_2} \right] + h_2 \left[T_2 D + \frac{Q_2}{2} \left[1 - \frac{D}{P_2} \right] \right]$$

where the variable cost of procurement and production has been omitted since it is independent of the production policy.

For a given k_1

$$\frac{\partial \bar{T}}{\partial Q_2} = 0 \quad \text{Then } Q_2^* = \left[\frac{2(S_1/k_1 + S_2)D}{h_1(k_1 - 1 + D/P_2) + h_2(1 - D/P_2)} \right]^{1/2}$$

and

$$T(k_1, Q_2^*) = \left[2(S_1/K_1 + S_2)D[h_1(k_1-1 + D/P_2) + h_2(1 - D/P_2)] \right]^{1/2} + h_2T_2D$$

These results give the conditional optimal production lot size and cost as a function of k_1 . To find the optimal value of k_1 , we calculate the first difference of $T^2(k_1, Q_2^*)$ as

$$\Delta_{k_1} T^2(k_1, Q_2^*) = 2D \left[S_2h_1 - \frac{S_1(h_2-h_1)(1-D/P_2)}{k_1(k_1+1)} \right]$$

The optimal k_1 satisfies

$$(k_1^*-1)k_1^* \leq \frac{S_1(h_2-h_1)(1-D/P_2)}{S_2h_1} \leq k_1^*(k_1^*+1)$$

Once k_1^* is obtained, the optimal lot size is found using the expression above.

The Nonsimultaneous Production and Consumption Case

The case of nonsimultaneous production and consumption is illustrated in Figure 8 for $Q_1 = 3Q_2$. Note that the entire production lot is completed before the lot is transferred to the available inventory.

Johnson and Montgomery [11] give the following expression for average inventories:

$$\bar{I}_1 = \bar{V}_1 = Q_2/2(k_1-1 + D/P_2)$$

$$\bar{I}_2 = \bar{W}_2 + \bar{U}_2 + \bar{V}_2 = T_2 D + \frac{Q_2 D}{2P_2} + \frac{Q_2}{2} = T_2 D + \frac{Q_2}{2}(1 + D/P_2)$$

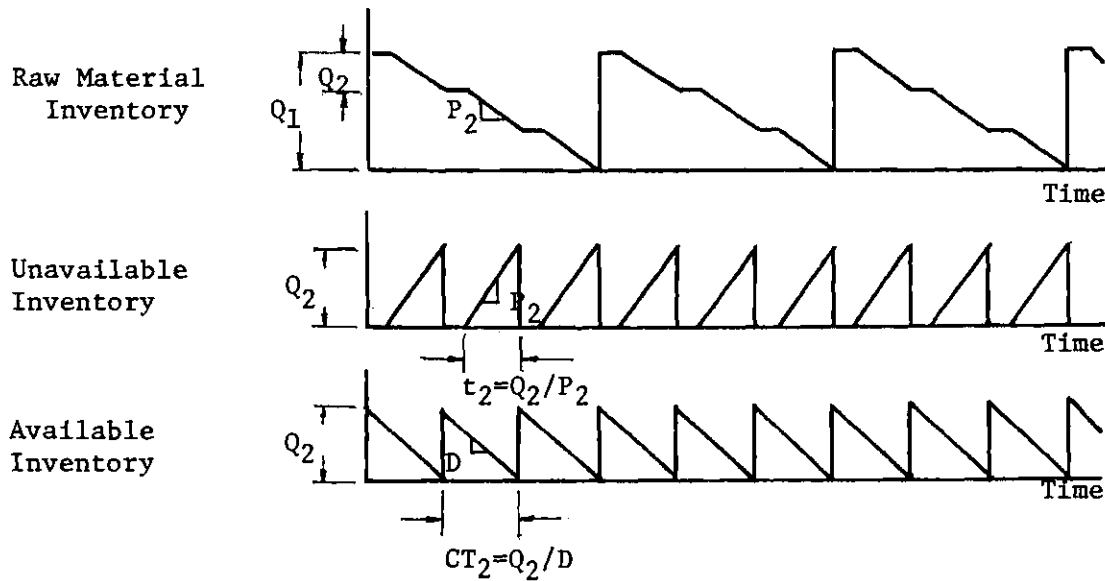


Figure 8. Inventories Under Integer Multiple Policy
With Nonsimultaneous Production and Consumption

The average cost per unit time would be

$$\bar{T}(k_1, Q_2) = \frac{S_1 D}{k_1 Q_2} + \frac{S_2 D}{Q_2} + h_1 \frac{Q_2}{2} (k_1 - 1 + D/P_2) + h_2 \left[T_2 D + \frac{Q_2}{2} (1 + D/P_2) \right]$$

For a given k_1

$$\frac{dT}{dQ_2} = 0 \rightarrow Q_2^* = \left[\frac{2(S_1/k_1 + S_2)D}{h_1(k_1 - 1 + D/P_2) + h_2(1 + D/P_2)} \right]^{1/2}$$

and

$$T(k_1, Q_2^*) = \left[2(S_1/k_1 + S_2)D[h_1(k_1-1 + D/P_2) + h_2(1 + D/P_2)] \right]^{1/2} + h_2T_2D$$

Differencing with respect to k_1 ,

$$\Delta T^2(k_1, Q_2^*) = 2D \left[S_2h_1 - \frac{S_1[h_2(1+D/P_2) - h_1(1-D/P_2)]}{k_1(k_1 + 1)} \right]$$

and then the optimal k_1 satisfies

$$(k_1^* - 1)k_1^* \leq \frac{S_1[h_2(1+D/P_2) - h_1(1-D/P_2)]}{S_2h_1} \leq k_1(k_1^* + 1)$$

The optimal lot size (Q_2^*) is then obtained using k_1^* .

A Deviation from the Cyclical Policy

In most research on multistage systems, the integer multiple policy is assumed to be the optimal structure. However, if we permit non-cyclical policies, this may not be the case. To show an exception, consider the purchasing-production system of the previous section (Figure 6), and suppose that material may be stocked only while production is in progress. Thus during each finite interval of production, $t_2 = Q_2/P_2$, a total material requirement of Q_2 units will be experienced at a uniform rate of P_2 . Schwarz [22] shows that for the finite horizon lot size problem with

uniform demand, the optimal policy is to use equal size lots. Thus the policy structure $Q_1 = Q_2/k_1$, $k_1 = 1, 2, \dots$, is used here. We call this the integer divisor policy.

The Simultaneous Production and Consumption Case

When simultaneous production and consumption of the production lot is possible, the integer divisor policy is as shown in Figure 9 (for $Q_1 = Q_2/3$).

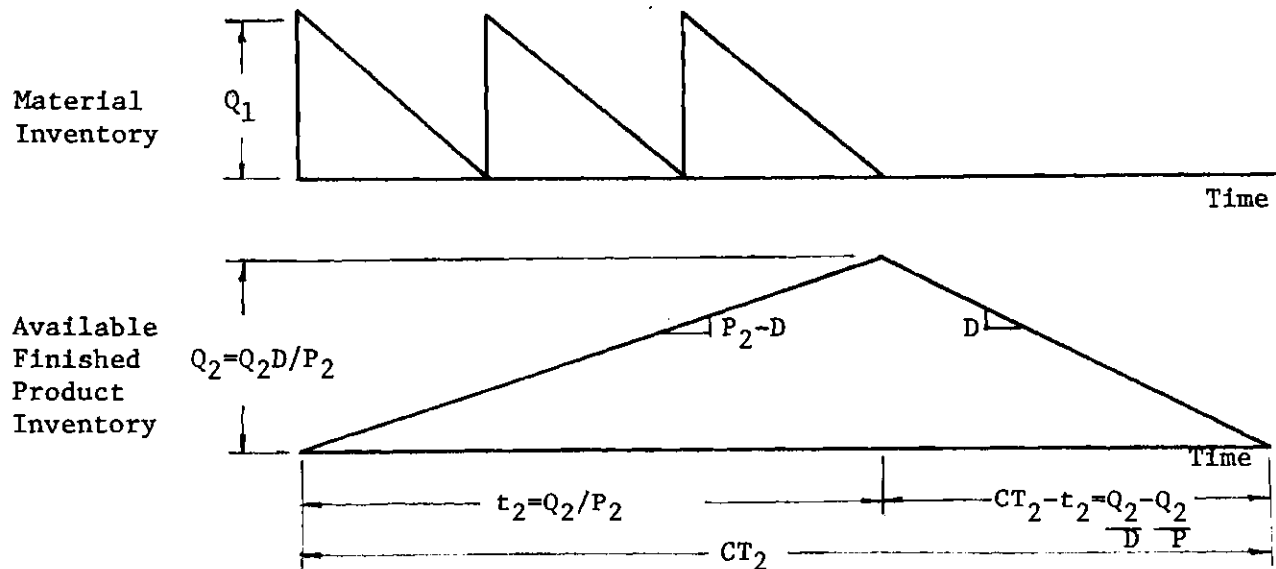


Figure 9. Inventories Under Integer Divisor Policy
With Simultaneous Production and Consumption

The average inventories may be calculated as

$$\bar{I}_1 = \frac{Q_1}{2} \frac{t_2}{CT_2} = \frac{Q_1}{2} \frac{Q_2/P_2}{Q_2/D} = \frac{Q_1}{2} \frac{D}{P_2} = \frac{Q_2 D}{2k_1 P_2}$$

$$\bar{V}_2 = Q_2/2(1-D/P_2)$$

$$\bar{I}_2 = \bar{W}_2 + \bar{V}_2 = T_2D + Q_2/2(1-D/P_2)$$

The average cost per unit would be

$$\bar{T}(k_1, Q_2) = (k_1S_1 + S_2)(D/Q_2) + Q_2/2[h_1D/k_1P_2 + h_2(1-D/P_2)] + h_2T_2D$$

where the variable cost of procurement and production has been omitted since it is independent of the production policy. For a given k_1 ,

$$\frac{\partial T}{\partial Q_2} = 0 \rightarrow Q_2^* = \left[\frac{2(k_1S_1 + S_2)D}{h_1D/k_1P_2 + h_2(1-D/P_2)} \right]^{1/2}$$

and

$$T(k_1, Q_2^*) = \left[2D(k_1S_1 + S_2)[h_1D/k_1P_2 + h_2(1-D/P_2)] \right]^{1/2} + h_2T_2D$$

These results give the conditional optimal production lot size and cost as a function of k_1 .

Differencing with respect to k_1 , we obtain

$$\Delta k_1 T^2(k_1, Q_2^*) = 2D \left[S_1 h_2 (1-D/P_2) - \frac{S_2 h_1 (D/P_2)}{k_1 (k_1+1)} \right]$$

Then the optimal k_1 satisfies the following inequality:

$$(k_1^* - 1)k_1^* \leq S_2 h_1 (D/P_2) \int S_1 h_2 (1 - D/P_2) \leq k_1^* (k_1^* + 1)$$

Using k_1^*, Q_2^* is then obtained.

The Nonsimultaneous Production and Consumption Case

When simultaneous production and consumption are not possible, the situation is as shown in Figure 10 (illustrated for $Q_1 = Q_2/3$).

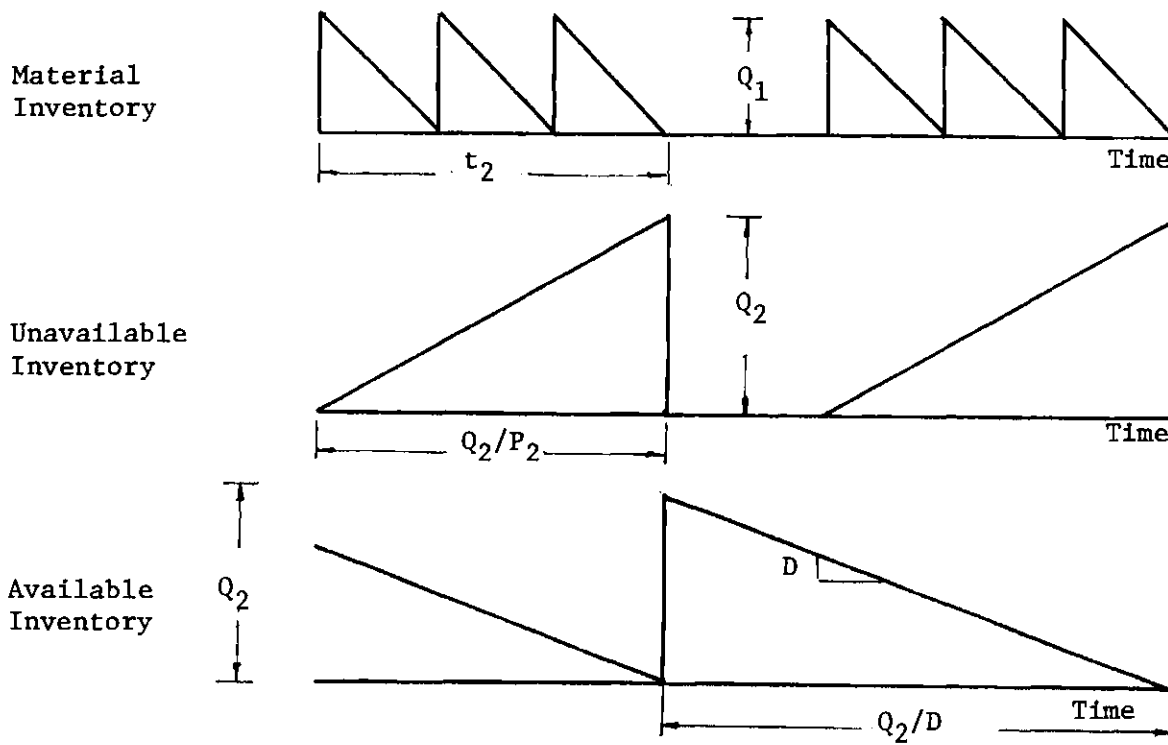


Figure 10. Inventories Under Integer Divisor Policy With Nonsimultaneous Production and Consumption

The average inventory levels are computed as

$$\bar{I}_1 = \frac{Q_1 t_2}{CT_2} = \frac{Q_2 D}{2k_1 P_2}$$

$$\bar{U}_2 = \frac{Q_2 t_2}{2 CT_2} = \frac{Q_2 D}{2P_2}$$

$$\bar{V}_2 = Q_2/2$$

$$\bar{I}_2 = \bar{U}_2 + \bar{V}_2 = \frac{Q_2 D}{2P_2} + \frac{Q_2}{2} = \frac{Q_2}{2} \left[1 + \frac{D}{P_2} \right]$$

The cost function is

$$\bar{T}(k_1, Q_2) = D/Q_2 [k_1 S_1 + S_2] + Q_2/2 [h_1 D/k_1 P_2 + h_2 (1+D/P_2)] + h_2 T_2 D$$

For given k_1 ,

$$\frac{\partial T}{\partial Q_2} = 0 \rightarrow Q_2^* = \left[\frac{2D(k_1 S_1 + S_2)}{h_1 D/k_1 P_2 + h_2 (1+D/P_2)} \right]^{1/2}$$

and

$$T(k_1, Q_2^*) = \left[2D(k_1 S_1 + S_2) [h_1 D/k_1 P_2 + h_2 (1+D/P_2)] \right]^{1/2} + h_2 T_2 D$$

These equations give the conditional optimal production lot size and cost as a function of k_1 .

By differencing, we obtain

$$\Delta k_1 T^2(k_1, Q_2^*) = 2D \left[S_1 h_2 (1+D/P_2) - \frac{S_2 h_1 (D/P_2)}{k_1 (k_1+1)} \right]$$

Then the optimal k_1 satisfies the following inequality:

$$k_1^* (k_1^* - 1) \leq S_2 h_1 (D/P_2) \int S_1 h_2 (1+D/P_2) \leq k_1^* (k_1^* + 1)$$

Using k_1^* , Q_2^* is then obtained from the expression above.

A Comparison Between Integer Multiple and Integer Divisor Policies

Computer subroutines were written for the solutions obtained in the previous sections for the four cases of integer multiple or integer divisor policy with simultaneous or nonsimultaneous production and consumption. These subroutines are shown in Appendix B. A program that generates problems, and using the subroutines, computes optimal lot sizes and costs under each policy was written and is also given in Appendix B. The policies are compared for different ratios of D/P_2 , S_1/S_2 and h_2/h_1 , to determine the effect of these parameters on the choice of policy structure.

The Simultaneous Production and Consumption Case

Using the information obtained by the program mentioned above, graphs were constructed for each of the S_1/S_2 ratios. In each graph there are curves corresponding to the seven levels of utilization ratios generated by the program; the horizontal axis is the h_2/h_1 ratio and vertical axis is difference (in dollars) of the integer multiple solution less the

integer divisor solution. Thus, below the horizontal axis the integer multiple policy is preferred and above the zero level, the preference changes and the integer divisor policy is desired.

To analyze the behavior of the preferences as the ratios are varied, first consider h_2/h_1 . As this ratio is increased, the curves fall denoting preference for the integer multiple policy. This is because if h_2/h_1 increases, then h_2 increases and the lot size decreases, and since h_1 is constant, Q_2 is reduced relative to Q_1 , and at some point the shift is to an integer multiple solution. Now as D/P_2 increases the curve is rising and tending to prefer integer divisor policies. This is because the greater the ratio, the greater the production lot size, so that for a constant Q_1 , Q_2 tends to be greater. Also observe that as D/P_2 increases, the slope of the curve increases, showing indifference to the h_2/h_1 ratio. Finally as S_1/S_2 increases, all the curves are moved down preferring the integer multiple policy. The reason is the greater the ratio, the greater S_1 for fixed S_2 , and the greater Q_1 relative to Q_2 .

The Nonsimultaneous Production and Consumption Case

In general the effect of the nonsimultaneous production and consumption has been a movement of all the curves toward the side of integer multiple policy preference, and a closer spread among the seven curves in each graph. This can be explained because of the greater finished product inventory, resulting from the addition of the unavailable inventory, which causes a reduction in Q_2 relative to Q_1 .

Again as in the simultaneous case, h_2/h_1 produces the same effect and it could be explained with the same arguments. On the other hand, the ratio D/P_2 now moves the curves downward. Let us observe the ratios that k_1^* has

to satisfy, the only difference between multiple and divisor policies is just a sign in the denominator, $-D/P_2$ and $+D/P_2$, respectively, so if D/P_2 increases the first ratio increases while the second decreases. Then k_1^* (multiple) tends to be greater than k_1^* (divisor). Looking at the cost equations, we see that increasing D/P_2 decreases the integer multiple solution's cost while increasing that of the integer divisor solution. Note also that increasing D/P_2 decreases the slope of the curves giving less elasticity to h_2/h_1 ratio. The S_1/S_2 ratio causes the same effects as in the simultaneous case; that is, moving all of the curves downward.

The graphs are presented in Appendix A.

Assembly Structure

Problem Definition

A multi-stage assembly system requires at each stage, or facility, inputs from a number of immediate predecessor stages, and supplies, in turn, one immediate successor. Specifically in the two-level system there is just one stage at level one and any number of stages at level two (see Figure 11). The case of constant demand over an infinite horizon, with instantaneous production, will be considered.

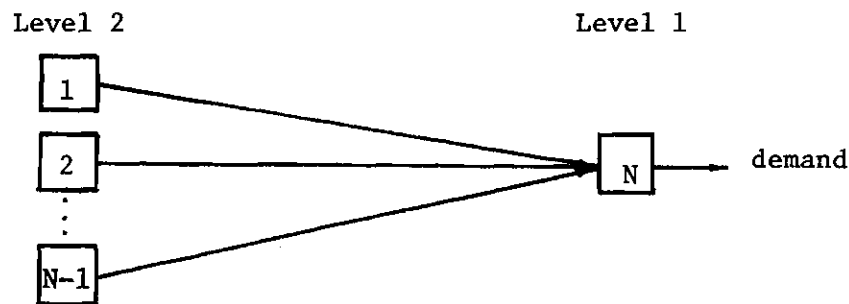


Figure 11. Two-Level Assembly Structure

The objective is the choice of a lot size for each facility so that the average total costs of production and inventory are minimized. At each stage, the production cost per lot is assumed to have the fixed charge form used in the previous section and inventory costs are assumed to be proportional to average inventory. Because the variable production cost is constant (independent of lot sizes), it will be omitted in the analysis to follow, and only set up costs will be considered.

The Echelon Stock Concept

Clark and Scarf [3] define echelon stock associated with a given stage in an assembly system as the quantity of the item produced at that stage that is still in the system, either at the stage in question or at some successor stage. This is in contrast to installation stock which may be defined as the inventory at a given stage (or installation). For multi-level assembly systems, there is an advantage to expressing inventory carrying costs in terms of echelon stock because the average echelon stock is a function only of the lot-size policy at the producing stage.

For a better understanding, consider the following analysis of the series structure of Figure 12:

Let I_n = total inventory at stage n , $n = 1, 2, \dots, N$ and

EI_n = echelon inventory at stage n , $n = 1, 2, \dots, N$

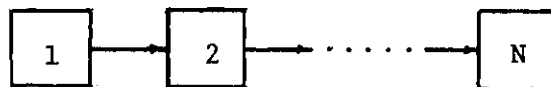


Figure 12. Series Structure

Then

$$EI_n = \sum_{J=n}^N I_J$$

and let h_n be the unit cost of carrying installation inventory at stage n and h'_n be the carrying cost per unit of echelon inventory, then

$$\begin{aligned} \text{Total Cost} &= \sum_{n=1}^N h'_n \bar{I}_n = \sum_{n=1}^N h_n \bar{EI}_n = \sum_{n=1}^N h_n \sum_{J=n}^N \bar{I}_J \\ &= h_1 (\bar{I}_1 + \bar{I}_2 + \dots + \bar{I}_{N-1} + \bar{I}_N) \\ &\quad + h_2 (\bar{I}_2 + \dots + \bar{I}_{N-1} + \bar{I}_N) \\ &\quad \cdot \\ &\quad \cdot \\ &\quad \cdot \\ &\quad + h_{N-1} (\bar{I}_{N-1} + \bar{I}_N) \\ &\quad + h_N (\bar{I}_N) \\ &= \sum_{n=1}^N \left[\sum_{J=1}^n h_J \right] \bar{I}_n \quad \text{and} \quad h'_n = \sum_{J=1}^n h_J \end{aligned}$$

Then $h'_i = h'_i - h'_{i-1} - 1$ can be considered as the additional unit carrying cost caused by processing at stage n .

Clark and Scarf [3] and Crowston, Wagner, and Williams show that the average echelon inventory at any stage n is $Q_n/2$, assuming instantaneous production (that is, $P_n = \infty$) and the integer multiple policy to be defined below.

To formulate a model based on echelon inventory, let

Q_n = Lot size at stage n , $n = 1, 2, \dots, N$,

N = Index of the first level stage,

S_n = Set up cost at stage n ,

h_n = Inventory holding cost per unit time per unit of stage n echelon inventory

D = Demand rate on stage N ,

$a(n)$ = Successor facility to stage n ,

$f_n(Q_n)$ = Cost of production and echelon inventory at stage n per unit time, and

T = Total cost per unit time.

We shall require that $Q_n = k_n Q_{a(n)}$, $k_n = 1, 2, \dots$, so that the lot size at stage n is an integer multiple of the policy at stage $a(n)$, the successor facility using stage n 's production. Crowston, Wagner, and Williams show this to be the optimal cyclical policy under relatively general conditions.

The cost relations defining the problem are the following:

$$f_n(Q_n) = \frac{S_n D}{Q_n} + h_n \frac{Q_n}{2}$$

$$T = \sum_{n=1}^N f_n(Q_n)$$

The problem is to choose Q_1, Q_2, \dots, Q_n to minimize T subject to the integer multiple requirement.

Literature Survey and Results

Crowston, Wagner, and Williams [5] prove the optimality of the integer

multiple policy where the lot size at any stage is an integer multiple of its successor. They also present graphically the difference between installation and echelon stock, and they give a dynamic programming algorithm for solving the assembly system problem.

Crowston, Wagner and Henshaw [4] present four different approaches for the assembly structure. The first is the dynamic programming algorithm given in [5]. The other three are heuristic rules which they compared with the dynamic programming solution. They ran six test problems and found that even though the dynamic programming obtained a lower cost in three problems by 0.07%, 0.16%, and 0.60% respectively, than the best heuristic, it consumed four times the computation time. They also show that the difference in computation time between the best and the worst heuristics is only 16% and conclude that their best heuristic is a good approach.

The third important paper treating the assembly structure with constant demand is by Schwarz and Schrage [24]. They suggest some adjustments for the parameters in order to use a branch and bound algorithm and present theoretical and empirical evidence for the near optimality of system myopic policies which are easy to understand, require less information, and are fast and easy to compute. Essentially the system myopic policies consider only two stages at a time in lot size determination.

Analysis

The following analysis presents seven different approaches: the dynamic programming described in [5], the three heuristic algorithms of [4], the branch and bound mentioned in [24], the myopic policies presented in the same paper, and a dynamic programming algorithm similar to that of [5], but with a slightly improved method of computation.

The Dynamic Programming Algorithm. For the two level system described by Figure 11, we require $Q_n = k_n Q_N$, $n = 1, 2, \dots, N-1$.

Let $T_n(Q_n)$ be the optimal cost at stage n and all prior stages when Q_n is given.

Then at level 2:

$$T_n(Q_n) = f_n(Q_n) = f_n(k_n Q_N)$$

and at level 1:

$$T_N(Q_N) = f_N(Q_N) + \sum_{n=1}^{N-1} \text{Minimum}_{k_n \in I} T_n(k_n Q_N)$$

The optimal cost is:

$$T^* = \text{Minimum}_{Q_N} \left[f_N(Q_N) + \sum_{n=1}^{N-1} \text{Minimum}_{k_n \in I} [f_n(k_n Q_N)] \right]$$

The solution proceeds from second stage to the first, computing $T_n(Q_n)$ for all possible Q_N , and then finding T .

To improve the computational efficiency of the algorithm, Crowston, Wagner, and Williams [5] propose the following procedure for bounding Q_N . For each stage, compute

$$Q_n^c = (2DS_n/h_n)^{1/2}$$

A lower bound for the cost at stage n is

$$LZ_n = (2S_n h_n D)^{1/2}$$

and therefore a lower bound for the complete system would be

$$LB = \sum_{n=1}^N LZ_n$$

An upper bound for system cost may be obtained from any feasible solution, such as one obtained by setting all k 's equal to 1 and $Q_N = Q_N^C$. The resulting upper bound would be

$$UB = \sum_{n=1}^N \left[\frac{DS_n}{Q_N} + \frac{Q_N h_n}{2} \right]$$

The upper and lower bounds on total system cost are used in the following manner to obtain bounds on Q_N . The maximum cost that can be associated with stage N is $UB - (LB - LZ_N)$. Since the cost at stage N is

$$\frac{DS_N}{Q_N} + \frac{Q_N h_N}{2}$$

which is convex in Q_N , solving the following quadratic equation will yield two positive values that can be used as bounds on the optimal lot size for stage N :

$$\frac{DS_N}{Q_N} + \frac{Q_N h_N}{2} = UB - (LB - LZ_N).$$

An Alternative Solution Procedure. The algorithm presented above suggests that the k-value for each stage at level 2 is found by searching on its possible range until the minimum cost is obtained.

An alternative approach to find k_n^* is to assume k_n is continuous and differentiate

$$T_n(k_n Q_n) = S_n D / k_n Q_n + k_n Q_n h_n / 2$$

and equate the derivative to zero to obtain

$$k_n^* = \left[2S_n D / Q_n^2 h_n \right]^{1/2}$$

Secondly, to limit the range of search over Q_N to find the minimum of $T_N(Q_N)$, a different bounding procedure was used. Q_N was limited to the range

$$\min_n \left| Q_n^c \right| \leq Q_N \leq \max_n \left| Q_n^c \right|$$

where

$$Q_n^c = \left[2S_n D / h_n \right]^{1/2}$$

Single Pass Heuristic Algorithm. Substituting $Q_n = k_n Q_n$, where $k_n = 1$ in the total cost equation, then

$$T = \sum_{n=1}^N (DS_n/k_n Q_N + k_n Q_N h_n/2)$$

Then requiring $\partial T/\partial Q_N = 0$ yields for a given set of k_n

$$Q_N^* = \left[2D \sum_{n=1}^N (S_n/k_n) / \sum_{n=1}^N k_n h_n \right]^{1/2}$$

Steps in the routine:

1. Set all $k_n = 1$ and compute Q_N , T_N , and do Best = T_N , set $n=N$.
2. Subtract one unit from n , and if n equals zero, stop.
3. Increase one unit the value of k_n .
4. Compute Q_N and T_N with that set of k 's.
5. Compare T_N with Best and if T_N is greater than or equal to Best subtract one unit from k_n and go to step 2.
6. Do Best equal to T_N and go to 3.

Multiple Pass Heuristic Algorithm. This algorithm is similar to the single pass. The differences are that in multiple pass the answers (k 's) of the single pass are used as the initial set of k 's, until no improvement is registered. This procedure also allows the reduction of k_n , that is after step 5, if T_N was greater than or equal to Best, then the algorithm will search for lower values of k_n , in other words.

5. Compare T_N with Best, and if T_N is greater than or equal to Best, subtract one unit from k_n and go to 7.

6. Do Best equal to T_N and go to 3.
7. Subtract one unit from k_n .
8. Compute Q_N and T_N with that set of k 's.
9. Compare T_N with Best, and if T_N is greater than or equal to Best, add one unit to k_n and go to 2.
10. Do Best equal to T_N and go to 7.

Modified Multiple Pass Heuristic Algorithm. This final heuristic is identical to the multiple pass heuristic except that it does not begin with $k_n = 1$. Instead for each stage $n = 1, 2, \dots, N$ the "unconstrained" lot size is calculated (the classical EOQ Model is used), and beginning with Q_N rounded to the nearest integer value, the lot sizes of all predecessors are adjusted to the integer multiple of Q_N nearest their "unconstrained" lot size.

The Branch and Bound Algorithm. Consider Q_n as the optimal individual lot size, that is computed using the classical lot size determinations for a single stage.

$$Q_n = (2S_n D / h_n)^{1/2}$$

Now, applying this formula to our total problem, minimize

$$\sum_{n=1}^N (S_n / k_n) / Q_N + (Q_N / 2) \sum_{n=1}^N k_n h_n$$

yields

$$Q_N = \left[\frac{2D \sum_{n=1}^N (S_n / k_n)}{\sum_{n=1}^N k_n h_n} \right]^{1/2}$$

which is exactly the same formula derived in the single pass algorithm.

Note that this corresponds to the standard EOQ where $\sum_{n=1}^N S_n/k_n$ is the average system setup cost per lot and $\sum_{n=1}^N k_n h_n$ is a composite system holding cost.

The search begins by solving:

minimize

$$\sum_{n=1}^N (S_n D/Q_n + h_n Q_n/2) \quad (2.1)$$

$$\text{Subject to } Q_n \geq Q_{a(n)} \text{ for } n = 1, 2, \dots, N-1 \quad (2.2)$$

with the next procedure:

(a) Compute Q_n for $n = 1, 2, \dots, N$

(b) if all constraints (2.2) are satisfied, i.e., $Q_n \geq Q_{a(n)}$ for all n , then stop; else find the set such that there is a n for which

$Q_n < Q_N$. Find $J \in (1, 2, \dots, N-1)$ that minimizes Q_n and then modify (2.1)

- (2.2) in the following fashion:

$$S_n \leftarrow S_n + S_j$$

$$h_n \leftarrow h_n + h_j$$

$$S_j \leftarrow 0$$

$$h_j \leftarrow 0$$

if this solution satisfies the next constraints

$$Q_n = k_n Q_N \text{ for } n = 1, \dots, N \quad (2.3)$$

$$k_n \geq 1 \text{ and integer} \quad (2.4)$$

That is if $k_n = Q_n/Q_N$ is integer for $J = 1, 2, \dots, N-1$, the optimal

policy is at hand, otherwise a noninteger k_n is chosen for branching.

System Myopic Policies. System myopic policies optimize with respect to any two consecutive stages and ignore other multi-stage interaction effects. The system myopic policy we chose to investigate determines the k_n values for problem (2.1) - (2.4) by considering each stage n and its $a(n) = N$ as a two-stage system. Schwarz [24] has shown that the optimal integer $k_n = Q_n/Q_N$ for such systems is the smallest integer k_n satisfying

$$k_n(k_n + 1) \geq M_n$$

where M_n is the "Myopia Ratio" defined as

$$M_n = (S_n h_N / S_N h_n)$$

Results

In order to compare the alternative algorithms, seven computer programs were written and tested with 10 different problems.

Those programs are given in Appendix B and correspond to the algorithms in the following way:

<u>Program Name</u>	<u>Approach</u>
THE 1	Dynamic Programming
THE 2	Single Pass
THE 3	Multiple Pass
THE 4	Modified Multiple Pass
THE 5	Branch and Bound
THE 6	Myopic Policies
THE 12	Alternate Dynamic Programming

The problems are presented in Table 2 of Appendix A, and as it can

be observed they represent a variety of cases: drastic changes in set-up costs, with inventory constant costs and vice versa; smooth changes in set up costs with constant inventory costs and vice versa; increasing functions in both set up costs and inventory costs; and increasing set up costs with decreasing inventory costs.

A summary of all the results of the seven programs is presented in Table 1. The asterisks in the table mean that the program dimensions were not adequate to solve those (three) problems. A brief summary can be presented as follows:

	Times it was the best solution	Percentage over the best time
Dynamic Programming	3	90.38
Single Pass	4	64.18
Multiple Pass	8	72.71
Modified Multiple Pass	9	96.74
Branch and Bound	7	287.28
Myopic Policies	2	0
Alternate Dynamic Programming	4	75.50

These results suggest that the heuristics (Multiple and Modified Multiple Pass) are the best algorithms to calculate lot sizes in assembly structures. Branch and Bound has good results but it takes too much time; even though the Alternate Dynamic Programming is not the best one, it seems to be better than the other two policies (Dynamic Programming and Myopic Policies) and equal to Single Pass. For the case of smooth changes in set up cost, constant inventory cost and low value in set up cost at stage N (problem 5), it yielded the best policy.

Dynamic Programming performs well when the inventory carrying cost at stage N is greater than or equal to the highest inventory cost at

Table 1. Results For Assembly Structure

Problem	Dynamic Programming (THE 1)	Single Pass (THE 2)	Multiple Pass (THE 3)	Modified Multiple Pass (THE 4)	Branch and Bound (THE 5)	Myopic Policies (THE 6)	Author's Approach (THE 12)	Best Solution
1	†6877.50 128 1.29	7652.89 437 1.065	7195.48 303 1,147	6877.50 128 1,281	6877.50 128 2,417	6877.50 128 .675	**	THE 1, THE 4, THE 5, THE 6
2	11329.17 1265 1.211	9590.36 590 1.072	9590.36 590 1,114	9590.36 590 1.237	9590.36 590 2,707	10436.94 796 .658	11329.17 1265 1.191	THE 2, THE 3, THE 4, THE 5
3	*	10646.67 111 1,074	10557.78 101 1,089	10557.78 101 1,220	10641,71 109 2,482	12939.97 147 .655	38443,07 894 1,155	THE 3 THE 4
4	11738.89 30 1.244	11878.31 96 1.027	11738.89 90 1,106	11738,89 90 1,260	11738.89 90 2,460	11739.18 91 .637	11738.89 90 1.126	THE 1, THE 3, THE 4, THE 5, THE 12
5	2588.45 90 1.241	2645.75 212 1,088	2645.75 212 1,086	2587.23 122 1.325	2627.98 162 2,463	2588,46 130 .658	2586.83 94 1.070	THE 12
6	2899.49 283 1.18	2828.43 226 1.035	2828,43 226 1,096	2828,43 226 1,234	2828.43 226 2,516	2829.95 219 .689	2899.45 283 1.163	THE 2, THE 3 THE 4, THE 5
7	6259.07 894 1.168	4695.74 383 1.158	4695.74 383 1,158	4695.74 383 1,320	4695.74 383 2,502	4800.00 400 .643	6259.07 894 1,130	THE 2, THE 3 THE 4, THE 5

preceding stages.

The reason dynamic programming did not always obtain the best solution as expected is that the methods of bounding Q_N programmed into THE 1 and THE 12 excluded the optimal Q_N in certain of the problems. For example, in problem 7, the best solution was obtained with $Q_N = 383$, while the lower bound computed in the dynamic programming solution was 894. The bounding methods used were modifications of the previously described procedure of Crowston, et al. Obviously their approach, while requiring more computation time, is to be preferred over the methods tried here.

Single Pass gives fair results for all the cases, especially for high values of set up cost at stage N. Multiple Pass yielded very good results, failing primarily for low values of set up cost at stage N. Modified Multiple Pass, the best one, being 90 percent of the time optimal, failing once by insignificant difference. Branch and Bound was optimal seven times and for the three remainder problems it was only 1.35 percent over the optimal cost. The great disadvantage of this algorithm is the computational time that it requires, even though Myopic Policies give an optimal solution just two times. The costs achieved by this algorithm were only an average of five percent above the optimal.

Arborescence Structure

Problem Definition

Let us now consider the case where there are any number of stages at level one and just one at level two. Such systems are encountered frequently; for example, a group of company-owned gasoline service stations supplied by a single company-owned bulk storage facility, or a

Table 1. Continued

Problem	Dynamic Programming (THE 1)	Single Pass (THE 2)	Multiple Pass (THE 3)	Modified Multiple Pass (THE 4)	Branch and Bound (THE 5)	Myopic Policies (THE 6)	Author's Approach (THE 12)	Best Solution
8	5041.50 331 1.230	5061.29 349 1.027	5041.50 331 1.096	5041.50 331 1.293	5041.50 331 2.427	5045.68 312 .619	5041.50 331 1.093	THE 1, THE 3, THE 4, THE 5, THE 12
9	20712.98 634 1.296	20712.92 632 1.125	20712.92 632 1.090	20712.92 632 1.261	20712.92 632 2.535	20712.92 632 .589	20712.92 632 1.130	THE 2, THE 3, THE 4, THE 5, THE 6, THE 12
10	17672.67 632 1.197	16151.38 376 1.022	15925.24 346 1.154	15925.24 346 1.261	16206.49 381 2.467	16730.30 411 .627	***	THE 3 THE 4
Time Averages	1.228	1.059	1.114	1.269	2.498	.645	1.132	

†

A
B
C

- A - Cost of production program (dollars)
- B - Lot size of stage 10 (units)
- C - Computer time used in the problem (seconds)

Table 2. Data For Assembly Problems

Problem	1			2			3			4			5		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage 1	10	1.25		10	1.25		100	.25		100	.25		10	1.25	
2	100	1.25		100	1.25		100	2.5		100	2.5		15	1.25	
3	1000	1.25		1000	1.25		100	25		100	25		20	1.25	
4	10	1.25		10	1.25		100	.25		100	.25		25	1.25	
5	100	1.25		100	1.25		100	2.5		100	2.5		30	1.25	
6	1000	1.25		1000	1.25		100	25		100	2.5		35	1.25	
7	10	1.25		10	1.25		100	.25		100	.25		40	1.25	
8	100	1.25		100	1.25		100	2.5		100	2.5		45	1.25	
9	1000	1.25		1000	1.25		100	25		100	25		50	1.25	
10	10	1.25	1000	1000	1.25	1000	100	.25	1000	100	25	1000	10	1.25	1000

S - Set up cost (dollars)

h - Inventory carrying cost (dollars)

D - Demand (units)

Table 2. Continued

Problem	6			7			8			9			10		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage 1	10	1.25		100	.25		100	.25		50	.25		50	1.00	
2	15	1.25		100	.50		100	.50		100	.50		100	6.00	
3	20	1.25		100	.75		100	.75		200	1.00		200	5.00	
4	25	1.25		100	1.00		100	1.00		300	1.50		300	4.00	
5	30	1.25		100	1.25		100	1.25		500	2.50		500	2.50	
6	35	1.25		100	1.50		100	1.50		800	4.00		800	1.50	
7	40	1.25		100	1.75		100	1.75		1000	5.00		1000	1.00	
8	45	1.25		100	2.00		100	2.00		1200	6.00		1200	.50	
9	50	1.25		100	2.25		100	2.25		1400	7.00		1400	.25	
10	50	1.25	1000	100	.25	1000	100	2.25	1000	1000	5.00	1000	1000	5.00	1000

chain of discount department stores supplied by a single regional warehouse. Figure 13 illustrates the two-level arborescence structure. The demand rates on the level one stages are known and constant.

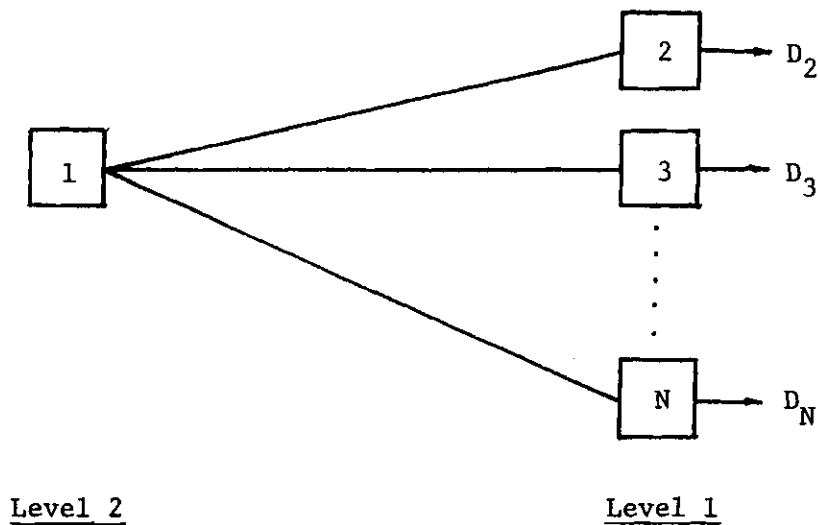


Figure 13. Arborescence Structure In Two Levels

The objective is to determine lot size policies at each stage in order to minimize the average cost per unit time for the system as a whole.

It is clear that the demand of the stage 1 will be the summation of the level 1 stages' demands, that is,

$$D_1 = \sum_{n=2}^N D_n$$

it is assumed that stage 1 receives material instantaneously and that it delivers to any facility at level 1, instantaneously. Backlogging is not permitted at any stage and costs of production and inventory have the same structure as in the previous section.

The total cost equation is therefore,

$$T = \sum_{n=1}^N (S_n D_n / Q_n + h_n Q_n / 2)$$

the same as in the assembly case. The goal is to choose Q_1, Q_2, \dots, Q_N to minimize T .

Literature Survey and Results

There are two relevant papers in the literature that treat this problem. Schwarz [23] considers the case of one warehouse and several retailers, develops equations for cycle times, and suggests a heuristic methodology to solve the problem. He does not show numerical results. Graves and Schwarz [10] provide a more general and comprehensive treatment of the problem, giving a branch and bound algorithm, and also a set of myopic single cycle policies. Finally they present an improvement procedure using the myopic policies. They give computational results, using 500 problems with 2, 3, 5, and 10 retailers. Their results are reproduced in Table 3; and in order to understand their meaning, define:

$$E=100 \frac{(\text{Cost of Heuristic Policy}) - (\text{Cost of Branch and Bound Algorithm})}{(\text{Cost of Branch and Bound Algorithm})}$$

It may be observed that as the number of stages increases, the heuristics turn out to be less accurate. However, they still perform very well since in the worst of the cases (10 retailers), the myopic policy averages just 1.74 percent over the optimal cost, and the improved myopic is 0.05 percent over optimal. In general, they show the convenience of using simple heuristics rather than complicated algorithms.

Graves and Schwarz also give the following characteristics of

Table 3. Heuristic Results

Number of Retailers	Policy	Number of Times Optimum	\bar{E}^*	σ_E	E_{Max}
2	Myopic	492	0.01%	0.12%	1.9%
	Improved Myopic	492	0.01%	0.12%	1.9%
3	Myopic	463	0.07%	0.42%	4.6%
	Improved Myopic	467	0.05%	0.32%	3.1%
5	Myopic	399	0.38%	1.12%	8.0%
	Improved Myopic	451	0.05%	0.22%	2.0%
10	Myopic	221	1.74%	2.85%	15.2%
	Improved Myopic	429	0.05%	0.23%	3.2%

*Variable defined in Section B of Chapter II.

optimal policies for arborescent systems (stated here for two-level systems):

1. Stage n produces (orders) only when its inventory is zero.
2. Stage 1 produces only when at least one of its successor stages ($n = 2, 3, \dots, N$) produces.
3. For each stage $n = 2, 3, \dots, N$ all lot sizes produced in the time interval between successive production runs at stage 1 are equal.

They also prove that the optimal stationary policy for the two-level system has the following structure, for $n = 2, 3, \dots, N$:

$$\frac{Q_1}{D_1} = \frac{k_n Q_n}{D_n}, \quad k_n = 1, 2, \dots$$

where $D_1 = \sum_{n=1}^N D_n$. Thus the problem is to find the optimal Q_1 and the integer variables k_2, k_3, \dots, k_N .

Analysis

As for the assembly case we will also develop here the different approaches mentioned in the literature survey, that is the heuristic procedure suggested in [23] and the branch and bound and the myopic policies given in [10], as well as a dynamic programming approach developed in this research.

Heuristic Method of Schwarz [23]. Define t_n as the cycle time at stage n , so

$$t_n = Q_n / D_n$$

Let $f_1(t)$ be the average cost per unit time for stage 1 if a cycle time of

t is used. Then,

$$f_1(t) = S_1/t + h_1 D_1 t/2$$

This cost is minimized for a cycle time of

$$t_1^* = [2S_1/(D_1 h_1)]^{1/2}$$

and the minimum cost is

$$f_1(t_1^*) = [2S_1 h_1 D_1]^{1/2}$$

Let $f_n^k(t)$ = Average cost per unit time for stage n , $n = 2, 3, \dots, N$, given a single cycle policy of length t at stage 1 and k deliveries from stage 1 to stage n during the cycle.

This cost is

$$f_n^k(t) = kS_n/t + h_n D_n t/2k$$

Since $f_n^k(t)$ is convex for all $t > 0$, it has a unique minimum at

$$t^*(k, n) = k[2S_n/(D_n h_n)]^{1/2}$$

Then

$$f_n^k(t^*(k, n)) = [2S_n h_n D_n]^{1/2}$$

The solution algorithm proposed by Schwarz is as follows:

1. Compute t_1^* and $t_1^*(1,n)$, the first local minimum of $f_n^1(t)$ for $n = 2, \dots, N$.
2. If $t_1^* \geq \max [t^*(1,n)]$ set $t^* = t_1^*$, go to (4); otherwise go to (3).
3. If $t_1^* < \max [t^*(1,n)]$ then let $J = [n | t_1^* < t_1^*(1,n)]$. Let t^* be the value which minimizes
$$\left[f_1(t) + \sum_J f_n^1(t) \right], \text{ that is } t^* = \left[2(S_1 + \sum_J S_n) / h_1 D_1 + \sum_J h_n D_n \right]^{1/2}$$
 Go to (4).
4. Determine k_n^* , $n = 2, \dots, N$, the value of k_n that minimizes $f_n^k(t^*)$ for the chosen cycle length.

The Branch and Bound Algorithm. Graves and Schwarz [10] prove that the problem can be defined as follows:

$$\text{MIN } Z = \sum_{n=1}^N \left[\frac{S_n D_n}{Q_n} + \frac{h_n Q_n}{2} \right]$$

subject to

$$\text{constant} = \frac{Q_1}{D_1} = \frac{k_n Q_n}{D_n}$$

$$\text{and } k_n \geq 1, \text{ integer}$$

From this, it follows that the problem may be restated as

$$\text{MIN } Z = \sum_{n=1}^N \left[\frac{k_n S_n D_1}{Q_1} + \frac{h_n D_n Q_1}{2k_n D_1} \right] \quad (2.5)$$

where

$$k_n \geq 1, \text{ integer, } n = 2, 3, \dots, N \text{ and } k_1 = 1 \quad (2.6)$$

Solving for the optimal Q_1 , given k_2, k_3, \dots, k_N .

$$Q_1^*(k_2, k_3, \dots, k_N) = \left[\frac{2D_1 \sum_{n=1}^N k_n S_n}{\sum_{n=1}^N \frac{h_n D_n}{k_n D_1}} \right]^{1/2}, \text{ where } k_1 = 1 \quad (2.7)$$

When branching at level n we set k_n for that level equal to some integer (1, 2, 3, ...). For a set of specified k_1, k_2, \dots, k_n ($n > 1$) values, problem (2.5) - (2.6) becomes the following minimization problem for the remaining k_J values, $J = n+1, \dots, N$:

$$\text{MIN } Z = \sum_{J=n+1}^N \left[\frac{k_J S_J D_1}{Q_1} + \frac{h_J D_J Q_1}{2k_J D_1} \right] + \frac{\hat{S}_1 D_1}{Q_1} + \frac{\hat{h}_1 Q_1}{2} \quad (2.8)$$

subject to

$$k_J \geq 1, \text{ integer, } J = n+1, \dots, N \quad (2.9)$$

$$\hat{S}_1 = S_1 + \sum_{J=1}^n k_J S_J \quad (2.10)$$

$$\hat{h}_1 = h_1 + \sum_{J=1}^n \frac{h_J D_J}{k_J D_1} \quad (2.11)$$

Thus as is clear from (2.10) and (2.11), branching at level n corresponds to "collapsing" stage n 's order and holding costs into their respective costs at stage 1. The lower bound on (2.8) - (2.9) for a given branch at level n is determined by ignoring the integrality constraint (2.9) and optimizing (2.8). This yields lower bound LB, where

$$LB = \sum_{J=n+1}^N [2S_J D_J h_J]^{1/2} + [2\hat{S}_1 D_1 \hat{h}_1]^{1/2} \quad (2.12)$$

for $n = 2, 3, \dots, N$, LB is the cost for the specified single cycle policy.

The search begins by generating a feasible solution by sequentially setting k_n at stage n . Branching then begins at stage 2 and proceeds until all k_2 values have been enumerated, at which time branching begins at stage 3, etc.

System Myopic Single Cycle Policies. Arborescence structure is characterized by one stage at level 2 which has relations with $N-1$ stages at level 1. Myopic policies optimize individually those relations, focusing on two stages at a time. Then for a given pair of stages, the objective function would be

$$\text{minimize} \left[\frac{S_1 D_1}{Q_1} + \frac{h_1 Q_1}{2} + \frac{S_n D_n}{Q_n} + \frac{h_n Q_n}{2} \right]$$

subject to

$$\frac{Q_1}{D_1} = \frac{k_n Q_n}{D_n}$$

$$D_1 = \sum_{n=2}^N D_n$$

$$k_n \geq 1, \text{ integer}$$

Substituting $Q_n = Q_1 D_1 / k_n D_n$ and optimizing with respect to Q_1 for fixed k_n , we obtain:

$$Q_1^*(k_n) = \left[\frac{2k_n D_n (S_1 D_1^2 + S_n D_n^2 k_n)}{D_1 (h_1 k_n D_n + h_n D_1)} \right]^{1/2}$$

This is substituted on cost equation and by differencing, the system myopic k_n^M value is found as the smallest integer k_n satisfying

$$k_n(k_n + 1) \geq S_1 (D_n/D_1) h_n / S_n h_1$$

Given the k_n , $n = 2, 3, \dots, N$, $Q_1^*(k_2, \dots, k_N)$ is determined as in (2.7). The corresponding Q_n values, $Q_n^*(k_n)$, $n = 2, 3, \dots, N$, are determined.

$$Q_n^*(k_n^M) = \frac{D_n}{k_n^M D_1} Q_1^*(k_2^M, \dots, k_N^M)$$

Improvement in Myopic Policies. Using the $Q_1^*(k_n^M)$ as determined in (2.7), we may determine new k_n values as follows:

$$k_n(k_n + 1) \geq (h_n D_n Q_1^2) / (2S_n D_1^2)$$

These k_n values may in turn be used to compute a new $Q_1^*(k_n)$. This process may be repeated until the k_n values converge or until cost improvements no longer justify further iteration.

A Dynamic Programming Approach. Using the property of the optimal single cycle solution,

$$Q_n = \frac{Q_1 D_n}{k_n D_1}$$

and making the same assumptions for the limits of stage 1 lot size as in the assembly structure, we define the following recursive procedure.

The process starts at stage n and finishes at stage 1, using the next recursive equations,

$$T_n(Q_1) = \min_{k_n} \left[\frac{S_n k_n D_1}{Q_1} + \frac{Q_1 D_n h_n}{2k_n D_1} + T_{n+1}(Q_1) \right]$$

$$T^* = \text{optimal } T = \min_{Q_1} T_1(Q_1)$$

To determine the value of k_n that minimizes $T_n(Q_1)$, we approximate k_n as continuous and, noting that $T_{n+1}(Q_1)$ is independent of k_n , differentiate to obtain,

$$k_n^*(Q_1) = \left[Q_1^* D_n h_n / 2D_1^2 S_n \right]^{1/2}$$

$T_1(Q_1)$ is computed for all integer values of Q_1 within the predetermined range.

Results

Computer programs were for each one of the methodologies described above, the corresponding names are:

<u>Computer Name</u>	<u>Approach</u>
THE 8	Schwarz's Method
THE 9	Branch and Bound
THE 10	Myopic Policies
THE 11	Modified Myopic Policies
THE 14	Dynamic Programming Approach

The ten problems used to test these algorithms are presented in Table 4. The average demand over the nine first level stages is the same for all problems.

Table 5-A contains the results. The asterisk under the branch and bound title means that those programs exceeded the time limit because of the great number of branches searched. A summary of the results is the following:

	Times it was the best solution	Percentage over the best time
Schwarz's Method	-	27
Branch and Bound	6	190.83
Myopic Policies	1	-
Modified Myopic Policies	6	27.33
Dynamic Programming Approach	-	172.33

Schwarz's Method definitely was inferior in four of the cases (Problem Numbers 1, 2, 6, 7), being far from optimal for problems with

Table 4. Problems For Arborescence Structure

Problem	1			2			3			4			5		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage															
1	10	1.25		10	1.25		100	.25		100	.25		10	1.25	
2	100	1.25	1000	100	1.25	1000	100	2.5	1000	100	2.5	1000	15	1.25	1000
3	1000	1.25	1000	1000	1.25	1000	100	25	1000	100	25	1000	20	1.25	1000
4	10	1.25	1000	10	1.25	1000	100	.25	1000	100	.25	1000	25	1.25	1000
5	100	1.25	1000	100	1.25	1000	100	2.5	1000	100	2.5	1000	30	1.25	1000
6	1000	1.25	1000	1000	1.25	1000	100	25	1000	100	25	1000	35	1.25	1000
7	10	1.25	1000	10	1.25	1000	100	.25	1000	100	.25	1000	40	1.25	1000
8	100	1.25	1000	100	1.25	1000	100	2.5	1000	100	2.5	1000	45	1.25	1000
9	1000	1.25	1000	1000	1.25	1000	100	25	1000	100	25	1000	50	1.25	1000
10	10	1.25	1000	10	1.25	1000	100	.25	1000	100		1000	10	1.25	1000

S - Set up cost (dollars)

h - Inventory carrying cost (dollars)

D - Demand (units)

Table 4. Continued

Problem	6			7			8			9			10		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage 1	10	1.25		10	1.25		100	.25		100	.25		10	1.25	
2	100	1.25	200	100	1.25	200	100	2.5	200	100	2.5	200	15	1.25	200
3	1000	1.25	400	1000	1.25	400	100	25	400	100	25	400	20	1.25	400
4	10	1.25	600	10	1.25	600	100	.25	600	100	.25	600	25	1.25	600
5	100	1.25	800	100	1.25	800	100	2.5	800	100	2.5	800	30	1.25	800
6	1000	1.25	1000	1000	1.25	1000	100	25	1000	100	25	1000	35	1.25	1000
7	10	1.25	1200	10	1.25	1200	100	.25	1200	100	.25	1200	40	1.25	1200
8	100	1.25	1400	100	1.25	1400	100	2.5	1400	100	2.5	1400	45	1.25	1400
9	1000	1.25	1600	1000	1.25	1600	100	25	1600	100	25	1600	50	1.25	1600
10	1000	1.25	1800	1000	1.25	1800	100	.25	1800	100	25	1800	10	1.25	1800

Table 5-A. Results For Arborescence Structure

Problem	Schwarz's Method (The 8)	Branch and Bound (The 9)	Myopic Policies (The 10)	Modified Myopic Policies (The 11)	Author's Approach (The 14)	Best Solution
1	† 79689.40 379 .782	*	12259.69 4903 .578	11512.60 5878 .758	61712.56 492 1.578	The 11
2	103169.31 379 .761	*	13958.87 5583 .612	13244.02 6428 .797	79822.32 492 1.636	The 11
3	10658.59 2683 .755	10382.68 4854 1.706	10658.33 2702 .621	10658.33 2702 .779	10542.77 3244 1.598	The 9
4	12534.40 2683 .770	12365.27 4803 1.669	12525.97 2586 .605	12525.97 2586 .771	12453.92 3180 1.610	The 9
5	7115.12 379 .770	3549.65 1419 1.703	3549.65 1419 .607	3549.65 1419 .766	5736.95 492 1.582	The 9 The 10 The 11
6	79689.40 379 .755	*	12259.69 4903 .613	11315.84 5678 .747	61712.56 492 1.636	The 11
7	103169.31 379 .758	*	13958.87 5583 .575	13343.96 6218 .770	79822.32 492 1.711	The 11

Table 5-A. Continued

Problem	Schwarz's Method (The 8)	Branch and Bound (The 9)	Myopic Policies (The 10)	Modified Myopic Policies (The 11)	Author's Approach (The 14)	Best Solution
8	10208.89 2683 .755	9877.25 4920 1.801	10195.42 2824 .619	9927.74 3988 .744	10001.86 3487 1.648	The 9
9	12825.09 2683 .762	12571.53 4868 1.819	12824.85 2666 .587	12824.85 2666 .732	12658.11 3487 1.680	The 9
10	7115.12 379 .749	3521.01 1482 1.769	3549.65 1419 .584	3521.01 1482 .771	5736.95 492 1.661	The 9 The 11
Time Averages	.762	1.745	.600	.764	1.634	

†

A
B
C

A - Cost of production program (dollars)

B - Lot size at stage 1 (units)

C - Computer time (seconds)

Table 5-B. Detailed Results For Modified Myopic Policies

Problem	Total Cost (\$)	Lot Size at Stage 1 (Units)	K_2	K_3	K_4	K_5	K_6	K_7	K_8	K_9	K_{10}
1	11512.60	5878	2	1	5	2	1	5	2	1	5
2	13244.02	6428	2	1	6	2	1	6	2	1	1
3	10658.33	2702	1	3	1	1	3	1	1	3	1
4	12525.97	2586	1	3	1	1	3	1	1	3	3
5	3545.65	1419	1	1	1	1	1	1	1	1	1
6	11315.84	5678	1	1	4	1	1	5	2	1	7
7	13343.96	6218	1	1	4	2	1	6	2	1	1
8	9927.74	3988	1	3	1	1	5	1	2	6	1
9	12824.85	2666	1	2	1	1	3	1	1	4	4
10	3521.01	1482	1	1	1	1	1	1	1	1	2

substantial variation in set up costs. In the other six problems, the results were better but not optimal. Branch and Bound exceeded the time limit in the four problems that caused troubles in Schwarz's Method, but in the other six cases gave the best solution with approximately twice the lowest time. Myopic Policies obtains the best solution only once, but the cost performance is very good and times is excellent. Modified Myopic Policies was the best for problems where Branch and Bound did not obtain any result, and best for two other problems. Finally, Dynamic Programming procedure yields somewhat better solutions than Schwarz's Method, but requires about twice the computation time.

Table 5-B shows detailed results for modified myopic policies; the first column presents the total cost in dollars; second, the lot size for stage 1 in units; and the remaining columns are k-factors ($k_n = D_n Q_1 / Q_n D_1$) for stages at level 1.

In the first five problems, demand is kept constant for the nine final products while the last five problems have differing requirements. Note that the average value of these requirements is exactly the demand used in the first problems.

By comparing problems with equal first level stage demand with those having similar costs but differing first level stage demands, (1 vs. 6, 2 vs. 7, 3 vs. 8, 4 vs. 9, 5 vs. 10), we observe that the lot sizes at stage 1 do not vary significantly, but the lot sizes of level 1 stages do.

Acyclic Structure

Problem Definition

Up to this moment we have analyzed specific two-level structures

of the serial, pure assembly (at most one successor stage), and pure arborescence (at most one predecessor stage) forms. A more general system, known as the acyclic structure, allows any number of stages at any level (see Figure 14).

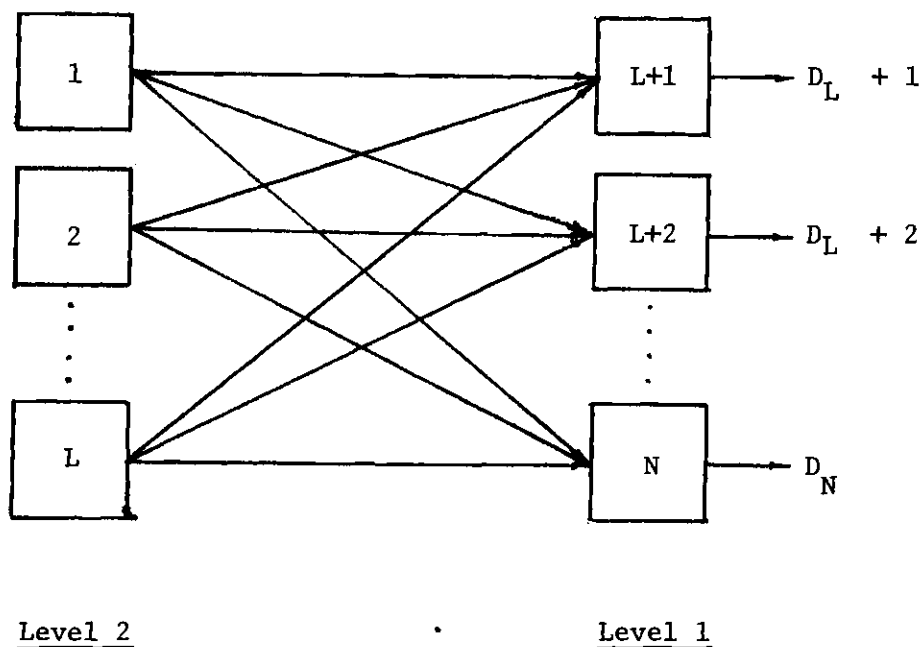


Figure 14. Acyclic Structure

In this structure, there may or may not be a relationship between given level 1 and level 2 stages. Any stage at any level must have at least one relation with a stage on the opposite level. The stages on level 1 possess their own demand, and stages at level 2 obtain their average demand rate by the summation of all the demands at level 1 stages that place requirements upon them.

The assumptions made before are also applied here: the demand is static, the deliveries and production are assumed to be instantaneous, and no backlogging is allowed.

Then the objective is to

$$\text{minimize } \sum_{i=1}^N (S_i D_i / Q_i + Q_i h_i / 2) \quad (2.13)$$

The literature on multi-stage systems with constant demand includes assembly and arborescence structures, but nothing was found on more general structures.

Analysis

Consider equation (2.13) subject to the policy constraint.

$$\frac{Q_i}{D_i} = k_{iJ} \frac{Q_J}{D_J} \quad k_{iJ} \geq 1, \text{ integer } 1 \leq i \leq L; L+1 \leq J \leq N \quad (2.14)$$

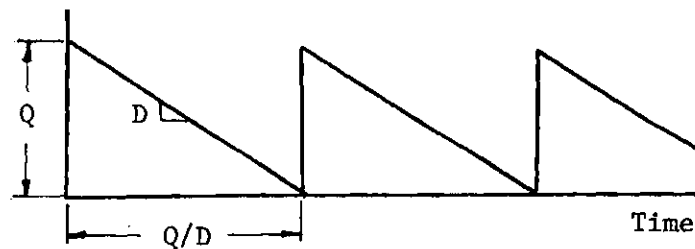


Figure 15. Inventory Model

Since Q/D is the cycle time, as shown in Figure 15, equation (2.14) can be interpreted as: the cycle time on any stage at level 1 must be an integer multiple of each of its successor stages' cycle time.

Let us define Q_i^c as the optimal individual lot size,

$$Q_1^c = \left[2S_1 D_1 / h_1 \right]^{1/2} \quad (2.15)$$

and let t be the cycle time defined as Q/D .

Using the same approach as in assembly and arborescence cases, optimal individual lot sizes are used to determine the bounds on cycle time, consider

$$t_{\min} = \text{Lower cycle time bound} = \text{MIN} \left[Q_1^c / D_1 \right]$$

$$t_{\max} = \text{Upper cycle time bound} = \text{MAX} \left[Q_1^c / D_1 \right] \text{ where } 1 \leq i \leq N$$

Then assuming constraint (2.14), the optimal cost for a stage located at level 2 given a t cycle time is

$$f_n^*(k_n t) = \text{MIN}_{k_n} \left[S_n / k_n t + k_n t D_n h_n / 2 \right] \quad (2.16)$$

and for stages on level 1, the cost is

$$f_n^*(k_n t) = \text{MIN}_{k_n} \left[S_n k_n / t + t D_n h_n / 2 k_n \right] \quad (2.17)$$

Observe that cycle time for stages at level 2 is a multiple of t while for stages at level 1 is a divisor of it. Then total cost (T) for a given t is as follows:

$$T(t) = \sum_{n=1}^N f_n(k_n^* t) \quad (2.18)$$

and the optimal cycle time t will be the one with minimum total cost $[T(t)]$,

$$T^* = T(t^*) = \underset{t_{\min} \leq t \leq t_{\max}}{\text{minimum}} [T(t)] \quad (2.19)$$

Results

A computer program was written (THE 15) to apply the algorithm for solving any kind of acyclic structure in two level systems. The data, as for the other programs, are the set up costs, inventory carrying costs and demands; but THE 15 also requires a matrix of zeroes and ones with L rows and $(N-L)$ columns that correspond to the existence or nonexistence of the relation between a stage on level 1 and a stage on level 2 (denoted by 1 and 0 respectively). Program THE 15 is in Appendix B.

Again ten problems were selected to test the behavior of the algorithm, all of them correspond to the specific structure shown in Figure 16, where stage 1 supplies stages 3 and 4, and stage 2 supplies stages 3, 4, and 5.

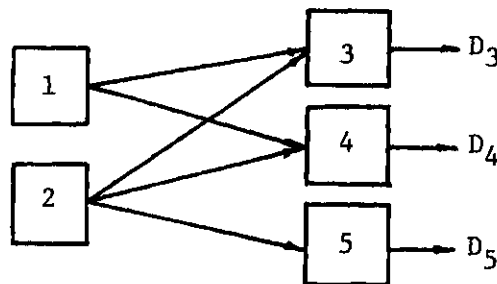


Figure 16. Acyclic Problem

Data and results are presented in Table 6. To interpret the results obtained, we note all that variations among the problems are (a) a decrement on the demand in the last five problems, (b) changes in set up costs, and (c) changes in inventory carrying costs.

Variation (a) results generally in an increase in the cycle time. This is explained by the relation $t = Q/D$, so decreasing D increases t . This does not happen always, because of the optimal t involves other factors besides demand. To explain the effects of (a) on k values let us derive some formulas.

At level 2, the cost at stage n is

$$f_n(kt) = S_n/kt + k_t D_n h_n / 2 \quad (2.20)$$

Minimizing for a given t , we obtain

$$k^* = (2S_n/t^2 D_n h_n)^{1/2} \quad (2.21)$$

At level 1, stage n ,

$$f_n(kt) = S_n k / t + t D_n h_n / 2k \quad (2.22)$$

Again minimizing for a given t ,

$$k^* = (t^2 D_n h_n / 2S_n)^{1/2} \quad (2.23)$$

Then, at level 2, k increases as D decreases, while at level 1 the reverse

Table 6. Data And Results For Acyclic Structure

Problem	1			2			3			4			5		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage															
1	10	1.25		100	1.25		100	.25		100	2.5		10	1.25	
2	100	1.25		1000	1.25		100	2.5		100	25		15	1.25	
3	1000	1.25	1000	10	1.25	1000	100	25	1000	100	.25	1000	20	1.25	1000
4	10	1.25	1000	100	1.25	1000	100	.25	1000	100	2.5	1000	25	1.25	1000
5	100	1.25	1000	1000	1.25	1000	100	2.5	1000	100	25	1000	30	1.25	1000

S - Set up cost h - Inventory carrying cost (dollars) D - Demand (units)

Problem	6			7			8			9			10		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage															
1	10	1.25		100	1.25		100	.25		100	2.5		10	1.25	
2	100	1.25		1000	1.25		100	2.5		100	25		15	1.25	
3	1000	1.25	200	10	1.25	200	100	25	200	100	.25	200	20	1.25	200
4	10	1.25	400	100	1.25	400	100	.25	400	100	2.5	400	25	1.25	400
5	100	1.25	600	1000	1.25	600	100	2.5	600	100	25	600	30	1.25	600

Table 6. Continued

Problem	1	2	3	4	5
Total Cost	4769.48	6288.24	5043.75	10859.08	1414.22
Stage 1	1119*	1619	1500	544	282
Stage 2	1679	2428	750	408	424
Stage 3	560	135	83	136	141
Stage 4	140	405	250	136	141
Stage 5	560	809	250	68	141
Cycle Time	.5597	.8094	.25	.1359	.1412
Computer Time	1.403	1.361	1.336	1.286	1.277

*Lot size (units)

Table 6. Continued

Problem	6	7	8	9	10
Total Cost	2844.72	3939.50	2870.75	7329.46	867.21
Stage 1	578	787	772	196	132
Stage 2	1156	1475	386	98	263
Stage 3	193	61	16	16	44
Stage 4	77	246	129	33	88
Stage 5	289	736	193	49	132
Cycle Time	.963333	1.228833	.321500	.081667	.219167
Computer Time	1.324	1.327	1.272	1.335	1.33

is true. Equations (2.20) - (2.23) also show that demand is directly proportional to the cost. Then, the lower the demand, the lower the cost of the system.

CHAPTER III

TIME VARYING DEMAND CASE

In this chapter, we consider the case where the external requirements imposed on first level stages vary over time. Instead of the continuous review, infinite horizon analysis used to treat the stationary demand processes of the previous chapter, we now assume a finite planning horizon, divided into M periods. External requirements in each period are assumed to be known. All units scheduled for production at a stage in a period are assumed to be produced as a single lot. The problem is to choose the lot sizes at each stage in each period to meet the requirements with the least total cost for production and inventory over the planning horizon. The series, assembly, arborescence, and acyclic structures are analyzed in the following sections.

Series Structure

Problem Definition

The two-stage series system is illustrated in Figure 17. As in the previous chapter, it is convenient to think of stage 1 as the raw material inventory and stage 2 as the production process and the finished goods inventory. Backlogging is not allowed. Production is assumed to be instantaneous (the presence of a finite production lead time will not affect the lot size decisions). The time dimension illustrates that in addition to considering the flow of material from one stage to the next within a period, we also must conceive of flow of material from one time

period to the next at a given stage. The latter is the inventory carried forward. Thus, the requirements placed on an inventory at a stage in a given period can be satisfied by a combination of two sources: (1) production (stage 2) or procurement (stage 1) in that period, and (2) inventory brought forward from the previous period.

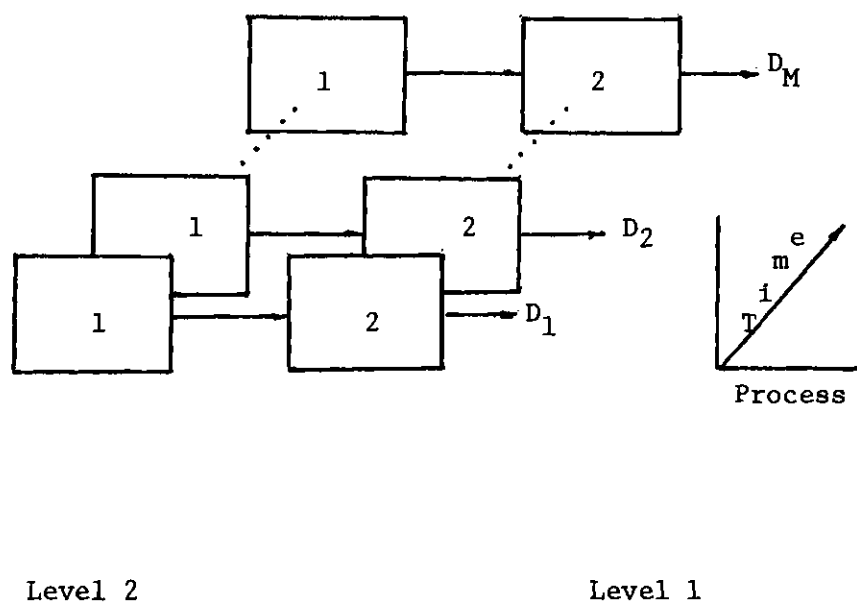


Figure 17. Dynamic Series Structure

Let Q_{ni} be the quantity produced at stage n in period i and define I_{ni} to be the inventory at the end of period i at stage n . The production or procurement cost of a lot size Q_{ni} at stage n in period i is assumed to have the fixed charge form:

$$P_{ni}(Q_{ni}) = \delta(Q_{ni})S_{ni} + c_{ni}Q_{ni} \quad (3.1)$$

where

$$\delta(Q) = \begin{cases} 1, & \text{if } Q > 0 \\ 0, & \text{if } Q = 0 \end{cases}$$

Inventory holding costs in a period are proportional to the ending inventory level; that is,

$$H_{ni}(I_{ni}) = h_{ni}I_{ni} \quad (3.2)$$

where h_{ni} is the cost to carry one unit in inventory at stage n from period i to $i+1$. The objective function to be minimized by the choice of lot sizes is

$$T = \sum_{n=1}^2 \sum_{i=1}^M [\delta(Q_{ni})S_{ni} + c_{ni}Q_{ni}] + \sum_{n=1}^2 \sum_{i=1}^M h_{ni}I_{ni} \quad (3.3)$$

where, if D_i is the external demand on stage 2 in period i ,

$$I_{2i} = I_{2,i-1} + Q_{2i} - D_i \quad (3.4)$$

(the finished goods inventory balance equation for period i), and

$$I_{1i} = I_{1,i-1} + Q_{1,i} - Q_{2,i} \quad (3.5)$$

(the raw material inventory balance equation for period i).

Literature Survey and Results

Zangwill [29][30] analyzes the multi-stage series system with dynamic demand and proves that, if production and inventory costs are each concave, there is an optimal solution having the property that $I_{n,i-1} Q_{ni}=0$, for all i and n . He then proposes an algorithm based on dynamic programming, with simplifications resulting from his characterization of the optimal solution. Love [16] also considers the series case and obtains additional results defining a class of solutions containing an optimal one. His most important result is that $Q_{n+1,i}=0$ implies $Q_{n,i}=0$, if costs are concave and inventory costs are non-decreasing in n . Applied to a two-stage system, this "nested" property implies that no raw material lots are scheduled for delivery in periods when there is no production. Johnson and Montgomery [11] give a discussion of Zangwill's algorithm and a numerical example illustrating its application for a three-stage system. However, the literature contains no true computational results and the methodology is not well-known.

Analysis

This section presents the dynamic programming algorithm of Zangwill [30], adapted specifically to the two level series system. He considered the general series structure with concave production cost functions, $P_{ni}(Q_{ni})$, and concave inventory holding cost functions, $H_{ni}(I_{ni})$. No backlogging is permitted. Zangwill's model for the case of two levels is the following:

$$\text{minimize } \sum_{i=1}^M \sum_{n=1}^2 [P_{ni}(Q_{ni}) + h_{ni}(I_{ni})] \quad (3.6)$$

subject to, for $i = 1, 2, \dots, M$ and $n = 1, 2,$

$$I_{1,i-1} + Q_{1,i} = I_{1,i} + Q_{2,i} \quad (3.7)$$

$$I_{2,i-1} + Q_{2,i} = I_{2,i} + D_i \quad (3.8)$$

$$I_{ni} \geq 0, Q_{ni} \geq 0 \quad (3.9)$$

$$I_{n,0} = I_{n,M} = 0 \quad (3.10)$$

Zangwill observed that this is the formulation of the problem of finding the least cost flow in a network consisting of a single source node, M transshipment nodes (representing stage 1 in each of the n periods) and M transshipment-sink nodes (representing stage 2). The network analogy is shown in Figure 18. Note that the flows represent either production, inventory, or demand. There is a cost associated with each production and inventory flow, and Zangwill assumes that the cost is a concave function of the amount of flow. He uses the fact that for a static, single source, concave cost network flow problem, there is an optimal solution that is an extreme flow. An extreme flow has the property that flow into any node will come from at most one arc. Practically this means that $I_{n,i-1}Q_{ni} = 0$, for all nodes (n,i) . Making use of this property characterizing an extreme flow, an algorithm can be developed to efficiently find the optimal extreme

flow and this solution will be an optimal solution to the problem (3.6) - (3.10). The implications of the extreme flow property for the two-level system are discussed in the following paragraphs.

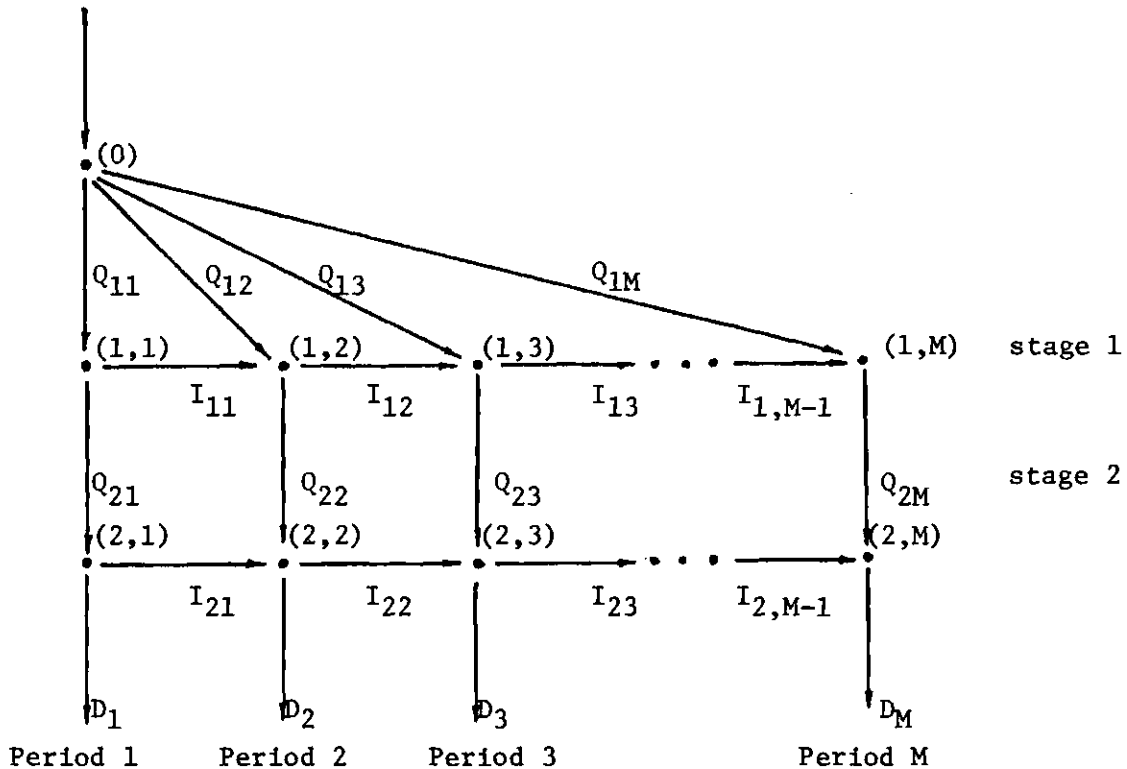


Figure 18. Network Representation

The flow into any node $(1,i)$ is restricted to either 0 or

$$\sum_{L=\alpha}^{\beta} D_L \quad (3.11)$$

for some $i < \alpha < \beta < M$. In other words, the amount of flow into node $(1,i)$ is 0

or the exact requirements for periods $\alpha, \alpha+1, \dots, \beta, i \leq \alpha \leq \beta \leq M$. Assuming $D_1 > 0$, we must produce at stage 1 at least D_1 units in period 1; therefore $i=1$ implies $\alpha=1$. Similarly for stage 2, we have the flow into any node $(2,i)$ restricted to one of the following values

$$\sum_{L=i}^{\beta} D_L \quad (3.12)$$

where $i \leq \beta \leq M$. Note that the stage 2 inventory is restricted to one of the following values:

$$0 \text{ or } \sum_{L=i+1}^{\beta} D_L, \quad i \leq \beta \leq M$$

If the flow into node $(1,i)$ is $\sum_{L=\alpha}^{\beta} D_L$, for some $i \leq \alpha \leq \beta \leq M$, we have the following:

(a) If $\alpha=i$, there is an integer $\gamma, i \leq \gamma \leq \beta$, such that

$$Q_2 = \sum_{L=1}^{\gamma} D_L, \text{ and } I_{1,i} = \sum_{L=\gamma+1}^{\beta} D_L \quad (3.13)$$

(b) If $\alpha > i$,

$$Q_{2i} = 0, \text{ and } I_{1,i} = \sum_{L=\alpha}^{\beta} D_L \quad (3.14)$$

A dynamic programming algorithm for calculating the optimal flow will now be developed. Suppose the flow into a first-stage node $(1,i)$ is $\sum_{L=\alpha}^{\beta} D_L$ and define $C_{1,i}(\alpha,\beta)$ as the optimal cost of shipping these units from node $(1,i)$ to their destination nodes $(2,\alpha), (2,\alpha+1), \dots, (2,\beta)$. The decision at node $(1,i)$ is to divide in the flow into two parts, the production at stage 2 in period i , Q_{2i} , and the inventory carried to the next period, $I_{1,i}$. If $\alpha=i$, we must select the optimal γ in (3.13), and if $\alpha>i$, (3.14) holds. Thus we have the following recursions:

$$C_{1,i}(\alpha,\beta) = \begin{cases} \min_{i \leq \gamma \leq \beta} \left[P_{2,i} \left(\sum_{L=1}^{\gamma} D_L \right) + H_{1,i} \left(\sum_{L=\gamma+1}^{\beta} D_L \right) + C_{2,i}(i,\gamma) + C_{1,i+1}(\gamma+1,\beta) \right] & \text{if } i = \alpha \leq \beta \leq M \\ H_{1,i} \left(\sum_{L=\alpha}^{\beta} D_L \right) + C_{1,i+1}(\alpha,\beta), & \text{if } i < \alpha \leq \beta \leq M \end{cases} \quad (3.15)$$

where $C_{2,i}(i,\beta)$ is the cost of allocating a flow of $\sum_{L=1}^{\beta} D_L$ units into node $(2,i)$ to satisfy the requirements in periods $i, i+1, \dots, \beta$. We have

$$C_{2,i}(i,\beta) = H_{2,i} \left(\sum_{L=i+1}^{\beta} D_L \right) + C_{2,i+1}(i+1,\beta) \quad (3.16)$$

We next allocate the flow into the source node in an optimal manner. This is equivalent to making decisions about what to produce at the first stage in each period. Let $C_{0,i}(\alpha,M)$ be the minimum cost of shipping $\sum_{L=\alpha}^M D_L$ units from the source to satisfy requirements at nodes $(2,\alpha), (2,\alpha+1), \dots, (2,M)$. If $\alpha=1$, we must produce in period i ; otherwise $Q_{1,i}$ might be zero.

If we produce in period i , the quantity can be described by an integer

$\alpha \leq \gamma \leq M$, since $Q_{1,i} = \sum_{L=\alpha}^{\gamma} D_L$. There are three equations defining $C_{0,i}(\alpha, M)$:

If $i < \alpha \leq M$,

$$C_{0,i}(\alpha, M) = \min \left[\begin{array}{l} \min_{\alpha \leq \gamma \leq M} P_{1,i} \left(\sum_{L=\alpha}^{\gamma} D_L \right) + C_{1,i}(\alpha, \gamma) + C_{0,i+1}(\gamma+1, M) \\ C_{0,i+1}(\alpha, M) \end{array} \right] \quad (3.17)$$

If $i = \alpha < M$,

$$C_{0,i}(i, M) = \min_{\alpha \leq \gamma \leq M} \left[P_{1,i} \left(\sum_{L=\alpha}^{\gamma} D_L \right) + C_{1,i}(\alpha, \gamma) + C_{0,i+1}(\gamma+1, M) \right] \quad (3.18)$$

If $i = \alpha = M$,

$$C_{0,M}(M, M) = P_{1,M}(D_M) + C_{1,M}(D_M) \quad (3.19)$$

The algorithm initially calculates the $C_{2,i}(i, \beta)$ using (3.16), then the $C_{1,i}(\alpha, \beta)$ using (3.15), and finally the $C_{0,i}(\alpha, M)$ using (3.17), (3.18), and (3.19). The minimum cost is given by $C_{0,1}(1, M)$.

Results

A computer program (THE 16) was written for the algorithm using the

fixed charge production cost form and proportional inventory holding costs. (See Equations 3.1 and 3.2.) To simplify the analysis, the cost coefficients were assumed to be stationary in time. This meant that the variable production cost would be independent of the lot size policies and could be ignored.

There remained three interesting parameter sets to analyze in this problem: set up cost, inventory carrying cost, and demand. Nine problems, each with $M=5$, were used to understand the behavior of the three variables (Table 7). The first three problems vary only the set up costs which increase from 50 to 150. At each stage the results shown in Table 8 indicated the expected tendency to increase the lot size and keep units on inventory rather to produce at each period of time. The second set of problems (4-6) have variation only in the inventory carrying costs, from .05 to .15. It can be observed that the greater the inventory carrying costs, the greater the tendency to produce in each of the periods rather than maintain stocks. Finally the last three problems have time varying demand at increasingly higher levels. As it was expected the effects are the same as when the inventory costs were increased. This is because the set up costs are not altered with the variation on the demand, while the inventory carrying costs are; the greater the demand, the greater amount of units that have to be carried, and consequently the greater the inventory costs.

Assembly Structure

Problem Definition

Now that the idea of time varying demand has been developed, it is not difficult to extend the model of Chapter II, Part B, to this condition.

Table 7. Problems For Series Structure

	Stage 1	Stage 2	Stage 1	Stage 2	Stage 1	Stage 2	Stage 1	Stage 2
	Problem 1		Problem 2		Problem 3		Problem 4	
Set Up	50*	50	100	100	150	150	100	100
Inventory	.10*	.10	.10	.10	.10	.10	.05	.05
Demand 1		1000		1000		1000		1000
Demand 2		1000		1000		1000		1000
Demand 3		1000		1000		1000		1000
Demand 4		1000		1000		1000		1000
Demand 5		1000		1000		1000		1000

*Dollars

Demand i = Demand at Period i

Table 7. Continued

	Stage 1	Stage 2	Stage 1	Stage 2	Stage 1	Stage 2	Stage 1	Stage 2
	Problem 5		Problem 6		Problem 7		Problem 8	
Set Up	100	100	100	100	100	200	100	200
Inventory	.10	.10	.15	.15	.10	.20	.10	.20
Demand 1		1000		1000		200		400
Demand 2		1000		1000		400		800
Demand 3		1000		1000		600		1200
Demand 4		1000		1000		800		1600
Demand 5		1000		1000		1000		2000

Problem 9		
Set Up	100	200
Inventory	.10	.20
Demand 1		800
Demand 2		1600
Demand 3		2400
Demand 4		3200
Demand 5		4000

Table 8. Production Schedule In Series Structure, Time Varying Demand

Stage	Problem 1		Problem 2		Problem 3		Problem 4		Problem 5		Problem 6		Problem 7	
	1	2	1	2	1	2	1	2	1	2	1	2	1	2
Period														
1	1000*	1000	1000	1000	2000	1000	2000	1000	1000	1000	1000	1000	600	200
2	1000	1000	1000	1000	-	1000	-	1000	1000	1000	1000	1000	-	400
3	1000	1000	1000	1000	2000	1000	2000	1000	1000	1000	1000	1000	1400	600
4	1000	1000	1000	1000	-	1000	-	1000	1000	1000	1000	1000	-	800
5	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

*Production units

Stage	Problem 8		Problem 9	
	1	2	1	2
Period				
1	1200	400	800	800
2	-	800	1600	1600
3	1200	1200	2400	2400
4	1600	1600	3200	3200
5	2000	2000	4000	4000

The problem will be to obtain a production program for all stages over an M -period horizon. The idea of Figure 19 is an assembly structure displaced through the time; in other words, we are adding one dimension to our old problem. Stages at level 2 (1 to $N-1$) are considered independent among

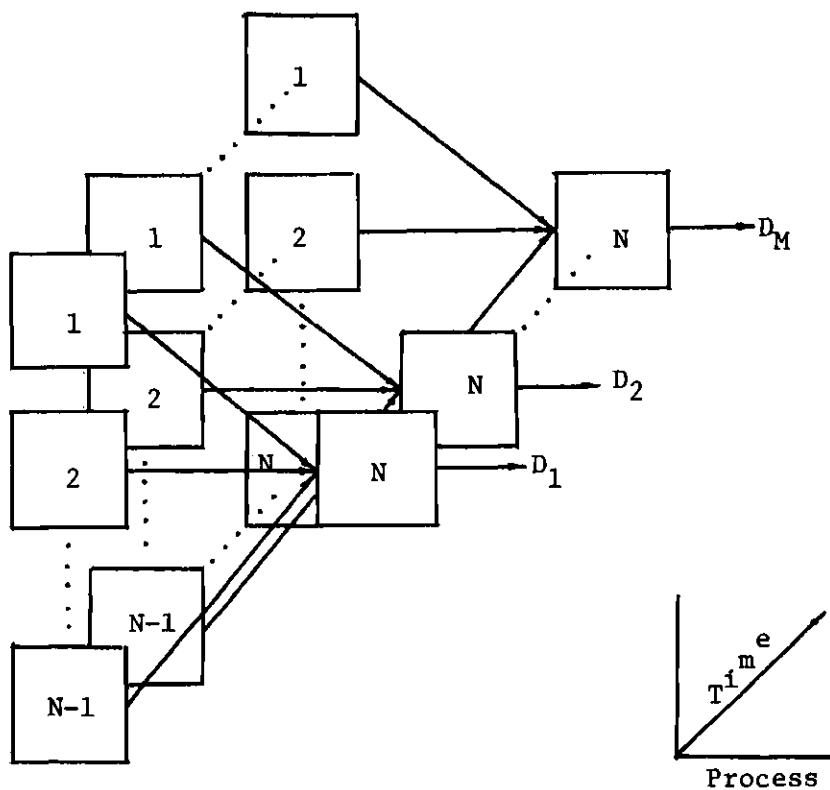


Figure 19. Dynamic Assembly Structure

themselves, and each one can receive material from two sources. The first one corresponds to the suppliers and the second one is merely the same

stage but displaced one period of time earlier. Obviously there are two alternatives, non-exclusive for disposition of the output. They are to send material to stage N in the same period i and to keep material in inventory until the next period.

Stage N has two possible exclusive inputs, the first is the N-1 deliveries from stages at level 2, and the second is inventory from the same stage N, but from period i-1. Logically stage N has two possible outputs to supply the demand of period i and optionally, store final product to satisfy further demands.

Costs, as in previous cases, are concave and composed of set up costs, which are incurred just if a new order or production takes place on the period i, inventory holding costs, which are proportional to the units carried at the end of the period.

Mathematically the problem can be expressed as

$$\text{minimize } T = \sum_{n=1}^N \sum_{i=1}^M \delta(Q_{ni}) S_n + \sum_{n=1}^N \sum_{i=1}^M h_n I_{ni} \quad (3.20)$$

$$\text{subject to: } \begin{array}{ll} I_{n,i} + I_{n,i-1} + Q_{n,i} - Q_{N,i} & n = 1, 2, \dots, N-1 \\ & i = 1, 2, \dots, M \end{array} \quad (3.21)$$

$$\begin{array}{ll} Q_{n,i} \geq 0 & n = 1, \dots, N \\ I_{n,i} \geq 0 & i = 1, 2, \dots, M \end{array} \quad \delta(Q) = \begin{cases} 1 & \text{if } Q > 0 \\ 0 & \text{if } Q = 0 \end{cases}$$

which is the minimization of all the costs incurred at the N stages during the M periods of time. S_n and h_n represent the set up cost and carrying

inventory cost respectively. It is assumed that these values do not change through the time.

Literature Survey and Results

The assembly structure with time varying demands has been considered by Crowston and Wagner [7] who, using an approach suggested by Love [16], extended his results to the assembly problem, and proposed two algorithms. The first one is a dynamic programming approach whose solution time is linear with the number of stages, but exponential with the number of time periods. The second is a branch and bound algorithm. Though the authors refer to computer experience, they do not give any results.

Analysis

Between the two algorithms proposed by Crowston and Wagner, the dynamic programming approach was selected to be developed in the present analysis, but initially let us consider the form of the optimal solution.

Veinott [26] shows that at least one optimal solution is an extreme flow, that is

$$I_{n,i-1} Q_{n,i} = 0 \quad i = 1, 2, \dots, M \quad n = 1, \dots, N \quad (3.22)$$

Love [16] proves the additional "nested" property for the facilities in series case:

$$Q_{n,i} > 0 \text{ implies } Q_{N,i} > 0 \text{ for } i = 1, \dots, M \quad (3.23)$$

$$n = 1, 2, \dots, N$$

under the conditions that:

- (1) the system is a pure series system,
- (2) production costs are non-increasing in time,
- $$C_{n,i}(Q) \leq C_{n,i-1}(Q) \text{ for } n = 1, \dots, N \quad (3.24)$$
- $$i = 2, \dots, M$$

- (3) installation inventory holding costs are non-decreasing over stages
- $$h'_{n,i} \geq h'_{m,i} \text{ for } m \in b(n), \quad (3.25)$$
- $$n = 1, 2, \dots, N$$
- $$i = 1, \dots, M$$

Crowston and Wagner at this point add the following assumption for assembly systems.

$$h'_{ni} \geq \bar{\sum}_{m \in b(n)} h'_{mi}, \quad n = 1, \dots, N, \quad i = 1, \dots, M \quad (3.26)$$

Note that (3.26) refers to installation inventory carrying cost rates.

For echelon inventory (3.26) implies

$$h_{ni} = h'_{ni} - \bar{\sum}_{m \in b(n)} h'_{mi} \geq 0 \quad (3.27)$$

Let us now introduce the algorithm:

Let $\Pi = (\Pi_1, \Pi_2, \dots, \Pi_M)$ be a vector of 1's and 0's. We interpret Π as a production profile vector, such that if the profile Π is chosen for stage n

$$Q_{n,i} = 0 \text{ if } \Pi_i = 0 \quad (3.28)$$

$$Q_{n,i} > 0 \text{ if } \Pi_i = 1$$

Then according with the optimal solution form

$$\begin{aligned}
Q_{n,i} &= 0 && \text{if } \Pi_i = 0 \\
Q_{n,i} &= D_i && \text{if } \Pi_i = 1 \text{ and } \Pi_{i+1} = 1 \\
Q_{n,i} &= D_i + D_{i+1} && \text{if } \Pi_i = 1, \Pi_{i+1} = 0, \text{ and } \Pi_{i+2} = 1
\end{aligned} \tag{3.29}$$

and in general,

$$Q_{n,i} = \sum_{t=1}^{i'} D_t \text{ if } \Pi_i = 1, \Pi_t = 0 \text{ for } t = i+1, \dots, i' \text{ and } \Pi_{i'+1} = 1$$

Let us denote the total production cost associated with the choice of production profile for stage n by $C_n(\Pi)$. Thus

$$C_n(\Pi) = \sum_{i=1}^M C_{n,i}(Q_{n,i}) \tag{3.30}$$

and the inventory would be

$$I_{n,i} = \sum_{t=1}^i (Q_{n,t} - D_t) \tag{3.31}$$

with a total carrying inventory cost for stage n

$$h_n(\Pi) = \sum_{i=1}^M h_{n,i} I_{n,i} \tag{3.32}$$

Using the nested schedules property, let $N(\Pi)$ be the set of production profiles which "nest" Π , that is,

$$\Pi' \in N(\Pi) \text{ if and only if } \Pi_i - \Pi'_i \geq 0; \quad i = 1, 2, \dots, M \quad (3.33)$$

Theorem 1 given in (7) and proved in (8) specifies that if Π is chosen for stage n , the production profile for each of the immediate predecessors of stage n must be an element of $N(\Pi)$.

Let $f_n(\Pi)$ be the minimum total cost from stages 1 through n of using production profile Π at stage n given that Π is chosen for stage n and given optimal choice of production profiles for all predecessor stages.

$$f_n(\Pi) = h_n(\Pi) + C_n(\Pi) + \sum_{m \in b(n)} \text{minimum}_{\Pi' \in N(\Pi)} f_m(\Pi') \quad (3.34)$$

Let $N^1(\Pi)$ be the set of profiles that nest Π and differ from Π in only one element; thus

$$N^1(\Pi) = \left[\Pi' \in N(\Pi) \mid \sum_{i=2}^M (\Pi_i - \Pi'_i) = 1 \right] \quad (3.35)$$

Then

$$N(\Pi) = \Pi \cup N(\Pi'), \quad \Pi' \in N^1(\Pi) \quad (3.36)$$

$$N(\Pi) = \Pi \cup_{\Pi' \in N^2(\Pi)} N(\Pi')$$

$$\text{Let } \Pi^*(\Pi') = \arg \text{minimum}_{\Pi'' \in N(\Pi')} f_m(\Pi'') \quad (3.37)$$

Then

$$\text{minimum}_{\Pi' \in N(\Pi)} f_m(\Pi') = \text{minimum} \left[f_m(\Pi), \text{minimum}_{\Pi' \in N^1(\Pi)} f_m[\Pi^*(\Pi')] \right] \quad (3.38)$$

Results

The algorithm obtained above was programmed as THE 17 (Appendix B). This program receives the characteristics of all the stages (costs) and the demands of stage N, and it obtains a production schedule for all the stages through the whole horizon time. For the purpose of computing the optimal lot sizes, only set up cost and inventory carrying costs were considered; however the program also uses the variable cost to evaluate the total cost incurred by the company.

Using representative problems (Table 9), the response of the model to variations in the parameters was studied.

Changes were of three types: set up costs, inventory carrying costs, and demands. The effect of set up cost changes can be observed in problems 1 and 2, where it is shown clearly that the greater the set up cost, the greater the tendency to satisfy future demands with inventory rather than produce in the period in which demand is required.

For different set up costs at stage N, problems 3, 4, 5, and 6 have variations on inventory cost. Comparing 3 with 5, 4 with 6, and taking a representative stage, say number 5, requirements for periods 3 and 5 were supplied by inventory material when inventory cost was low; while the requirements were filled by production when costs were higher.

Finally problems 7 through 10 have variations in their demand values. Obviously the greater the demand, the greater the frequency of production. This is because, the greater the demand, the greater the

Table 9. Problems For Assembly Structure, Time Varying Demand

Problem	1			2			3			4			5		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage															
1	50	.15		50	.15		50	.15		50	.15		50	.25	
2	100	.15		100	.15		100	.15		100	.15		100	.25	
3	1000	.15		1000	.15		150	.15		150	.15		150	.25	
4	50	.15		50	.15		200	.15		200	.15		200	.25	
5	100	.15		100	.15		250	.15		250	.15		250	.25	
6	1000	.15		1000	.15		300	.15		300	.15		300	.25	
7	50	.15		50	.15		350	.15		350	.15		350	.25	
8	100	.15		100	.15		400	.15		400	.15		400	.25	
9	1000	.15		1000	.15		450	.15		450	.15		450	.25	
10	50	.15	1000*	1000	.15	1000*	50	.15	1000*	450	.15	1000*	50	.25	1000*

S - Set up cost (dollars)

h - Inventory carrying cost (dollars)

D - Demand (units)

*The demand is kept constant through the time

Table 9. Continued

Problem	6			7			8			9			10		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage															
1	50	.25		50	.25		50	.25		50	.25		50	.25	
2	100	.25		100	.25		100	.25		100	.25		100	.25	
3	150	.25		150	.25		150	.25		150	.25		150	.25	
4	200	.25		200	.25		200	.25		200	.25		200	.25	
5	250	.25		250	.25		250	.25		250	.25		250	.25	
6	300	.25		300	.25		300	.25		300	.25		300	.25	
7	350	.25		350	.25		350	.25		350	.25		350	.25	
8	400	.25		400	.25		400	.25		400	.25		400	.25	
9	450	.25		450	.25		450	.25		450	.25		450	.25	
10	450	.25	1000*	50	.25	500*	450	.25	500*	50	.25	2000*	450	.25	2000*

amount that has to be carried in inventory and the greater the resulting cost. Considering that set up costs do not vary with the size of the lot, there is a trade-off point where it is more convenient to produce at every period, rather than produce for several periods and keep the material in stock. (Table 10 shows all the production schedules for THE 10 tested problems.)

Arborescence Structure

Problem Definition

From Chapter II, an arborescence structure is one where each one of the facilities has a unique immediate predecessor that supplies facility n . Consider the same structure shown in Figure 13, but with the additional time dimension (Figure 20). In Figure 20, the arrows corresponding to the demand flows were omitted for clarity of presentation.

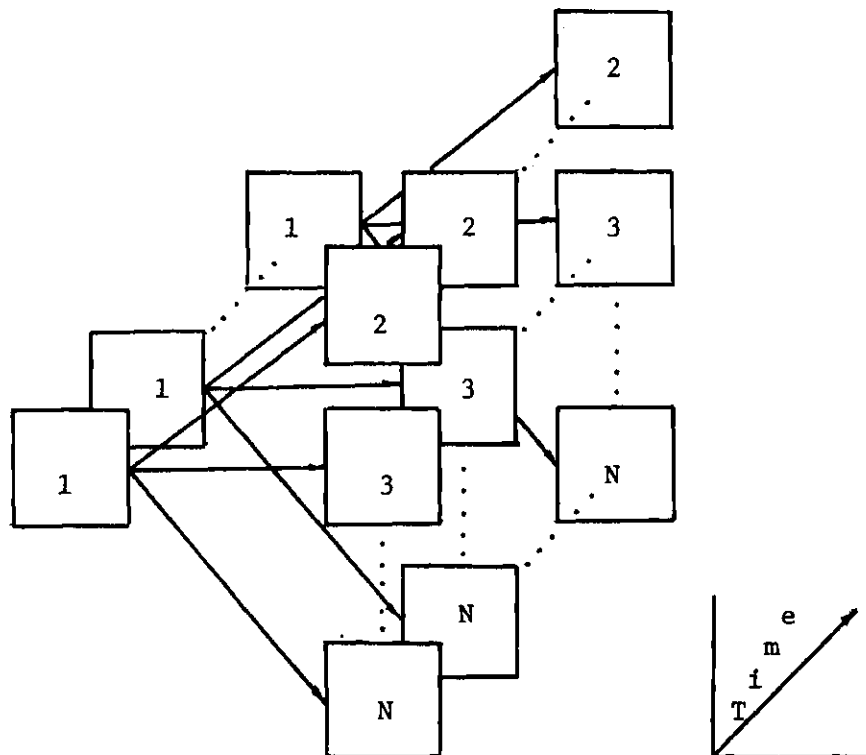


Figure 20. Dynamic Arborescence Structure

Table 10. Production Schedules For Assembly Structure, Time Varying Demand

Period	Problem 1					Problem 2					Problem 3				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	1000*	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
3	5000	0	0	0	0	5000	0	0	0	0	1000	1000	1000	1000	1000
4	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	2000	0	2000	0
5	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	2000	0	2000	0
6	5000	0	0	0	0	5000	0	0	0	0	2000	0	3000	0	0
7	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	2000	0	3000	0	0
8	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	2000	0	3000	0	0
9	5000	0	0	0	0	5000	0	0	0	0	2000	0	3000	0	0
10	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

*Production in units

Table 10. Continued

Period	Problem 4					Problem 5					Problem 6				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
2	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
3	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
4	1000	2000	0	2000	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
5	1000	2000	0	2000	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
6	2000	0	3000	0	0	1000	2000	0	2000	0	1000	2000	0	2000	0
7	2000	0	3000	0	0	1000	2000	0	2000	0	1000	2000	0	2000	0
8	2000	0	3000	0	0	1000	2000	0	2000	0	1000	2000	0	2000	0
9	2000	0	3000	0	0	1000	2000	0	2000	0	1000	2000	0	2000	0
10	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Table 10. Continued

Period	Problem 7					Problem 8					Problem 9				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	1000	0	1000	0	500	1000	0	1000	0	500	2000	2000	2000	2000	2000
2	1000	0	1000	0	500	1000	0	1000	0	500	2000	2000	2000	2000	2000
3	1000	0	1000	0	500	1000	0	1000	0	500	2000	2000	2000	2000	2000
4	1000	0	1000	0	500	1000	0	1000	0	500	2000	2000	2000	2000	2000
5	1000	0	1000	0	500	1000	0	1000	0	500	2000	2000	2000	2000	2000
6	1000	0	1500	0	0	1000	0	1500	0	0	2000	2000	2000	2000	2000
7	1000	0	1500	0	0	1000	0	1500	0	0	2000	2000	2000	2000	2000
8	1000	0	1500	0	0	1000	0	1500	0	0	2000	2000	2000	2000	2000
9	1000	0	1500	0	0	1000	0	1500	0	0	2000	2000	2000	2000	2000
10	1000	0	1000	0	500	1000	0	1000	0	500	2000	2000	2000	2000	2000

Table 10. Continued

Period	Problem 10				
	1	2	3	4	5
Stage					
1	2000	2000	2000	2000	2000
2	2000	2000	2000	2000	2000
3	2000	2000	2000	2000	2000
4	2000	2000	2000	2000	2000
5	2000	2000	2000	2000	2000
6	2000	2000	2000	2000	2000
7	2000	2000	2000	2000	2000
8	2000	2000	2000	2000	2000
9	2000	2000	2000	2000	2000
10	2000	2000	2000	2000	2000

As is characteristic of dynamic systems, any stage has two exclusive inputs and two nonexclusive outputs. Stage 1 now is located at level two and receives material from suppliers or from an excess of input over its own demand in previous periods. Stages 2 through N form level one, and each produces a different product that has its own demand in each specific period of time. These stages also have two inputs, stage 1 or an inventory carried from period $i-1$ and they have two outputs, the requirements for the products, and the inventory to satisfy further demands.

Costs are assumed to be concave and composed of two elements, set up cost and inventory carrying cost. Then the mathematical objective formulation will be identical to the models discussed before with small differences in inventory constraints.

$$\text{minimize } T = \sum_{n=1}^N \sum_{i=1}^M \delta(Q_{ni}) S_n + \sum_{h=1}^N \sum_{i=1}^M h_n I_{n,i} \quad (3.39)$$

subject to

$$I_{n,i} = I_{n,i-1} + Q_{n,i} - \sum_{m \in a(n)} Q_{m,i} \quad \begin{array}{l} n = 1 \\ i = 1, 2, \dots, M \end{array} \quad (3.40)$$

$$Q_{ni} \geq 0 \quad I_{n,i} \geq 0 \quad n = 1, 2, \dots, N; \quad i = 1, 2, \dots, M$$

$$\delta(Q) = \begin{cases} 1 & \text{if } Q > 0 \\ 0 & \text{if } Q = 0 \end{cases}$$

Again in this model the values of the costs for the stages are considered to be constant over time; only demand values may vary.

Literature Survey and Results

Veinott [26] develops an algorithm for the arborescence structure using the extreme point properties of Leontief substitution models to characterize the optimal solution. He does not present any computational experience or results.

Kalymon [13] develops a more efficient algorithm for this system using decomposition methods, and apparently these two papers are the only ones that fall under this classification.

Analysis

Veinott's algorithm will be developed in this analysis. Before presenting the main steps, we consider the characteristics of the optimal solution form. Let

$$Q_{1,i} - \sum_{n=2}^N Q_{n,i} + I_{1,i-1} - I_{1,i} = 0 \quad (3.41)$$

There is a J , $i \leq J \leq M$, such that

$$Q_{1,i} = \sum_{n=2}^N \sum_{u=i}^J Q_{n,u} \quad (3.42)$$

and

$$Q_{n,i} = \sum_{u=i}^J D_{n,u}, \quad 2 \leq n \leq N$$

These state simply that the amount ordered at facility i in period i is

the sum of the orders at its immediate followers (stages 2 to N), and for facilities at level 1, the amount ordered at period i should satisfy demands for periods i to J. Then the total cost is:

$$T = \sum_{n,i} \left[C_{n,i}(Q_{n,i}) + H_{n,i}(I_{n,i}) \right] \quad (3.43)$$

where $C_{n,i}(Q_{n,i})$ is the production cost at stage n in period i and $H_{n,i}(I_{n,i})$ is the inventory cost. Now consider

$$\delta(Q) = \begin{cases} 1 & \text{if } Q > 0 \\ 0 & \text{if } Q = 0 \end{cases} \quad (3.44)$$

$$\delta(Q, Q') = [1 - \delta(Q)]\delta(Q')$$

Notice that $\delta(Q, Q') = 1$ if and only if $Q=0$ and $Q' \neq 0$. Then T can be rewritten for the usual assumptions about costs as

$$T = \sum_{n=2}^N \sum_{i=1}^M \left[S_n \delta(Q_{1,i}, Q_{n,i}) + h_{n,i} I_{n,i} \right] + \sum_{i=1}^M \left[S_n \delta(Q_{1,i}) + h_{1,i} I_{1,i} \right] \quad (3.45)$$

Observe that fixed charge S_n is incurred at facility n in period i if and only if an order is placed at facility n in period i (i.e., $Q_{ni} > 0$) and if stage 1 does not place an order in that period (i.e., $Q_{1,i} = 0$). Finally because of the "nested" properties

$$I_{n,i-1} \cdot Q_{1,i} = 0 \quad (i = 2, \dots, M \text{ and } 2 \leq n \leq N) \quad (3.46)$$

The algorithm is the following. Let

$$\left[Q_{n,u:i,J} : i \leq u \leq J \right] \quad (3.47)$$

Denote optimal order quantities at facility n in periods $u = i+1, \dots, J$.

Let

$$\gamma_{1,u:iJ} = \sum_{v=u+1}^J \sum_{n=2}^N Q_{n,v:iJ} \quad (3.48)$$

Be the associated total requirements at facility 1 in periods $u+1, \dots, J$, and

let

$$\gamma_{n,u:iJ} = \sum_{v=u+1}^J D_{n,v:iJ} \quad 2 \leq n \leq N \quad (3.49)$$

Be the associated total requirements at facility n in periods $U+1, \dots, J$.

Let

$$C_{1,iJ} = S_1 \delta(\gamma_{1,i:iJ}) - \sum_{n=2}^M S_n \delta(Q_{n,iH:iJ}) + \sum_{U=i+1}^J h_{1,u}(\gamma_{1,U:iJ}) \quad (3.50)$$

be the cost for stage 1, and

$$C_{n,iJ} = S_n \delta(\gamma_{n,i:iJ}) + \sum_{u=i+1}^J h_{n,u}(\gamma_{n,u:iJ}) \quad 2 \leq n \leq N \quad (3.51)$$

be the cost for stages at level 1.

Let $f_{n,iJ}$ be the cost to produce in stage n ($2 \leq n \leq N$) the requirements for periods i to J and $f_{1,iJ}$ be the minimum cost for all N stages satisfying requirements for periods i to J .

$$f_{1,im} = \min_{i \leq j \leq m} \left[C_{1,iJ} + f_{n,iJ} + f_{1,Jm} \right] \quad (3.52)$$

It is important to note that $f_{n,ii} \equiv 0$. The algorithm sequentially finds $f_{1,M-1M}$, $f_{1,M-2M}$, ..., until $f_{1,1M}$ is evaluated.

Results

The computer algorithm written for arborescence structure has the name of THE 18. This algorithm achieves the same objective as previous ones: receives costs and demands and obtains optimal production schedule. This program also has the alternative to include variable costs for the processed items at any stage, and with a small modification it can also accept variations on the parameters through the time.

Tables 11 and 12 present data and results respectively. Ten problems were tested on THE 18 using the same variations as in the assembly case, with the following results.

Variations on set up costs appear to be not significant, since the production schedule is the same in the first problem as it is in the second. The cause of this abnormality is that constraint (3.47) gives a heavy weight to stage 1 in such a way that the production schedule for this stage dominates the remainder schedules and forces them to operate in the same time.

Table 11. Problems For Arborescence Structure, Time Varying Demand

Problem	1			2			3			4			5		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage															
1	5000	.15		5000	.15		3000	.15		3000	.15		3000	.25	
2	100	.15	1000*	100	.15	1000*	100	.15	1000*	100	.15	1000*	100	.25	1000*
3	1000	.15	1000*	1000	.15	1000*	150	.15	1000*	150	.15	1000*	150	.25	1000*
4	50	.15	1000*	50	.15	1000*	200	.15	1000*	200	.15	1000*	200	.25	1000*
5	100	.15	1000*	100	.15	1000*	250	.15	1000*	250	.15	1000*	250	.25	1000*
6	1000	.15	1000*	1000	.15	1000*	300	.15	1000*	300	.15	1000*	300	.25	1000*
7	50	.15	1000*	50	.15	1000*	350	.15	1000*	350	.15	1000*	350	.25	1000*
8	100	.15	1000*	100	.15	1000*	400	.15	1000*	400	.15	1000*	400	.25	1000*
9	1000	.15	1000*	1000	.15	1000*	450	.15	1000*	450	.15	1000*	450	.25	1000*
10	50	.15	1000*	1000	.15	1000*	50	.15	1000*	450	.15	1000*	50	.25	1000*

S - Set up cost (dollars)

h - Inventory carrying cost (dollars)

D - Demand (units)

*The demand is kept constant through the five periods

Table 11. Continued

Problem	6			7			8			9		
	S	h	D	S	h	D	S	h	D	S	h	D
Stage												
1	3000	.25		3000	.25		3000	.25		3000	.25	
2	100	.25	1000*	100	.25	500-1000**	100	.25	500-1000**	100	.25	1000-500**
3	150	.25	1000*	150	.25	1000-500**	150	.25	1000-500**	150	.25	500-1000**
4	200	.25	1000*	200	.25	500-1000**	200	.25	500-1000**	200	.25	1000-500**
5	250	.25	1000*	250	.25	1000-500**	250	.25	1000-500**	250	.25	500-1000**
6	300	.25	1000*	300	.25	500-1000**	300	.25	500-1000**	300	.25	1000-500**
7	350	.25	1000*	350	.25	1000-500**	350	.25	1000-500**	350	.25	500-1000**
8	400	.25	1000*	400	.25	500-1000**	400	.25	500-1000**	400	.25	1000-500**
9	450	.25	1000*	450	.25	1000-500**	450	.25	1000-500**	450	.25	500-1000**
10	450	.25	1000*	50	.25	500-1000**	450	.25	500-1000**	50	.25	1000-500**

**Demand alternates the two values over the five periods

Table 11. Continued

Problem	10		
	S	h	D
Stage			
1	3000	.25	
2	100	.25	1000-500**
3	150	.25	500-1000**
4	200	.25	1000-500**
5	250	.25	500-1000**
6	300	.25	1000-500**
7	350	.25	500-1000**
8	400	.25	1000-500**
9	450	.25	500-1000**
10	450	.25	1000-500**

S - Set up cost (dollars)

h - Inventory carrying cost (dollars)

D - Demand (units)

**Demand alternates the two values over the five periods

Table 12. Production Schedule In Arborescence Structure, Time Varying Demand

Period	Problem 1					Problem 2					Problem 3				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	36000*	0	0	0	9000	36000	0	0	0	9000	18000	0	18000	0	9000
2	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
3	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
4	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
5	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
6	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
7	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
8	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
9	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
10	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000

*Production units

Table 12. Continued

Period	Problem 4					Problem 5					Problem 6				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	18000	0	18000	0	9000	18000	0	18000	0	9000	18000	0	18000	0	9000
2	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
3	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
4	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
5	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
6	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
7	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
8	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
9	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000
10	2000	0	2000	0	1000	2000	0	2000	0	1000	2000	0	2000	0	1000

Table 12. Continued

Period	Problem 7					Problem 8					Problem 9				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	22500	0	22500	0	10500	22500	0	22500	0	10500	22500	0	22500	0	12000
2	2500	0	2500	0	500	2500	0	2500	0	500	2500	0	2500	0	2000
3	2500	0	2500	0	2000	2500	0	2500	0	2000	2500	0	2500	0	500
4	2500	0	2500	0	500	2500	0	2500	0	500	2500	0	2500	0	2000
5	2500	0	2500	0	2000	2500	0	2500	0	2000	2500	0	2500	0	500
6	2500	0	2500	0	500	2500	0	2500	0	500	2500	0	2500	0	2000
7	2500	0	2500	0	2000	2500	0	2500	0	2000	2500	0	2500	0	500
8	2500	0	2500	0	500	2500	0	2500	0	500	2500	0	2500	0	2000
9	2500	0	2500	0	2000	2500	0	2500	0	2000	2500	0	2500	0	500
10	2500	0	2500	0	500	2500	0	2500	0	500	2500	0	2500	0	2000

Table 12. Continued

Period	Problem 10				
	1	2	3	4	5
Stage					
1	22500	0	22500	0	12000
2	2500	0	2500	0	2000
3	2500	0	2500	0	500
4	2500	0	2500	0	2000
5	2500	0	2500	0	500
6	2500	0	2500	0	2000
7	2500	0	2500	0	500
8	2500	0	2500	0	2000
9	2500	0	2500	0	500
10	2500	0	2500	0	2000

There is a variation on stage one's schedule when the set up cost is decreased from 5000 to 3000. Instead of producing in periods 1 and 5, it produces in periods 1, 3, and 5.

Variations on the inventory carrying costs have no effect. The variability range in which we are working is too small. The model requires greater changes to show variability because of its small elasticity.

Larger values on demands produce larger lot sizes.

Acyclic Structure

Problem Definition

In previous sections, computational algorithms have been developed for the cases where the stages are arranged in an assembly or arborescence structure, allowing just one stage at either level two or level one respectively. Now we consider the general acyclic structure, illustrated in Figure 21, where the maximum number of relations have been allowed. Figure 21 is an extension of Figure 14 in Chapter II.

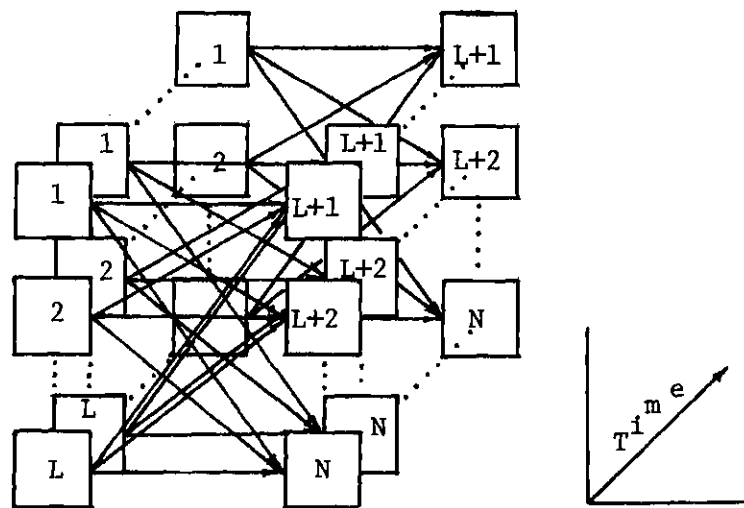


Figure 21. Dynamic Acyclic Structure

It is important to recognize that the arrows are optional relations between stages at level 2 and stages at level 1 that may or may not exist. It is not a requisite that a particular stage at level 2 has relations with all stages at level 1.

Considering this completely general structure, any stage n at level 2 may have two different inputs, one from its suppliers and the other from inventory carried by itself from a preceding period. The outputs will be deliveries to all the stages at level 1 which facility n has relation with and also storage inventory carried forward for demand in future periods. A stage at level 1 receives material from all its related stages from level 2 or from inventory coming from previous periods, and ships material to satisfy its external demand and/or ships to the next period in the form of inventory.

The inventories at the beginning and at the end of the horizon time are assumed to be zero since all demands must be accomplished. Then production always takes place at the first period of time.

Let us now represent mathematically the problem

$$\text{minimize } T = \sum_{n=1}^N \sum_{i=1}^M \delta (Q_{ni}) S_n + \sum_{n=1}^N \sum_{i=1}^M h_n I_{ni} \quad (3.53)$$

subject to

$$I_{ni} = I_{n,i+1} + Q_{n,i} - \sum_{m \in a(n)} Q_{m,i} \quad n = 1, 2, \dots, L \quad i = 1, 2, \dots, M$$

$$I_{ni} = I_{n,i-1} + Q_{n,i} - D_{n,i} \quad n = L+1, \dots, N \quad i = 1, 2, \dots, M \quad (3.54)$$

$$\delta(Q) = \begin{cases} 1 & \text{if } Q > 0 \\ 0 & \text{if } Q = 0 \end{cases}$$

The constraints on the inventory vary according with the position of the n stage. At level 2 it will have an output which is the requirements from all the $a(n)$ successors, and at level 1 the output is the corresponding demand for that product at that specific period of time.

Literature Survey and Results

The acyclic structure has apparently not been yet extensively investigated. Only one paper dealing with this structure (Zangwill [24]) was found in the literature. He characterizes the optimal solution, but does not present any further analysis.

Analysis

The following is an heuristic methodology based on the properties developed in previous models.

Consider the "nested" properties suggested by Love [16], then:

$$I_{n,i-1} \cdot Q_{n,i} = 0 \quad i = 1, 2, \dots, M \quad n = 1, \dots, N \quad (3.55)$$

$$Q_{n,i} > 0 \text{ implies } Q_{a(n),i} > 0 \quad \text{for } i = 1, \dots, M \quad n = 1, 2, \dots, N \quad (3.56)$$

Assume production costs are non-increasing in time

$$C_{n,i}(Q) \leq C_{n,i-1}(Q) \quad \text{for } n = 1, \dots, N \quad i = 2, \dots, M \quad (3.57)$$

and inventory holding costs are non-increasing over stages

$$h_{n,i} \leq h_{a(n),i} \quad (3.58)$$

Let $\Pi (\Pi_1, \Pi_2, \dots, \Pi_M)$ be a vector of 1's and 0's. We interpret Π as a production profile vector, such that if the profile Π is chosen for stage n ,

$$\begin{aligned} Q_{n,i} &= 0 \text{ if } \Pi_i = 0 \\ Q_{n,i} &> 0 \text{ if } \Pi_i = 1 \end{aligned} \quad (3.59)$$

Then according with the optimal solution form

$$\begin{aligned} Q_{n,i} &= 0 && \text{if } \Pi_i = 0 \\ Q_{n,i} &= D_i && \text{if } \Pi_i = 1 \text{ and } \Pi_{i+1} = 1 \\ Q_{n,i} &= D_i + D_{i+1} && \text{if } \Pi_i = 1, \Pi_{iM} = 0 \text{ and } \Pi_{i+2} = 1 \end{aligned} \quad (3.60)$$

and in general

$$Q_{n,i} = \sum_{t=1}^{i'} D_t \text{ if } \Pi_i = 1, \Pi_t = 0 \text{ for } t = i+1, \dots, i' \text{ and } \Pi_{i'+1} = 1 \quad (3.61)$$

Let us denote the total production cost associated with the choice of production profile Π for stage n by $C_n(\Pi)$. Thus

$$C_n(\Pi) = \sum_{i=1}^M C_{n,i}(Q_{n,i}) \quad (3.62)$$

The inventory would be

$$I_{n,i} = \sum_{t=1}^i (Q_{n,t} - D_t) \quad (3.63)$$

with a total carrying inventory cost for stage n

$$h_n(\Pi) = \sum_{i=1}^M h_{n,i} I_{n,i} \quad (3.64)$$

Using the nested schedules property, let $N(\Pi)$ be the set of production profiles which "nest" Π , that is

$$\Pi' \in N(\Pi) \text{ if and only if } \Pi_i - \Pi'_i \leq 0 \quad i = L+1, L+2, \dots, N \quad (3.65)$$

and let $P(\Pi)$ be the set of production profiles which are "nested" by Π , that is

$$\Pi'' \in P(\Pi) \text{ if and only if } \Pi_i - \Pi''_i \geq 0 \quad i = 1, 2, \dots, L \quad (3.66)$$

Define $f_n(\Pi)$ as the minimum total cost given that Π production profile has been chosen to be the stage n predecessor's or successor's schedule if n is at level 1 or level 2, respectively.

Π must be an element from the U universe, composed by 2^{M-1} elements. The reason the exponent is $M-1$ is because at the first period of time we must produce; that is, $\Pi_1 = 1$.

Then, the minimum cost for stages located in level 2, for a given Π , is

$$f_n(\Pi) = \text{minimum}_{\Pi'' \in P(\Pi)} \left[C_n(\Pi'') + h_n(\Pi'') \right] \quad n = 1, 2, \dots, L \quad (3.67)$$

and for those stages at level 1,

$$f_n(\Pi) = \text{minimum}_{\Pi' \in N(\Pi)} \left[C_n(\Pi') + h_n(\Pi') \right] \quad n = L+1, L+2, \dots, N \quad (3.68)$$

The total minimum cost will be

$$T^* = \text{optimal } T = \text{minimum}_{\Pi \in U} \left[\sum_{n=1}^N f_n(\Pi) \right] \quad (3.69)$$

Results

The above equations were programmed as THE 19. This program allows any structure in two levels with variations in demands. Again the extension for variations on the cost parameters is a matter of few program statements.

In contrast to previous programs, THE 19 accepts as input the number of stages at each level. After demands of stages at level 1, a matrix of relations between stages from level 2 and level 1 must be provided.

The acyclic structure selected to be tested in this study is presented in Figure 22. In this structure, stage 1 is related with stages 6, 8, and 10; stage 2 with 6, 7, and 8; stage 3 with 7 and 9; stage 4 with 6, 7, 8, 9, and 10; and stage 5 with 8, 9, and 10.

The problems to be tested appear in Table 13 and the results in Table 14.

Without a doubt, problems 1 and 2 show that the greater the set

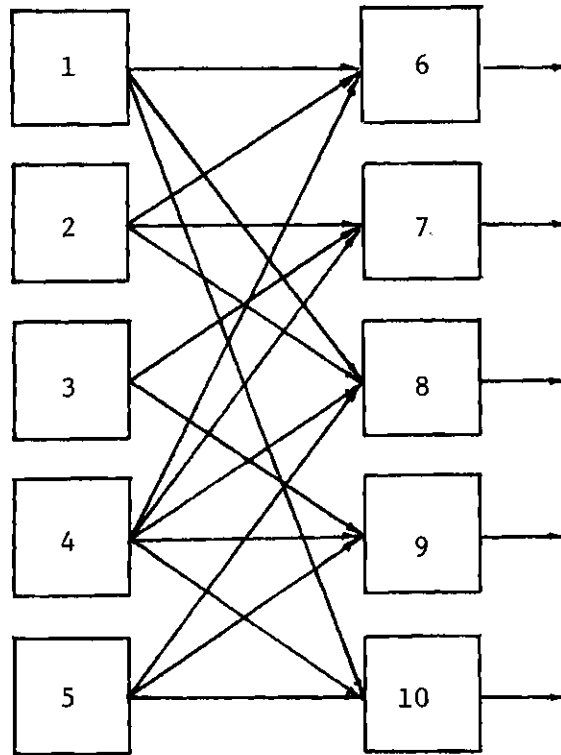


Figure 22. Acyclic Problem

up cost, the greater the inventories and logically the system tends to produce on fewer occasions.

On the other hand, problems 4 and 6, with a small variation of inventory carrying cost show a strong reduction on inventories.

As demand values increase (Problems 7, 8, 9, and 10), there is an increasing tendency to satisfy requirements with production rather than inventory.

Table 13. Problems For Acyclic Structure, Time Varying Demand

Problem	1			2			3			4			5		
	S	h	D	S	h	D	S	h	D	S	h	D	S	h	D
Stage															
1	5000	.15		5000	.15		3000	.15		3000	.15		3000	.25	
2	100	.15		100	.15		100	.15		100	.15		100	.25	
3	1000	.15		1000	.15		150	.15		150	.15		150	.25	
4	50	.15		50	.15		200	.15		200	.15		200	.25	
5	100	.15		100	.15		250	.15		250	.15		250	.25	
6	1000	.15	1000*	1000	.15	1000*	300	.15	1000*	300	.15	1000*	300	.25	1000*
7	50	.15	1000*	50	.15	1000*	350	.15	1000*	350	.15	1000*	350	.25	1000*
8	100	.15	1000*	100	.15	1000*	400	.15	1000*	400	.15	1000*	400	.25	1000*
9	1000	.15	1000*	1000	.15	1000*	450	.15	1000*	450	.15	1000*	450	.25	1000*
10	50	.15	1000*	1000	.15	1000*	50	.15	1000*	450	.15	1000*	50	.25	1000*

S - Set up cost (dollars)

h - Inventory carrying cost (dollars)

D - Demand (units)

*The demand is kept constant through the five periods

Table 13. Continued

Problem	6			7			8			9		
	S	h	D	S	h	D	S	h	D	S	h	D
Stage												
1	3000	.25		3000	.25		3000	.25		3000	.25	
2	100	.25		100	.25		100	.25		100	.25	
3	150	.25		150	.25		150	.25		150	.25	
4	200	.25		200	.25		200	.25		200	.25	
5	250	.25		250	.25		250	.25		250	.25	
6	300	.25	1000*	300	.25	500-1000**	300	.25	500-1000**	300	.25	1000-500**
7	350	.25	1000*	350	.25	1000-500**	350	.25	1000-500**	350	.25	500-1000**
8	400	.25	1000*	400	.25	500-1000**	400	.25	500-1000**	400	.25	1000-500**
9	450	.25	1000*	450	.25	1000-500**	450	.25	1000-500**	450	.25	500-1000**
10	450	.25	1000*	50	.25	500-1000**	450	.25	500-1000**	50	.25	1000-500**

**Demand alternates thw two values over the five periods

Table 13. Continued

Problem	10		
	S	h	D
Stage			
1	3000	.25	
2	100	.25	
3	150	.25	
4	200	.25	
5	250	.25	
6	300	.25	1000-500**
7	350	.25	500-1000**
8	400	.25	1000-500**
9	450	.25	500-1000**
10	450	.25	1000-500**

S - Set up cost (dollars)

h - Inventory carrying cost (dollars)

D - Demand (units)

**The demand is kept constant through the five periods

Table 14. Production Schedule For Acyclic Structure, Time Varying Demand

Period	Problem 1					Problem 2					Problem 3				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	9000*	0	0	6000	0	9000	0	0	6000	0	9000	0	0	6000	0
2	3000	6000	0	6000	0	3000	6000	0	6000	0	3000	3000	3000	6000	0
3	6000	0	0	4000	0	6000	0	0	4000	0	2000	2000	2000	4000	0
4	5000	10000	0	10000	0	5000	10000	0	10000	0	5000	5000	5000	1000	0
5	3000	6000	0	6000	0	3000	6000	0	6000	0	3000	3000	3000	6000	0
6	1000	2000	0	2000	0	1000	2000	0	2000	0	1000	1000	1000	2000	0
7	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	2000	0
8	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000	2000	0
9	1000	2000	0	2000	0	1000	2000	0	2000	0	1000	1000	1000	2000	0
10	1000	1000	1000	1000	1000	1000	2000	0	2000	0	1000	1000	1000	1000	1000

*Production units

Table 14. Continued

Period	Problem 4					Problem 5					Problem 6				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	9000	0	0	6000	0	12000	0	0	0	3000	12000	0	0	0	3000
2	3000	3000	3000	6000	0	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000
3	2000	2000	2000	4000	0	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
4	5000	5000	5000	10000	0	5000	5000	5000	5000	5000	5000	5000	5000	5000	5000
5	3000	3000	3000	6000	0	3000	3000	3000	3000	3000	3000	3000	3000	3000	3000
6	1000	1000	1000	2000	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
7	1000	1000	1000	2000	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
8	1000	1000	1000	2000	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
9	1000	1000	1000	2000	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
10	1000	1000	1000	2000	0	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Table 14. Continued

Period	Problem 7					Problem 8					Problem 9				
	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
Stage															
1	9000	0	0	6000	1500	9000	0	0	7500	0	7500	0	7500	0	6000
2	3000	4500	3000	4500	3000	3000	4500	3000	7500	0	4500	3000	4500	3000	4500
3	4000	1000	4000	1000	4000	4000	1000	4000	5000	0	1000	4000	1000	4000	1000
4	5500	7000	5500	7000	5500	5500	7000	5500	12500	0	7000	5500	7000	5500	7000
5	3000	4500	3000	4500	3000	3000	4500	3000	7500	0	4500	3000	4500	3000	4500
6	500	2000	500	2000	500	500	2000	500	2500	0	2000	500	2000	500	2000
7	2000	500	2000	500	2000	2000	500	2000	2500	0	500	2000	500	2000	500
8	500	2000	500	2000	500	500	2000	500	2500	0	2000	500	2000	500	2000
9	2000	500	2000	500	2000	2000	500	2000	2500	0	500	2000	500	2000	500
10	500	2000	500	2000	500	500	2000	500	2500	0	2000	500	2000	500	2000

Table 14. Continued

Period	Problem 10				
	1	2	3	4	5
Stage					
1	7500	0	7500	0	6000
2	4500	3000	4500	3000	4500
3	1000	4000	1000	4000	1000
4	7000	5500	7000	5500	7000
5	4500	3000	4500	3000	4500
6	2000	500	2000	500	2000
7	500	2000	500	2000	500
8	2000	500	2000	500	2000
9	500	2000	500	2000	500
10	2000	500	2000	500	2000

CHAPTER IV

CONCLUSIONS AND RECOMMENDATIONS

Results of the Investigation

The problem of determining economically optimal lot sizes in two-level multistage production systems was studied under the following assumptions. Each stage produces a single item and each item is produced at a single stage. Costs at each stage are for production and inventory. No backlogging is permitted. Production costs are of the "fixed charge" type, where the cost of a lot consists of a fixed set up cost and a variable cost that is proportional to the lot size. Inventory costs are proportional to the average inventory level in the stationary demand, infinite horizon case and to the end-of-period inventory in the dynamic demand, finite horizon case. Only deterministic problems are considered.

Four system structures were investigated: series, assembly, arborescence, and general acyclic. In each case both stationary and time-varying demand situations were analyzed. The following results were obtained:

1. For the series system with stationary demand, two lot-size policies were considered. The first was the integer multiple policy, where the lot size at the second level stage is an integer multiple of the lot size at the first level stage, has been shown to be the optimal policy structure when equal size lots are to be produced at equal intervals of time at each stage. The second

policy analyzed permitted level two inventory to be maintained only during intervals when level one was producing. The optimal form of this policy has the lot size at level two an integer divisor of the lot size used at level one. Exact solutions for the optimal lot sizes were obtained for each policy under each of two conditions: simultaneous production and consumption of the level one lot. Analysis of conditions under which each policy structure is superior revealed that in general the integer multiple policy is better. Only for small set up cost ratios (S_1/S_2) or small inventory cost ratios (h_2/h_1) would the integer divisor policy be preferred. The effect of the utilization ratio (D/P_2) depends on whether or not production and consumption are simultaneous at level one (stage two). In the former case, the greater the ratio, the greater the preference for the integer divisor policy, while in the latter situation, the opposite is true.

2. For the assembly structure with stationary demand, seven algorithms were adapted or developed for the two level case. An integer multiple policy was used in each. Computer programs were written for each algorithm and were used to solve 10 test problems. In the dynamic programming algorithm suggested by Crowston, Wagner, and Williams [5], the main problem appears to be the bounding procedure. In three out of 10 problems tested the algorithm found the optimal solution, and in general the time used was 90% over that of the fastest algorithm. An alternative to dynamic programming which basically changes the computation of the integer multiplier and the

bounding procedure, obtains the best solution in four of the problems and it is 75% over the best computational time. Three heuristics are considered by Crowston, Wagner, and Henshaw [4]: Single Pass, Multiple Pass and Modified Multiple Pass. They reach the best solution 4, 8, and 9 times out of ten, being 64, 73, and 96 percent over the best time, respectively. A Branch and Bound algorithm as proposed by Schwarz and Schrage [24] was found to be accurate with the best solution seven times. However, it requires much computation time, as evidenced by its being 287 percent over the best time. Schwarz and Schrage in the same paper suggest myopic policies that consider just two stages at a time. This algorithm finds the best solution in 2 out of 10 cases, but is not far from the best in the other problems, and it obtains the best computational time. It was concluded that the myopic policy and the modified multiple pass algorithm are attractive solution methods.

3. The arborescence structure under stationary demand has been analyzed by Schwarz [23] and Graves and Schwarz [10]. The algorithms they proposed were adapted to the two-level case, programmed, and tested on 10 problems. The algorithm of Schwarz did not obtain the best solution for any of the problems tested, and was 27 percent over the best computational time. Graves and Schwarz propose a branch and bound algorithm which found the best solution six times, the solution time was high. It was 190 percent over the fastest algorithm. In the same paper Graves and Schwarz give a myopic policy for the case of arborescence structure. They also propose an improvement using the output of the algorithm as a new input

until no improvement is achieved. These algorithms (myopic and modified myopic) find the best solution, one and six times respectively; and they are zero and 27 percent over the best time. Finally a dynamic programming approach developed in the research appears to give fairly good results even though it never achieves the best solution; however, it is not particularly fast, being 172 percent over the best time.

4. The literature does not contain solution algorithms for the acyclic structure with stationary demand. An algorithm was developed that is based on assuming that the cycle times for stages on level two are an integer multiple of a given cycle time, while for stages at level one they are an integer divisor of that cycle time. Using a search on the cycle time, the lowest total cost is selected as the best alternative. The algorithm is tested for 10 problems and for a specific structure and results are feasible and appear to be good.

5. The series structure with time varying demand case is considered by Zangwill [30] where he proposes a network-based dynamic programming approach. For the two level cases the algorithm was programmed and tested with 10 problems. The optimal solution was obtained in each case with good computation time performance.

6. The assembly structure problem with time varying demand has been considered by Crowston and Wagner [7], who suggest a dynamic programming approach. This approach was adapted to the two level case and an algorithm developed and tested, obtaining satisfactory results.

7. The arborescence structure under time varying demand case has been treated by Veinott[24]. In this paper, he defines the characteristics of the optimal solution through the use of the Leontief substitution matrix and proposes a dynamic programming solution. His concepts were used in developing an algorithm for the two level structure. It was programmed and tested. The conclusion is that the algorithm is a satisfactory means of solving this type of problem.

8. The optimal solution for the acyclic structure under time varying demand is characterized by Zangwill [28, 29], but he does not suggest any solution algorithm. In his research, using Zangwill's characteristics and Love's [16] "nested" properties, an algorithm allowing any kind of two level acyclic structure is developed and tested, obtaining good results.

Recommendations

Recommendations for the use of these algorithms and for further studies are:

1. Further studies on acyclic structure and stationary demand should be oriented toward improving the speed of convergence as well as the bounding procedure.
2. The acyclic structure and time varying demand algorithm may also be improved by increasing the convergence time. This could probably be done by using the dynamic programming idea developed by Crowston and Wagner in [7].
3. Comparison of these methods with the heuristic methods currently proposed for use in multilevel production system (that is in material

requirements planning algorithm) would be desirable.

APPENDICES

APPENDIX A

This Appendix contains figures comparing integer multiple policy versus integer divisor policy in series structure. The input data are:

S_1 - Set up cost at stage 1 = 280 (dollars)

h_1 - Inventory carrying cost at stage 1 = 2 (dollars)

D - Demand at stage 2 = 1000 (units per time)

T - Production lead time = .00% (time units)

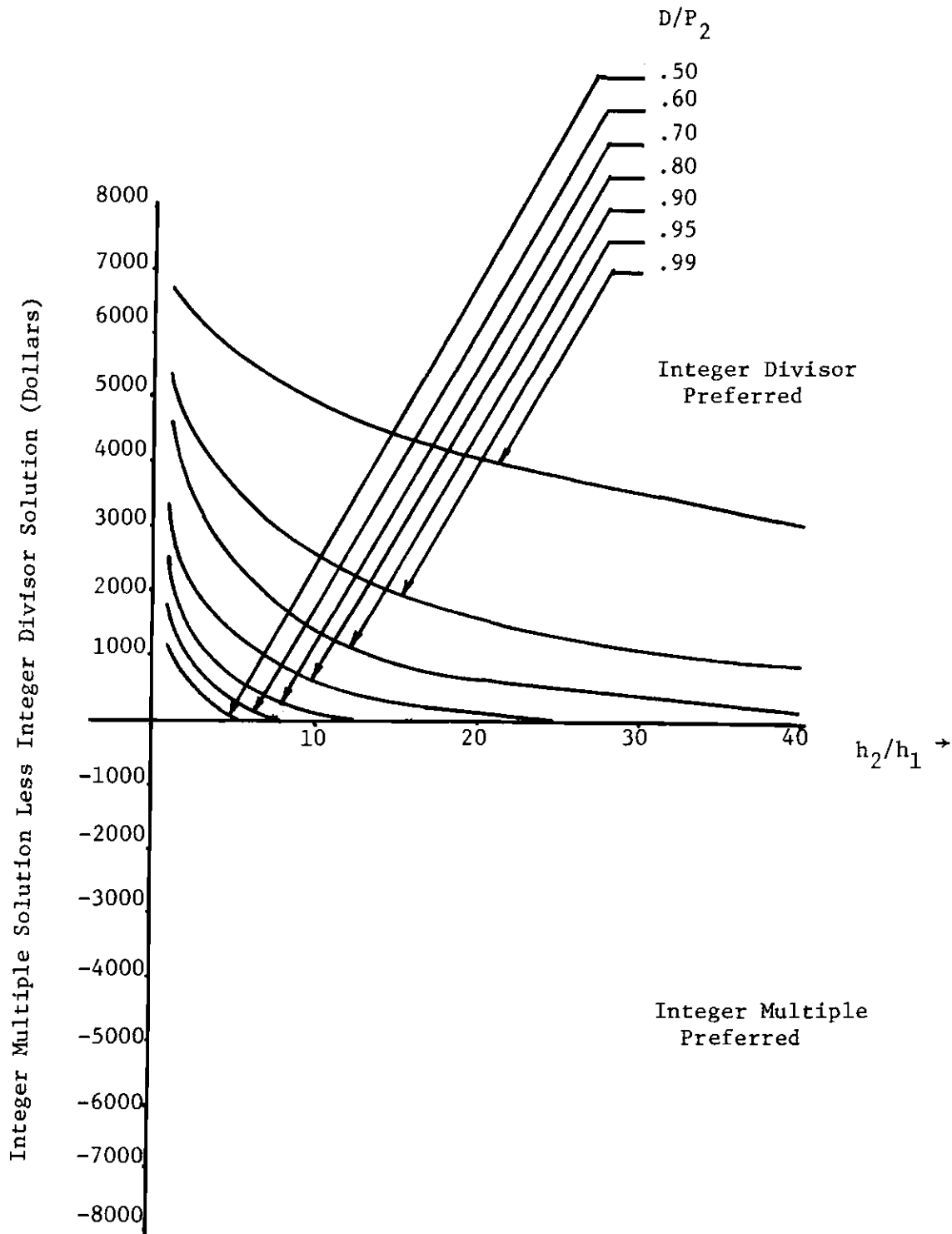


Figure A-1. $S_1/S_2 = .1$ Simultaneous Production & Consumption

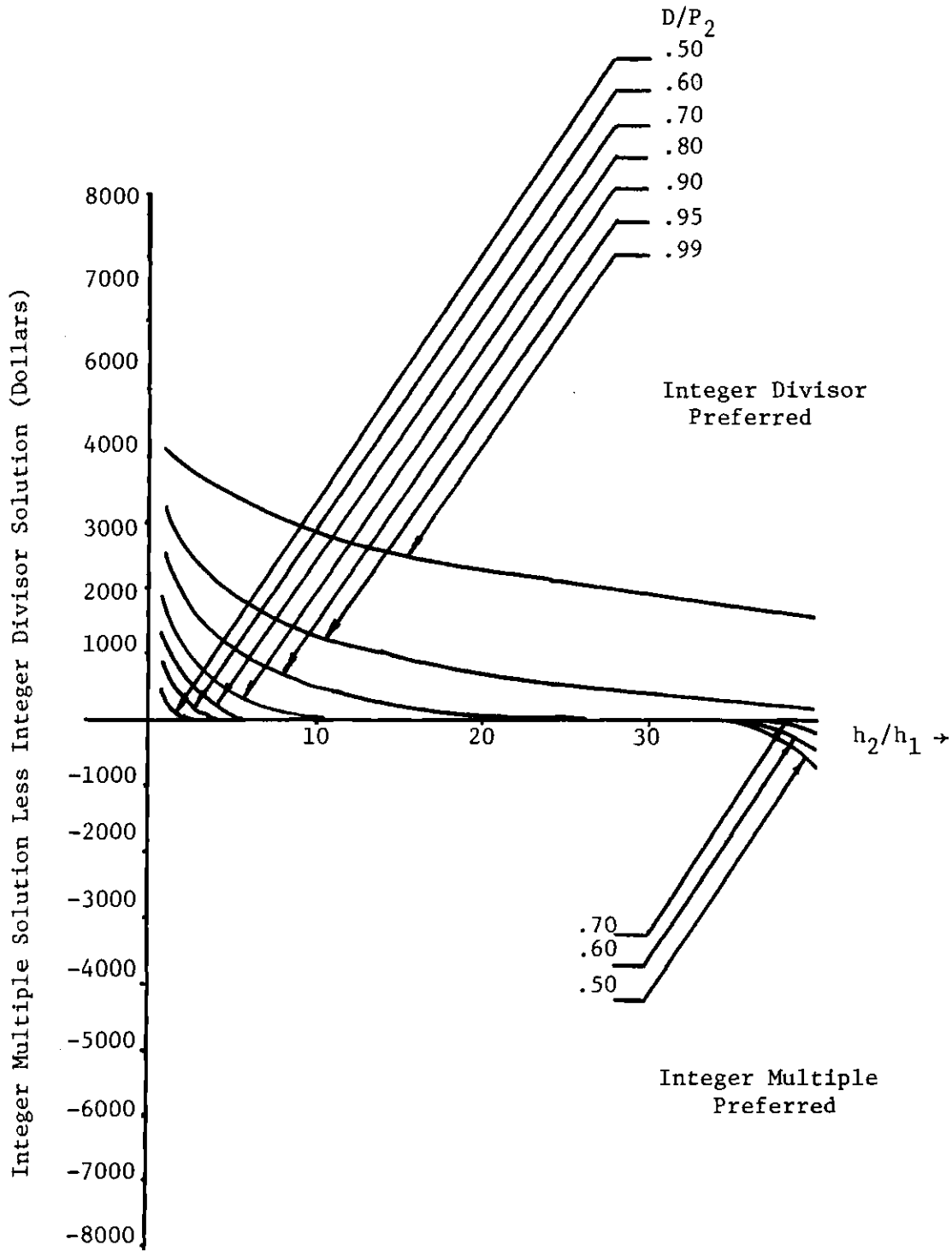


Figure A-2. $S_1/S_2 = .20$ Simultaneous Production & Consumption

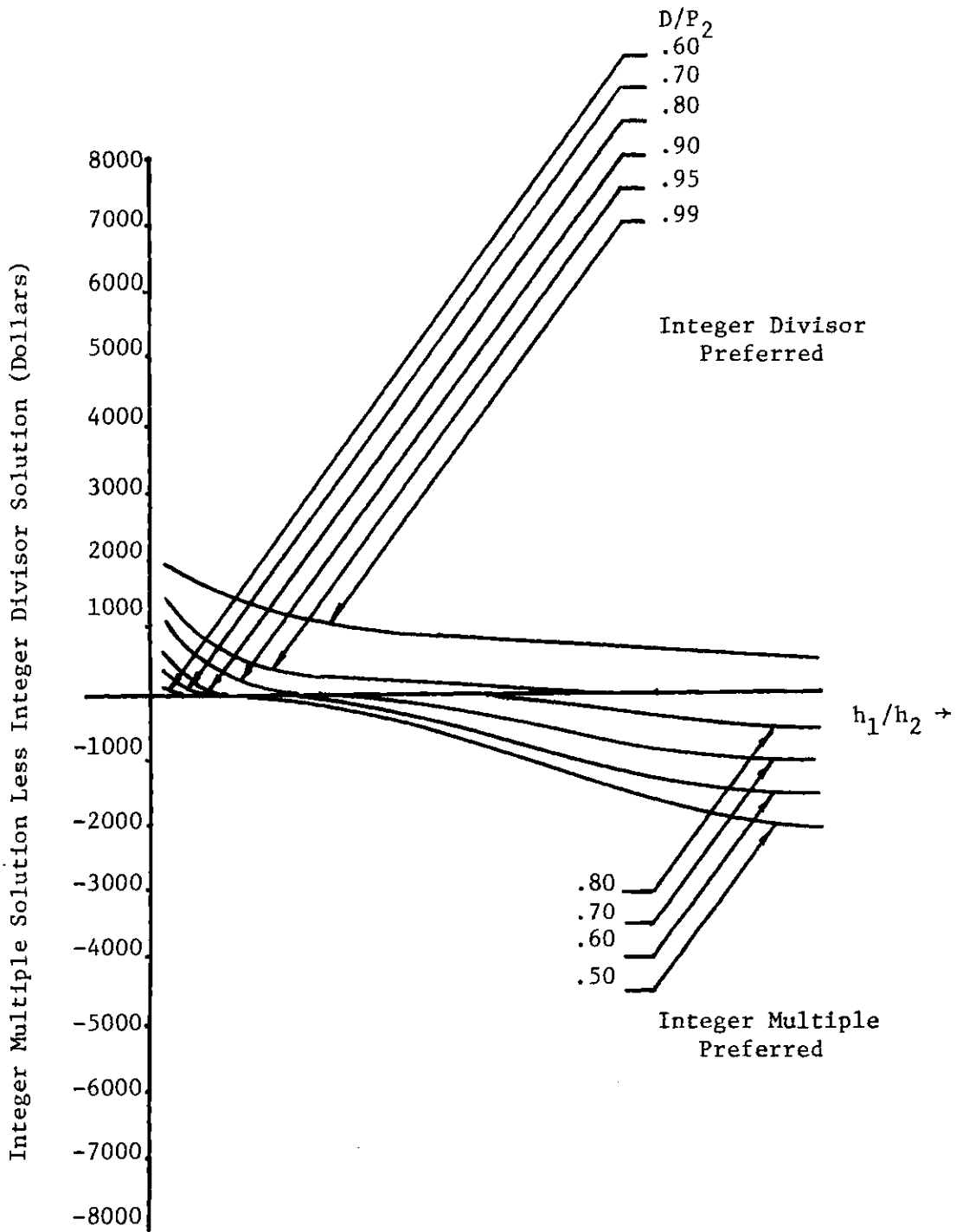


Figure A-3. $S_1/S_2 = .50$ Simultaneous Production & Consumption

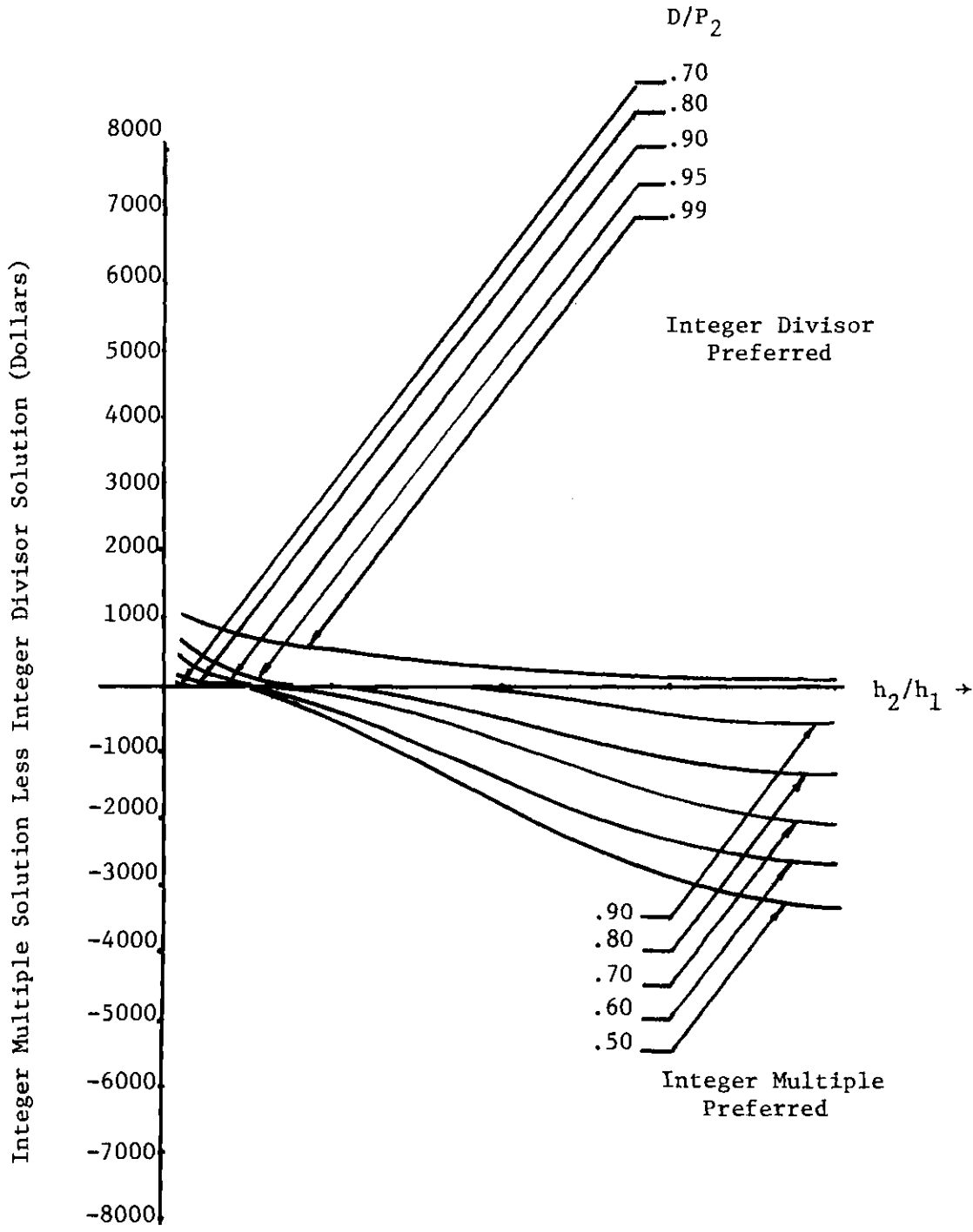


Figure A-4. $S_1/S_2 = 1$ Simultaneous Production & Consumption

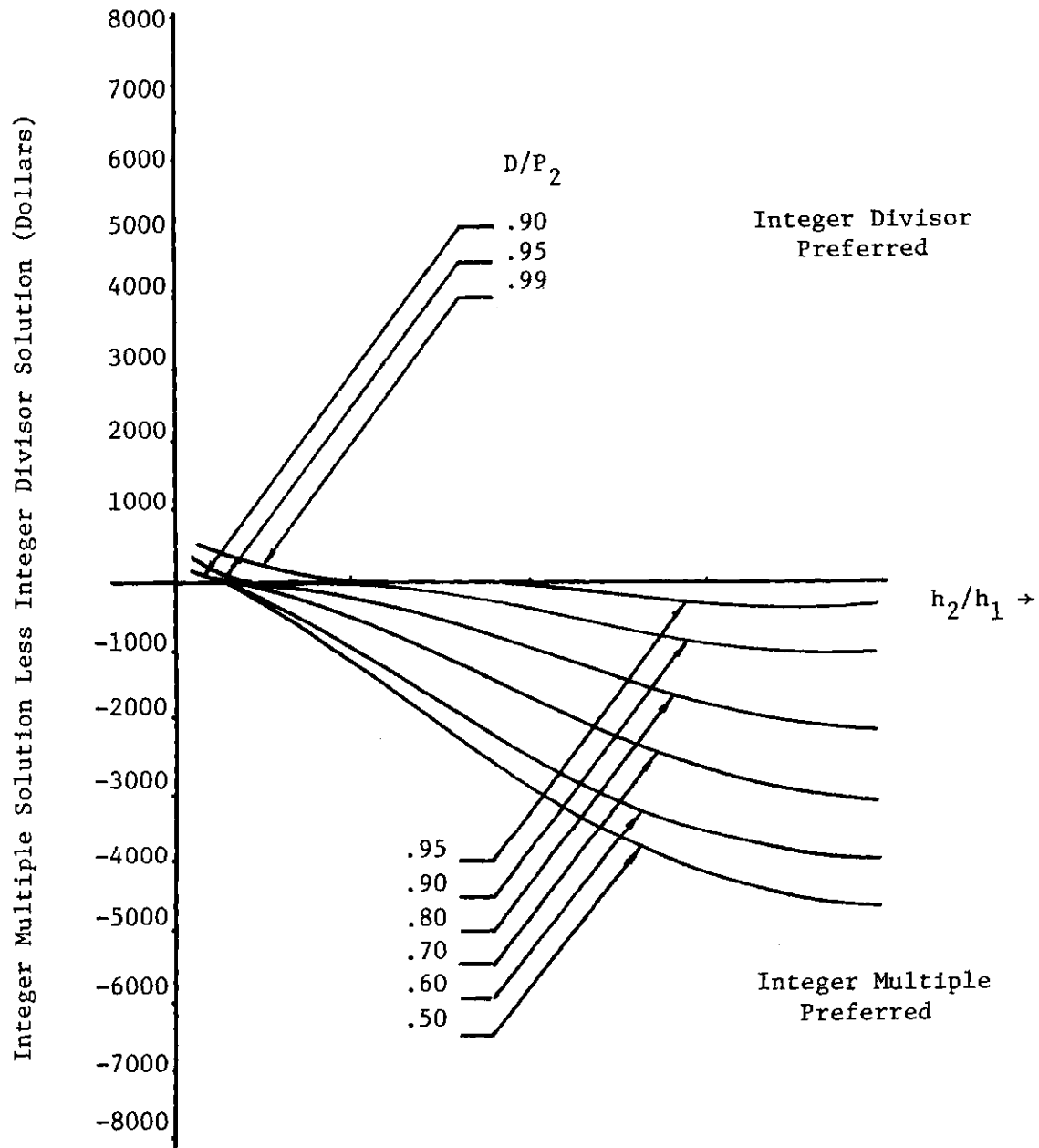


Figure A-5. $S_1/S_2 = 2$ Simultaneous Production & Consumption

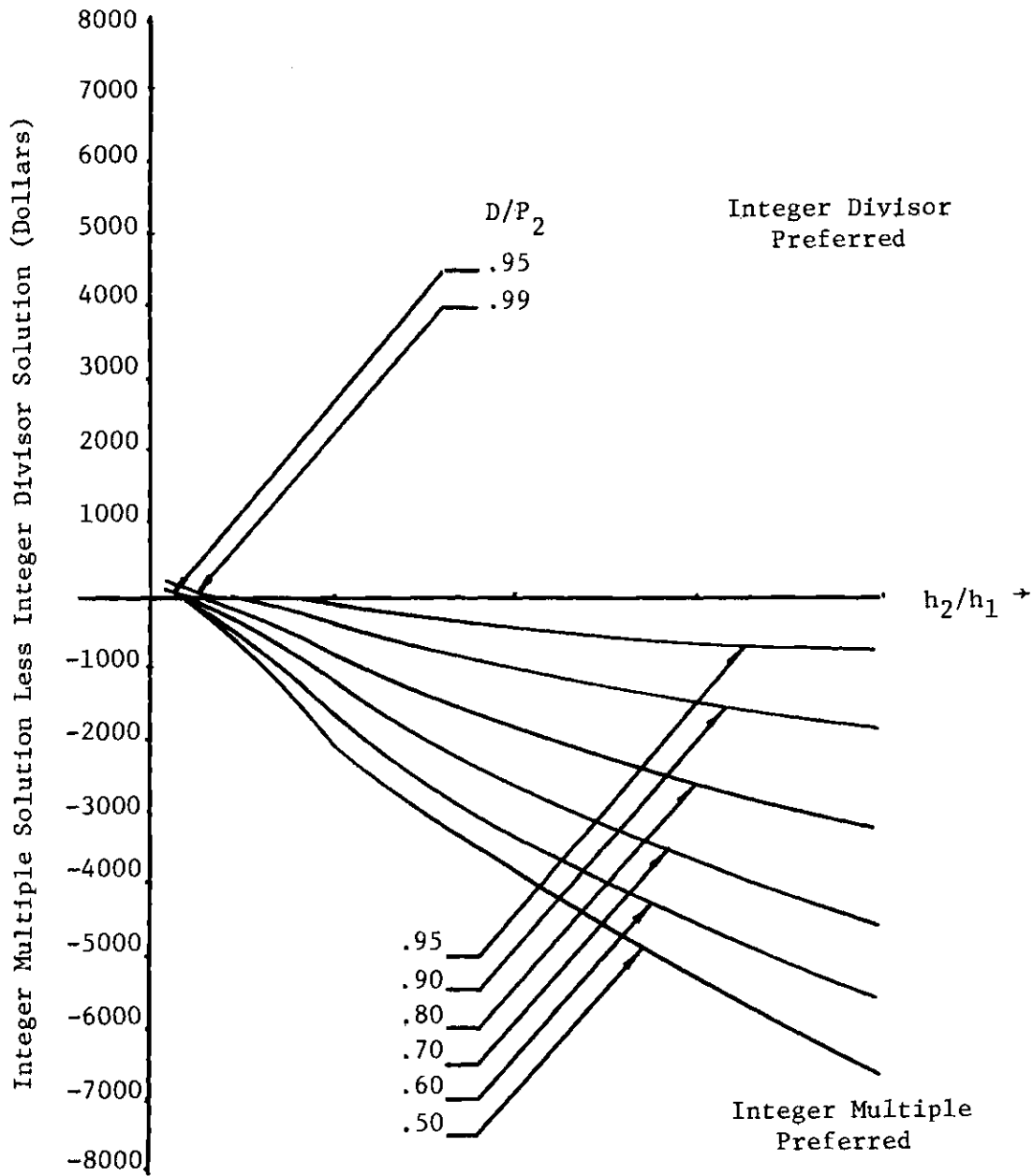


Figure A-6. $S_1/S_2 = 5$ Simultaneous Production & Consumption

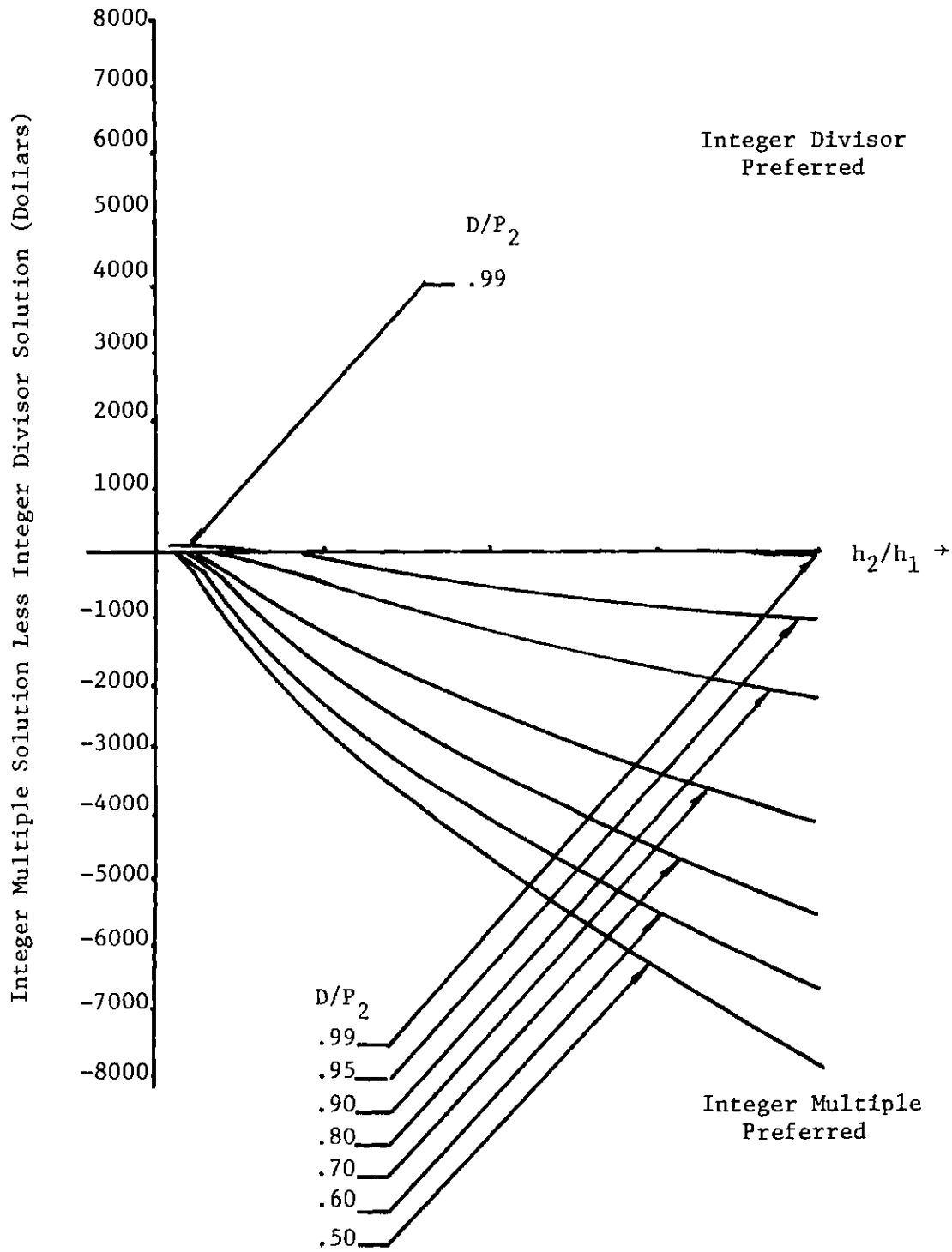


Figure A-7. $S_1/S_2 = 10$ Simultaneous Production & Consumption

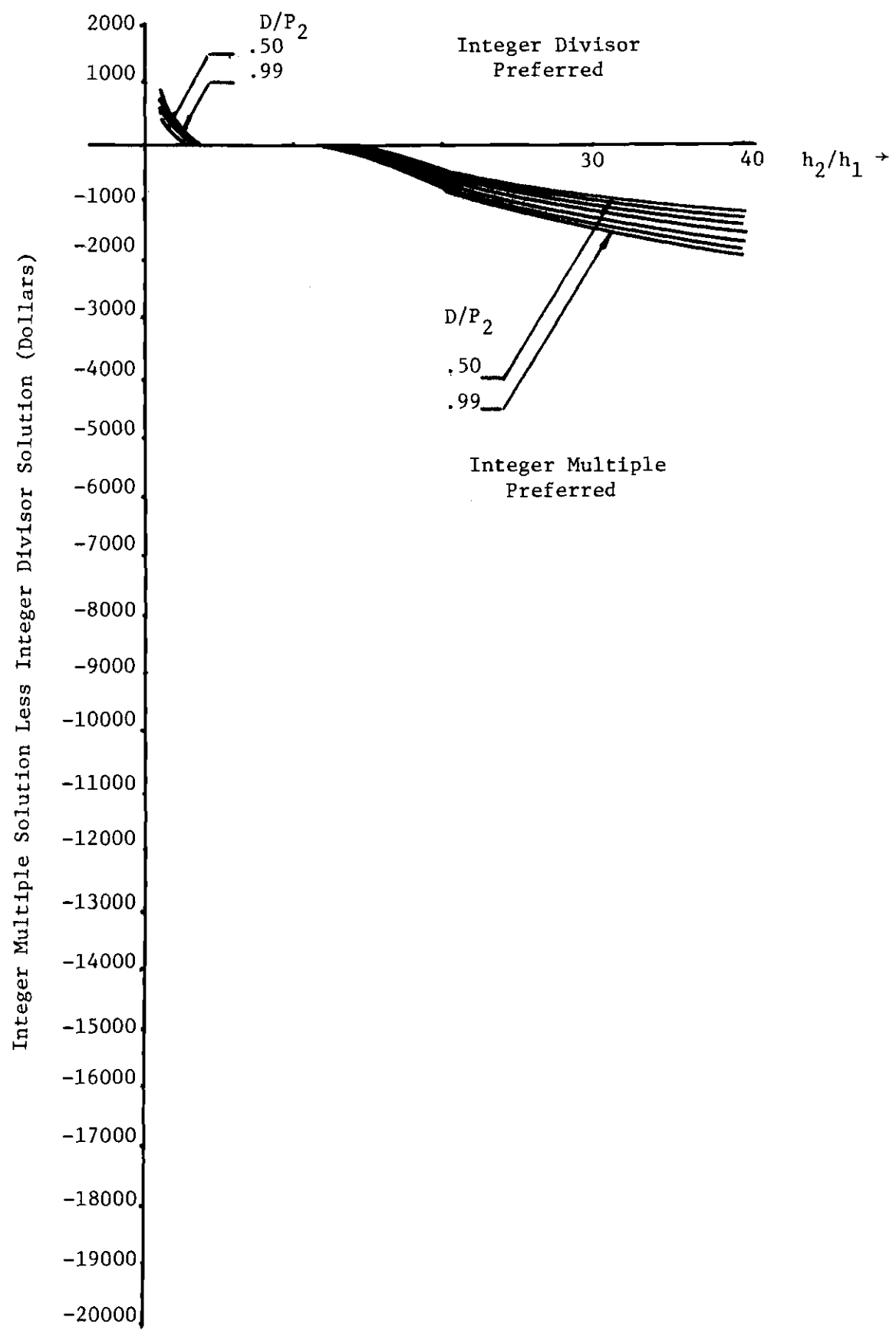


Figure A-8. $S_1/S_2 = .10$ Nonsimultaneous Production & Consumption

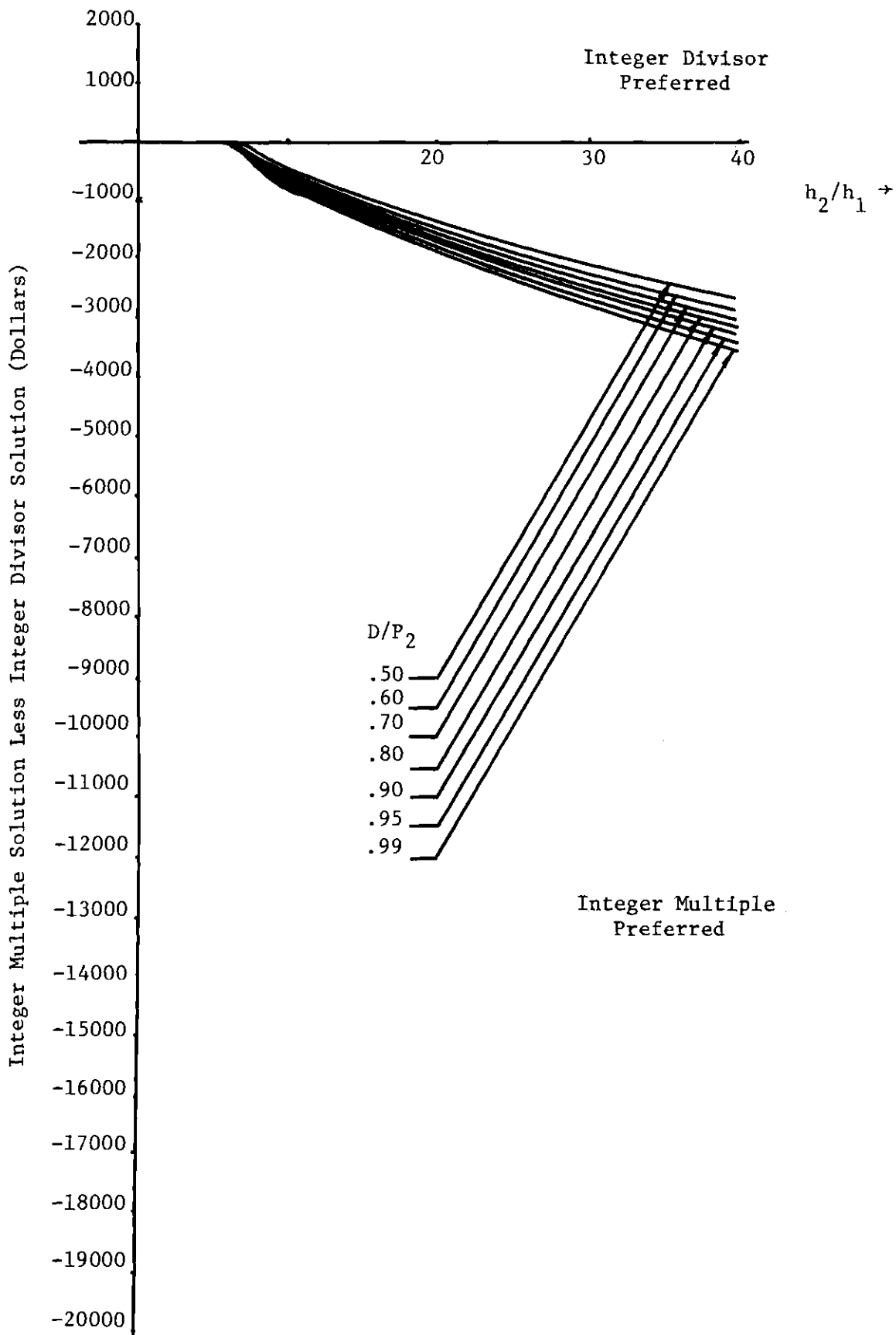


Figure A-9. $S_1/S_2 = .20$ Nonsimultaneous Production & Consumption

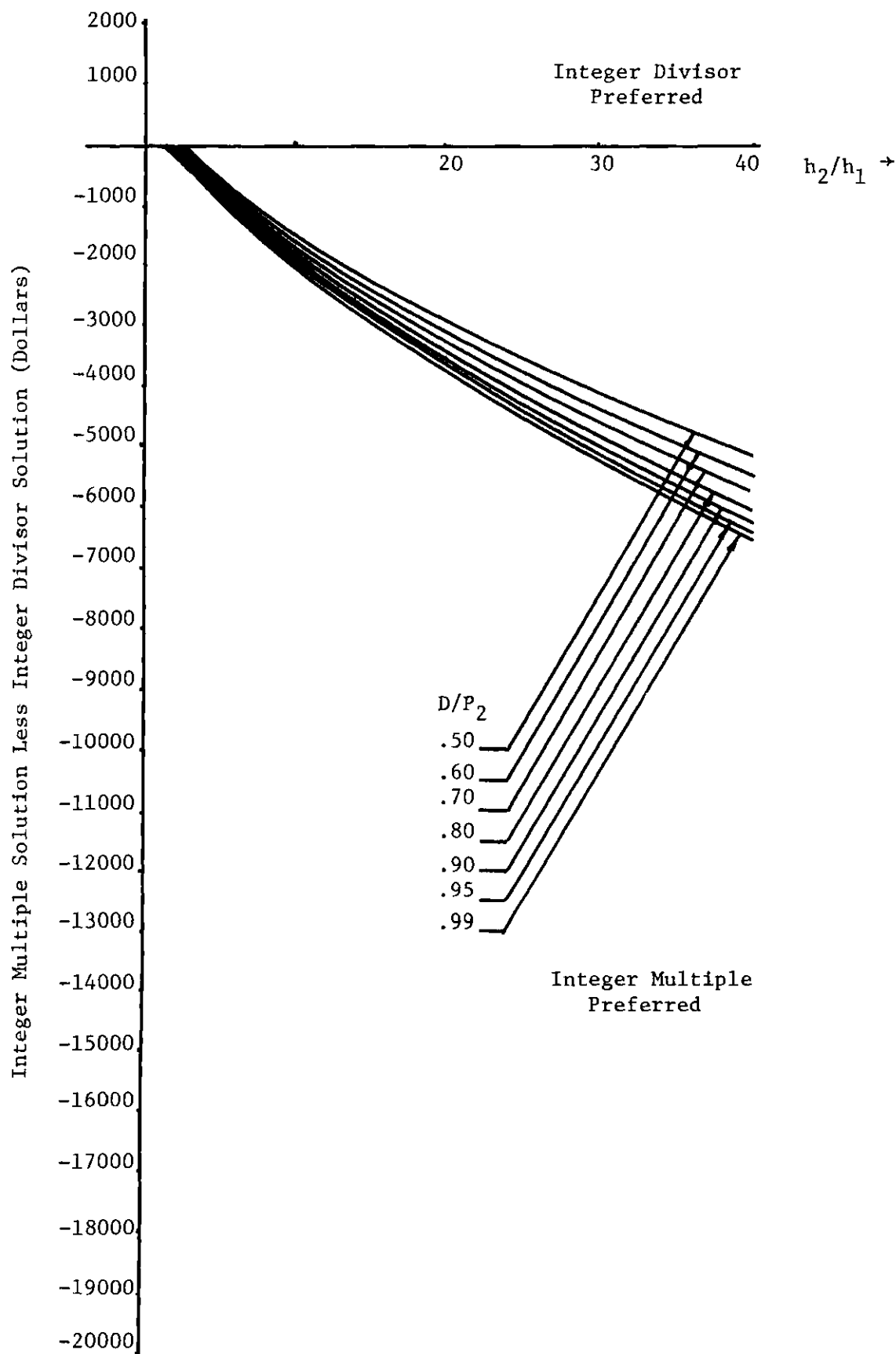


Figure A-10. $S_1/S_2 = .50$ Nonsimultaneous Production & Consumption

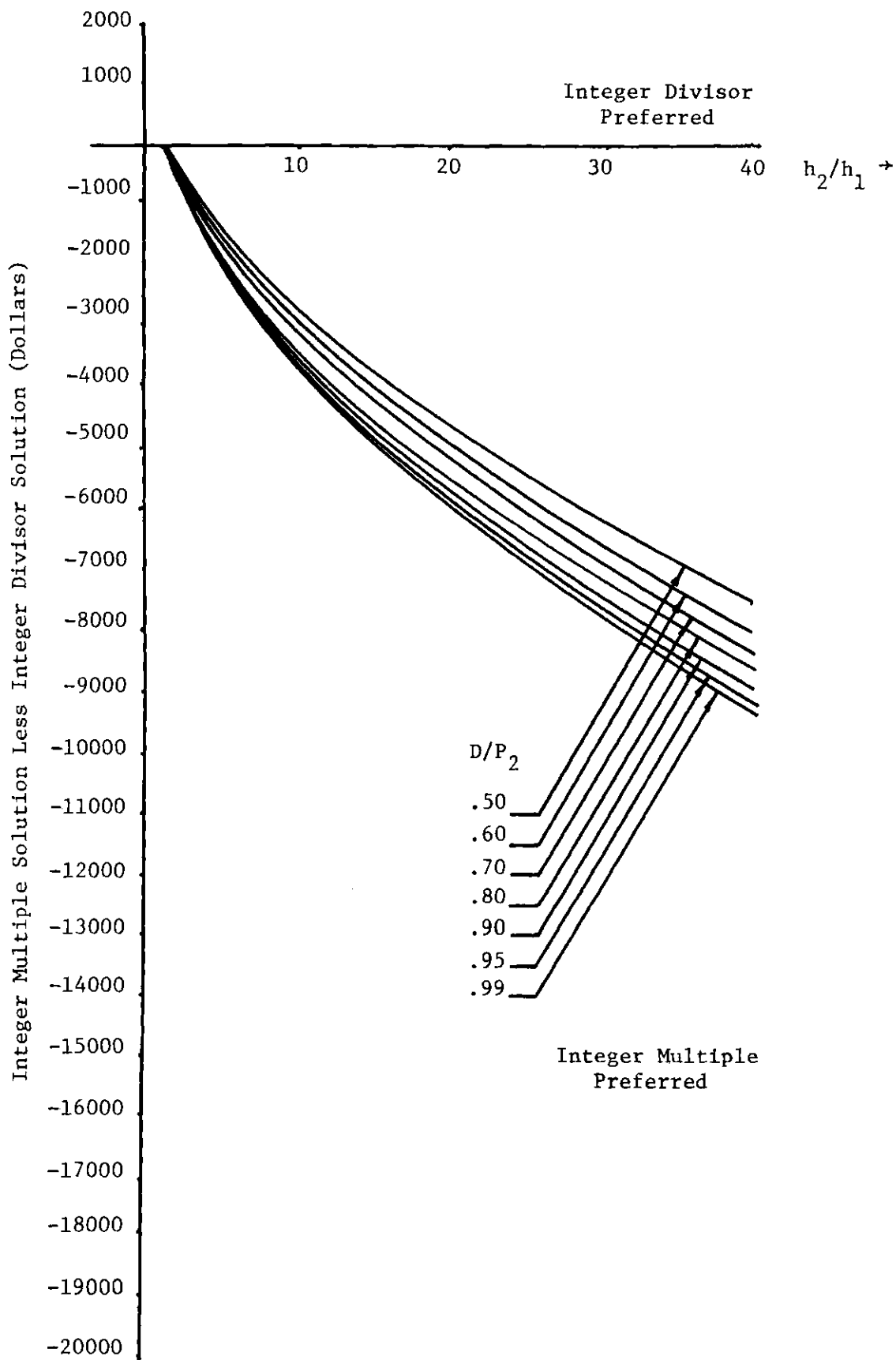


Figure A-11. $S_1/S_2 = 1$ Nonsimultaneous Production & Consumption

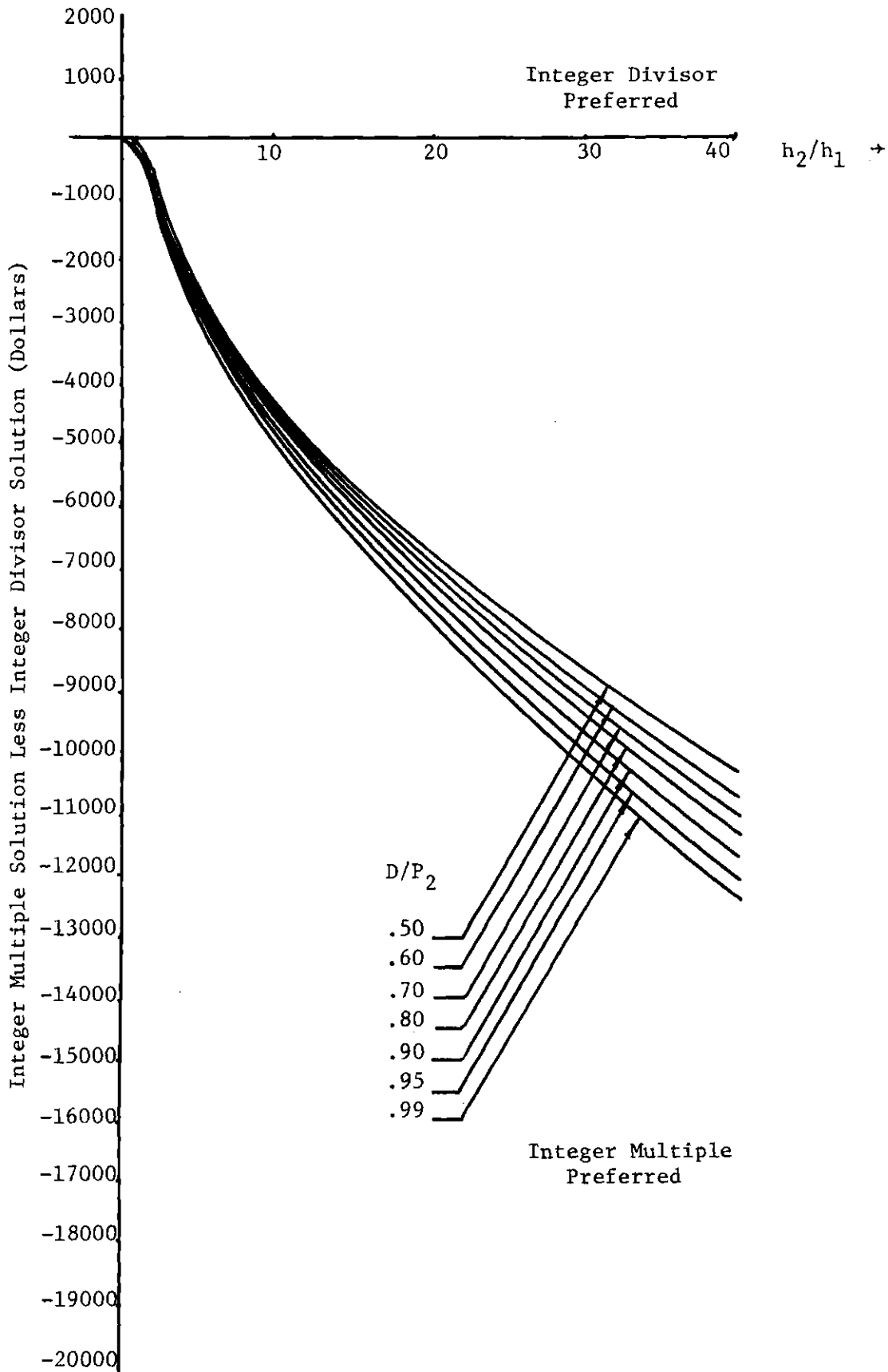


Figure A-12. $S_1/S_2 = 2$ Nonsimultaneous Production & Consumption

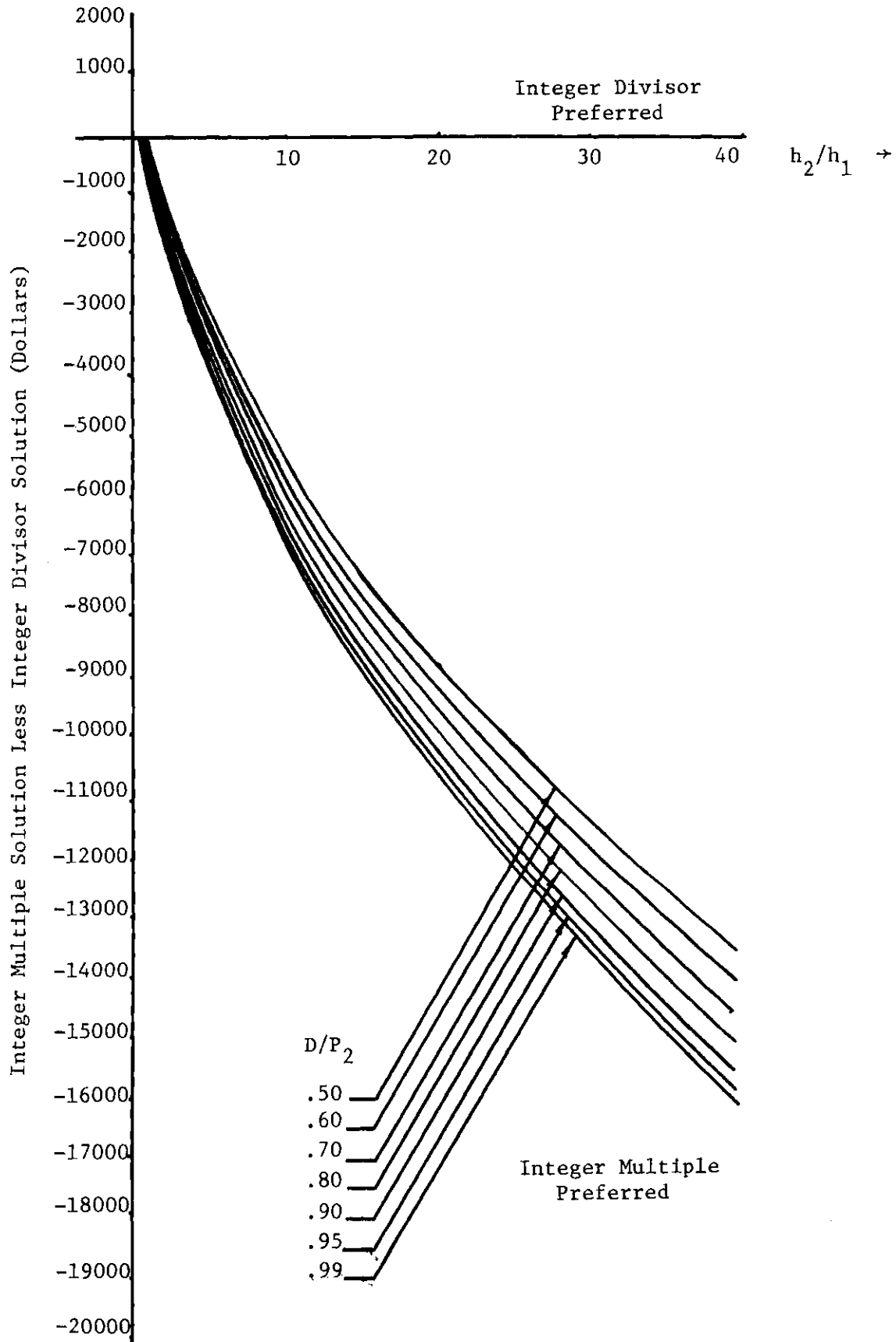


Figure A-13. $S_1/S_2 = 5$ Nonsimultaneous Production & Consumption

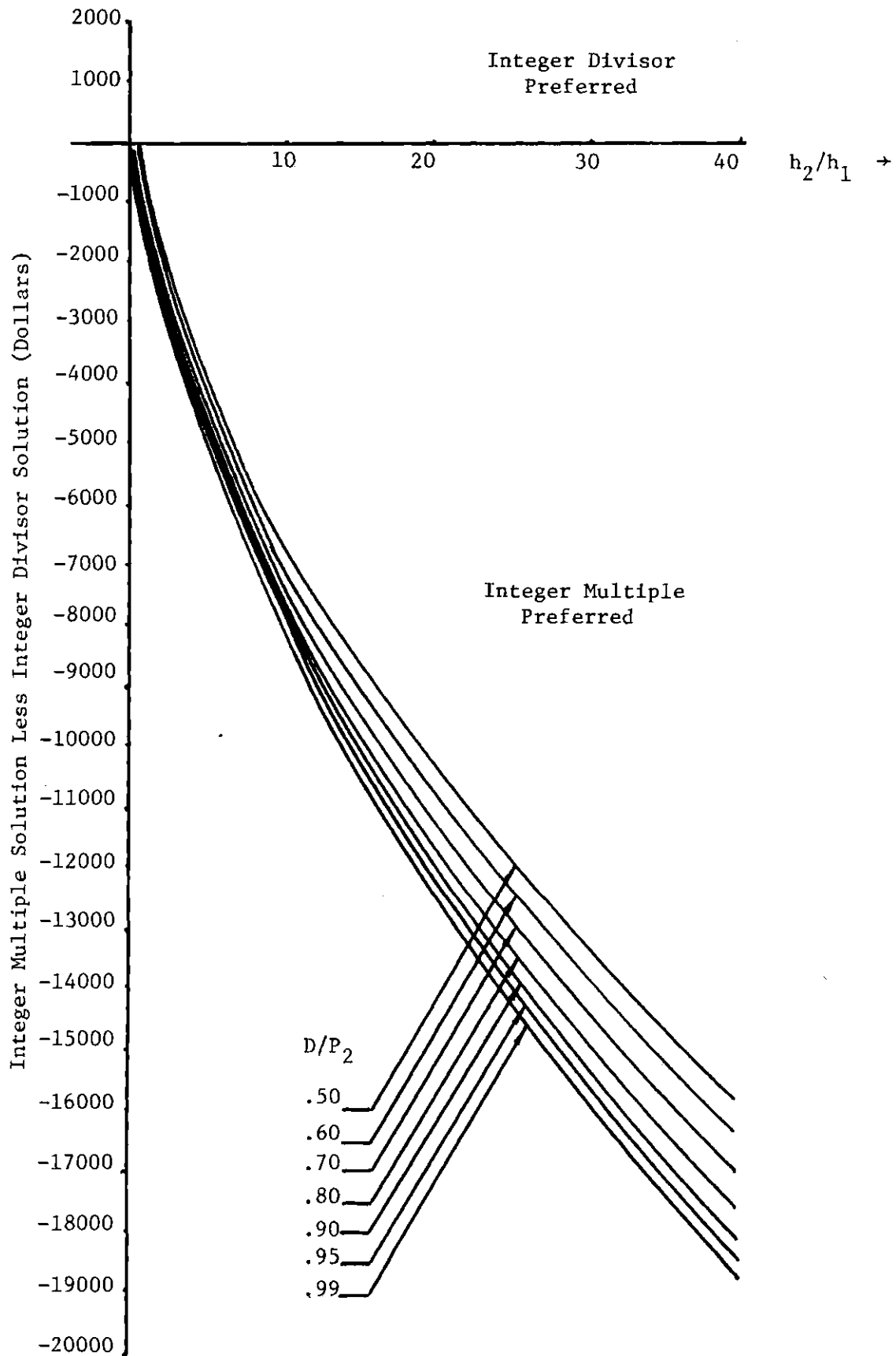


Figure A-14. $S_1/S_2 = 10$ Nonsimultaneous Production & Consumption

APPENDIX B

T H E S I S 1

```

PROGRAM THE1 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C THIS PROGRAM OBTAINS THE OPTIMAL LOT SIZE IN THE CASE OF AN ASSEMBLY
C STRUCTURE, SINGLE ITEM, STATIONARY DEMAND, CONTINUOUS REVIEW, IN-
C FINITE HORIZON. THE METHODOLOGY USED IN THE ALGORITHM IS A DYNAMIC
C PROGRAMMING SUGGESTED BY W. B. CROWSTON, M. H. WAGNER, AND A. HENSHAW
C THIS PROGRAM REQUIRES THE USE OF FUNCTION "RED" FOUND AT THE
C END OF PROGRAM "THESIS 4"

```

C

C

C INPUT VARIABLES

C N = NUMBER OF STAGES

C S = SET UP COST

C H = INVENTORY CARRYING COST

C DEM = DEMAND

C OUTPUTS

C TOTAL COST

C Q = LOT SIZE

C

```

DIMENSION S(10),H(10),Q(10),IQ(10),ZN1(10),IQMIN(10),IQMAX(10)

```

```

DIMENSION LN(10),LNMIN(10)

```

```

READ(5,1)N

```

1 FORMAT(I2)

```

READ(5,2)(S(J),H(J),J=1,N)

```

2 FORMAT(F9.2,F6.3)

```

READ(5,3)DEM

```

3 FORMAT(F7.0)

```

Q(N)=SQRT(2*DEM*S(N)/H(N))

```

```

IQ(N)=RED(Q(N))

```

```

BOUNDL=0

```

```

BOUNDU=0

```

```

DO 4 I=1,N

```

```

ZN1(I)=DEM*S(I)/SQRT(2*DEM*S(I)/H(I))+SQRT(2*DEM*S(I)/H(I))*H(I)/2

```

```

BOUNDL=BOUNDL+ZN1(I)

```

```

ZN2=DEM*S(I)/IQ(N)+IQ(N)*H(I)/2

```

```

BOUNDU=BOUNDU+ZN2

```

```

WRITE(6,75)ZN1(I),ZN2

```

75 FORMAT(1H0,15X,"ZN1",F9.2,5X,"ZN2",F9.2)

4 CONTINUE

```

DO 5 I=1,N

```

```

QMIN1=SQRT(2*DEM*S(I)/H(I))

```

```

QMIN=AMAX1(QMIN1,Q(N))

```

```

IQMIN(I)=RED(QMIN)

```

```

CT=ZN1(I)+BOUNDU-BOUNDL

```

```

WRITE(6,76)CT

```

76 FORMAT(1H0,10X,"CT",F20.2)

```

QM11=(2*CT)**2

```

```

QM12=4*H(I)+2*DEM*S(I)

```

```

QM13=(2*CT+SQRT(QM11-QM12))/(2*H(I))

```

```

WRITE(6,74)QM11,QM12,QM13

```

74 FORMAT(1H0,15X,"QM11",F20.2,5X,"QM12",F20.2,5X,"QM13",F20.2)

```

QM1=QM13

```

```

IQMAX(I)=RED(QM1)

```

```

WRITE(6,83)IQMIN(I),IQMAX(I)

```

83 FORMAT(1H0,10X,"IQMIN",I6,10X,"IQMAX",I6)

5 CONTINUE

```

KKMA=IQMAX(N)

```

```

      KKMI=IQMIN(N)
      DO 6 IIQ=KKMI, KKMA
      FN1=DEM*S(N)/IIQ+IIQ*H(N)/2
      NN=N-1
      DO 7 I=1, NN
      L=0
11    L=L+1
      IQ(I)=L*IIQ
      IF(IQ(I).GT.IQMAX(I))GO TO 9
      FN2=DEM*S(I)/IQ(I)+IQ(I)*H(I)/2
      IF(L.LE.1)GO TO 10
      IF(FN2.GT.FN2MIN)GO TO 11
10    FN2MIN=FN2
      LN(I)=L
      GO TO 11
9     FN1=FN1+FN2MIN
7     CONTINUE
      IF(IIQ.EQ.IQMIN(N))GO TO 12
      IF(FN1.GT.FN1MIN)GO TO 6
12    CONTINUE
      DO 13 I=1, NN
      LNMIN(I)=LN(I)
13    CONTINUE
      FN1MIN=FN1
      IQNMIN=IIQ
6     CONTINUE
      DO 14 I=1, NN
      IQ(I)=LNMIN(I)*IQNMIN
14    CONTINUE
      IQ(N)=IQNMIN
      WRITE(6,15)FN1MIN
15    FORMAT(1H1,20X,"TOTAL COST = ",F12.2)
      DO 16 I=1, N
      WRITE(6,17)I, IQ(I)
17    FORMAT(1H0,40X,"Q(",I3,") = ",I5)
16    CONTINUE
      STOP
      END

```

T H E S I S 2

PROGRAM THE2 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF AN ASSEMBLY STRUCTURE
 C SINGLE ITEM, STATIONARY DEMAND, CONTINUOUS REVIEW, INFINITE HORIZON
 C THE METHODOLOGY USED IN THE ALGORITHM IS AN HEURISTIC METHOD CALLED
 C "SINGLE PASS" SUGGESTED BY W.B. CROWSTON, M.H. WAGNER, AND A. HENSHAW
 C THIS PROGRAM REQUIRES THE USE OF FUNCTION "RED" AND SUBROUTINE
 C "QNTN" FOUND AT THE END OF PROGRAM "THESIS 4"

C
 C

C INPUT VARIABLES
 C N = NUMBER OF STAGES
 C S = SET UP COST
 C H = INVENTORY CARRYING COST
 C DEM = DEMAND

C OUTPUTS
 C TOTAL COST
 C Q = LOT SIZE

C
 C

DIMENSION S(10),H(10),K(10),KQ(10),IQ(10)
 READ(5,1)N

```

1  FORMAT(I2)
   READ(5,2) (S(J),H(J),J=1,N)
2  FORMAT(F9.2,F6.3)
   READ(5,3) DEM
3  FORMAT(F7.0)
   DO 4 I=1,N
     K(I)=1
4  CONTINUE
   CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
   DO 5 I=1,N
     K0(I)=K(I)
5  CONTINUE
   TNJMIN=TNJ
   IQNJM=IQNJ
   J=N
12  J=J-1
7  K(J)=K(J)+1
   CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
   IF(TNJ.GE.TNJMIN)GO TO 6
   TNJMIN=TNJ
   IQNJM=IQNJ
   K0(J)=K(J)
   GO TO 7
6  K(J)=K(J)-1
   IF(K(J).EQ.1)GO TO 8
10  K(J)=K(J)-1
   IF(K(J).EQ.0)GO TO 9
   CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
   IF(TNJ.GE.TNJMIN)GO TO 9
   TNJMIN=TNJ
   IQNJM=IQNJ
   K0(J)=K(J)
   GO TO 11
9  K(J)=K(J)+1
8  CONTINUE
   IF(J.EQ.1)GO TO 11
   GO TO 12
11  DO 15 I=1,N
     IQ(I)=K0(I)*IQNJM
13  CONTINUE
   WRITE(6,14)TNJMIN
14  FORMAT(1H1,20X,"TOTAL COST = ",F12.2)
   DO 16 I=1,N
     WRITE(6,17)I,IQ(I)
17  FORMAT(1H1,40X,"Q(",I3,") = ",I5)
16  CONTINUE
   STOP
   END

```

T H E S I S 3

```

PROGRAM THES3 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF AN ASSEMBLY STRUCTURE,
C SINGLE ITEM, STATIONARY DEMAND, CONTINUOUS REVIEW, INFINITE HORIZON.
C THE METHODOLOGY USED IN THE ALGORITHM IS AN HEURISTIC METHOD CALLED
C "MULTIPLE PASS" SUGGESTED BY W.D.CROWSTON, M.H. WAGNER, AND JACK
C WILLIAMS.
C THIS PROGRAM REQUIRES THE USE OF FUNCTION "RED" AND SUBROUTINE
C "QNTN" FOUND AT THE END OF PROGRAM "THESIS 4"
C
C INPUT VARIABLES

```

```

C      N = NUMBER OF STAGES
C      S = SET UP COST
C      H = INVENTORY CARRYING COST
C      DEM = DEMAND
C      OUTPUTS
C      TOTAL COST
C      Q = LOT SIZE
C
C      DIMENSION S(10),H(10),K(10),KC(10),IQ(10)
1      READ(5,1)N
      FORMAT(I2)
2      READ(5,2)(S(J),H(J),J=1,N)
      FORMAT(F3.2,F6.3)
3      READ(5,3)DEM
      FORMAT(F7.2)
      DO 4 I=1,N
      K(I)=1
4      CONTINUE
      CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
      DO 5 I=1,N
      KC(I)=K(I)
5      CONTINUE
      TNJMIN=TNJ
      IQNJ=IQNJ
      L=1
15     J=N
12     J=J-1
7      K(J)=K(J)+1
      CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
      IF(TNJ.GE.TNJMIN)GO TO 6
      TNJMIN=TNJ
      IQNJ=IQNJ
      KI(J)=K(J)
      GO TO 7
6      K(J)=K(J)-1
      IF(K(J).EQ.1)GO TO 8
10     K(J)=K(J)-1
      IF(K(J).EQ.0)GO TO 9
      CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
      IF(TNJ.GE.TNJMIN)GO TO 9
      TNJMIN=TNJ
      IQNJ=IQNJ
      KC(J)=K(J)
      GO TO 15
9      K(J)=K(J)+1
8      CONTINUE
      IF(J.EQ.1)GO TO 11
      GO TO 12
11     IF(L.EQ.1)GO TO 13
      IF(TTNJ.EQ.TNJMIN)GO TO 14
13     TTNJ=TNJMIN
      L=L+1
      GO TO 15
14     DO 16 I=1,N
      IQ(I)=K(I)*IQNJ
16     CONTINUE
      WRITE(6,13)TNJMIN
18     FORMAT(2H1,2LX,"TOTAL COST = ",F12.2)
      DO 19 I=1,N
      WRITE(6,17)I,IQ(I)
17     FORMAT(1H0,4CX,"Q(",I3,"") = ",I5)
19     CONTINUE
      STOP

```

END

T H E S I S 4

PROGRAM THE4 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF AN ASSEMBLY STRUCTURES
 C SINGLE ITEM, STATIONARY DEMAND, CONTINUOUS REVIEW, INFINITE HORIZON
 C THE METHODOLOGY USED IN THE ALGORITHM IS AN HEURISTIC METHOD CALLED
 C "MODIFIED MULTIPLE PASS" SUGGESTED BY W.B.CROWSTON, M.H. WAGNER, AND
 C JACK WILLIAMS.

C
 C INPUT VARIABLES
 C N = NUMBER OF STAGES
 C S = SET UP COST
 C H = INVENTORY CARRYING COST.
 C DEM = DEMAND

C OUTPUTS
 C TCTAL COST
 C Q = LOT SIZE

C DIMENSION S(10),H(10),K(10),K0(10),IQ(10),Q(10)

1 READ(5,1)N
 1 FORMAT(I2)
 2 READ(5,2)(S(J),H(J),J=1,N)
 2 FORMAT(F9.2,F6.3)
 3 READ(5,3)DEM
 3 FORMAT(F7.6)
 K(N)=1
 Q(N)=SQRT(2*DEM*S(N)/H(N))
 IQ(N)=RED(Q(N))
 NN=N-1
 DO 4 I=1,NN
 Q(I)=SQRT(2*DEM*S(I)/H(I))
 IQ(I)=RED(Q(I))
 RP=IQ(I)/IQ(N)
 K(I)=RED(RP)
 IF(K(I).NE.0)GO TO 4
 K(I)=1
 4 CONTINUE
 CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
 DO 5 I=1,N
 K0(I)=K(I)
 5 CONTINUE
 TNJMIN=TNJ
 IQNJM=IQNJ
 L=1
 15 J=N
 12 J=J-1
 7 K(J)=K(J)+1
 CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
 IF(TNJ.GE.TNJMIN)GO TO 6
 TNJMIN=TNJ
 IQNJM=IQNJ
 K0(J)=K(J)
 GO TO 7
 6 K(J)=K(J)-1
 IF(K(J).EQ.1)GO TO 8
 10 K(J)=K(J)-1
 IF(K(J).EQ.0)GO TO 9
 CALL QNTN(S,K,H,DEM,IQNJ,N,TNJ)
 IF(TNJ.GE.TNJMIN)GO TO 9

```

      TNJMIN=TNJ
      IQNJM=IQNJ
      KC(J)=K(J)
      GO TO 10
9     K(J)=K(J)+1
8     CONTINUE
      IF(J.EQ.1)GO TO 11
      GO TO 12
11    IF(L.EQ.1)GO TO 13
      IF(TTNJMJ.EQ.TNJMIN)GO TO 14
13    TTNJMJ=TNJMIN
      L=L+1
      GO TO 15
14    CONTINUE
      DO 16 I=1,N
      IQ(I)=KC(I)*IQNJM
16    CONTINUE
      WRITE(6,18)TNJMIN
18    FORMAT(1H1,20X,"TOTAL COST = ",F12.2)
      DO 19 I=1,N
      WRITE(6,17)I,IQ(I)
17    FORMAT(1H1,45X,"Q(",I3,") = ".I5)
19    CONTINUE
      STOP
      END

```

C

C FUNCTION "RED"

C

```

FUNCTION RED(X)
  IJ=X
  RESI=X-IJ
  P5=.5
  IF(RESI.GE.P5)GO TO 97
  RED=IJ
  GO TO 98
97  RED=IJ+1
98  CONTINUE
  RETURN
  END

```

C

C SUBROUTINE "QNTN"

C

```

SUBROUTINE QNTN(S,K,H,DEM,IQNJ,N,TNJ)
  DIMENSION S(10),K(10),H(10)
  T01=0
  T02=0
  DO 20 I=1,N
  T01=T01+S(I)/K(I)
  T02=T02+H(I)*K(I)
20  CONTINUE
  QNJ=SQRT(2*DEM*T01/T02)
  IQNJ=RED(QNJ)
  TNJ=0
  DO 21 I=1,N
  TNJ=TNJ+DEM*S(I)/(K(I)*IQNJ)+K(I)*IQNJ*H(I)/2
21  CONTINUE
  RETURN
  END

```

```
PROGRAM TH5 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
```

```
C
```

```
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF AN ASSEMBLY STRUCTURE,  
C SINGLE ITEM, STATIONARY DE AND. CONTINUOUS REVIEW, INFINITE HORIZON.
```

```
C THE METHODOLOGY USED IN THE ALGORITHM IS A BRANCH AND BOUND METHOD  
C SUGGESTED BY LEROY B. SCHWAFX AND LINUS SCHRAGE.
```

```
C
```

```
INPUT VARIABLES
```

```
C N = NUMBER OF STAGES
```

```
C S = SET UP COST
```

```
C H = INVENTORY CARRYING COST
```

```
C DEM = DEMAND
```

```
OUTPUTS
```

```
C TOTAL COST
```

```
C Q = LOT SIZE
```

```
C
```

```
DIMENSION S(10),H(10),SS(10),HH(10),K(10),IQ(10),IIQ(10),I4(10)
```

```
DIMENSION JIIQ(10),IG(10000),KG(10000),I2(10)
```

```
DIMENSION IJQR(10000)
```

```
READ(5,1)N
```

```
1 FORMAT(I2)
```

```
READ(5,2)(S(J),H(J),J=1..N)
```

```
2 FORMAT(F9.2,F6.3)
```

```
READ(5,3)DEM
```

```
3 FORMAT(F7.0)
```

```
CZV=0
```

```
RIER=1
```

```
DO 4 I=1,N
```

```
SS(I)=S(I)
```

```
HH(I)=H(I)
```

```
K(I)=1
```

```
4 CONTINUE
```

```
7 Q=SQRT(2*SS(N)+DEM/HH(N))
```

```
IQ(N)=RED(Q)
```

```
ISMAIL=IQ(N)
```

```
II=N
```

```
N1=N-1
```

```
DO 5 I=1,N1
```

```
Q=SQRT(2*SS(I)+DEM/HH(I))
```

```
IQ(I)=RED(Q)
```

```
IF(K(I).EQ.1)GO TO 5
```

```
IF(IQ(I).GE.ISMAIL)GO TO 5
```

```
ISMAIL=IQ(I)
```

```
II=I
```

```
5 CONTINUE
```

```
IF(ISMAIL.GE.IQ(N))GO TO 6
```

```
SS(N)=SS(N)+SS(II)
```

```
HH(N)=HH(N)+HH(II)
```

```
K(II)=1
```

```
GO TO 7
```

```
6 J=0
```

```
DO 8 I=1,N
```

```
IF(K(I).EQ.1)GO TO 8
```

```
IIQ(I)=IQ(I)
```

```
GO TO 8
```

```
8 CONTINUE
```

```
DO 83 LI=1,N
```

```
WRITE(6,32)LI,IIQ(LI)
```

```
82 FORMAT(1H0,40X,"0(",I3,"") = ",I5)
```

```
83 CONTINUE
```

```
M=0
```

```
DO 10 I=1,N1
```

```
IF(K(I).EQ.1)GO TO 10
```

```

CAMIS1=IIQ(I)
CAMIS2=IIQ(N)
RR=CAMIS1/CAMIS2
WRITE(6,85)RR
85  FORMAT(1HL,30X,"RR = ",F9.3)
IR1=RED(RR)
WRITE(6,94)IR1
94  FORMAT(1HL,30X,"IR1 = ",I7)
RR2=RR-IR1
RR3=ABS(RR2)
WRITE(6,96)RR3
96  FORMAT(1HL,30X,"RR3 = ",F9.4)
PP5=.05
IF(RR3.LE.PP5)GO TO 432
M=M+1
I4(M)=I
GO TO 10
432 K(I)=IR1
10  CONTINUE
WRITE(6,95)M
95  FORMAT(1HL,30X,"M = ",I2)
JS=0
J=0
K1MIN=0
K(N)=1
IF(M.GT.0)GO TO 11
WRITE(6,12)
12  FORMAT(1HL,25X,"EVERYTHING GREAT")
CALL TN(S,DEM,N,H,K,TOT,IJQ)
WRITE(6,13)TOT
13  FORMAT(1HL,20X,"TOTAL COST = ",F12.2)
DO 14 I=1,N
WRITE(6,15)I,IIQ(I)
15  FORMAT(1HL,40X,"O(",I3,") = ",I5)
14  CONTINUE
GO TO 35
11  I=1
30  CONTINUE
DO 16 JV=1,M
IF(I.EQ.JV)GO TO 16
IP=I4(JV)
K(IR)=1
16  CONTINUE
JANA=I4(I)
CAMIS1=IIQ(JANA)
CAMIS2=IIQ(N)
RSE=CAMIS1/CAMIS2
K(JANA)=RED(RSE)
420 CONTINUE
WRITE(6,71)(K(JJ),JJ=1,N)
71  FORMAT(1HL,5X,1' (74))
CALL TN(S,DEM,N,H,K,TOT,IJQ)
J=J+1
JS=JS+1
TG(J)=TOT
KG(J)=K1MIN*10+K(JANA)
IJQF(J)=IJQ
K(JANA)=K(JANA)+1
IF(JS.LE.5)GO TO 420
JS=0
K(JANA)=K(JANA)-6
19  IF(K(JANA).LE.0)GO TO 18
J=J+1
JS=JS+1

```

```

CALL TN(S,DEM,N,H,K,TOT,IJQ)
TG(J)=TOT
KG(J)=K1MIN*1(+K(JANA)
IJQ(J)=IJQ
K(JANA)=K(JANA)-1
IF(JS.LE.5)GO TO 19
JS=0
18 KGMIN=KG(J)
TGMIN=TG(J)
DO 20 KK=1,J
IF(TG(KK).GE.TGMIN)GO TO 20
TGMIN=TG(KK)
KK=KK
KGMIN=KG(KK)
IJQMIN=IJQ(KK)
20 CONTINUE
K1MIN=KGMIN
DO 21 LL=1,J
IF(LL-KKK)22,21,23
22 LLL=LL
GO TO 24
23 LLL=LL-1
24 TG(LLL)=TG(LL)
KG(LLL)=KG(LL)
21 CONTINUE
J=J-1
DO 25 I9=1,M
IF(KGMIN.GE.1)GO TO 26
GO TO 27
26 KLM=KGMIN-KGMIN/10*10
I2(I9)=KLM
KGMIN=KGMIN/10
25 CONTINUE
CZV=1
27 CONTINUE
I91=I9-1
DO 29 J3=1,I91
J94=I9-J3
J95=I2(J94)
J96=I4(J95)
K(J96)=J95
29 CONTINUE
IF(CZV.EQ.1)GO TO 31
I=I9
GO TO 30
31 WRITE(6,32)TGMIN
32 FORMAT(1H1,20X,"TOTAL COST = ",F12.2)
DO 48 I=1,N
JIIQ(I)=K(I)*IJQMIN
48 CONTINUE
DO 33 I=1,N
WRITE(6,34)I,JIIQ(I)
34 FORMAT(1H1,4(X,"0(",13,"") = ",15)
33 CONTINUE
35 CONTINUE
STOP
END

```

C
C
C

```

SUBROUTINE "TN"
SUBROUTINE TN(S,DEM,N,H,K,TOT,IJQ)
DIMENSION S(10),H(10),K(10)
TOT1=0
TOT2=0

```

```
DO 99 I=1,N
TOT1=TOT1+S(I)*DEM/K(I)
TOT2=TOT2+H(I)*K(I)/2
Q=SQRT(TOT1/TOT2)
IJQ=FED(Q)
99 CONTINUE
TOT=J
DO 100 I=1,N
TOT=TOT+S(I)*DEM/(K(I)*IJQ)+H(I)*IJQ*K(I)/2
100 CONTINUE
RETURN
END
```

C

C FUNCTION "RED"

C

```
FUNCTION FED(X)
IJ=X
RESI=X-IJ
P5=.5
IF(RESI.GE.P5)GO TO 97
RED=IJ
GO TO 95
97 RED=IJ+1
98 CONTINUE
RETURN
END
```

T H E S I S 6

PROGRAM THE6 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

C
 C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF AN ASSEMBLY STRUCTURE,
 C SINGLE ITEM, STATIONARY DEMAND, CONTINUOUS REVIEW, INFINITE HORIZON.
 C THE METHODOLOGY USED IN THE ALGORITHM IS A GROUP OF MYOPIC POLICIES
 C SUGGESTED BY LERDY B. SCHWARZ AND LINUS SCHRAGE.

C
 C INPUT VARIABLES
 C N = NUMBER OF STAGES
 C S = SET UP COST
 C H = INVENTORY CARRYING COST
 C DEM = DEMAND

C OUTPUTS
 C TOTAL COST
 C Q = LOT SIZE
 C

DIMENSION S(10),H(10),RM(10),K(10),IQ(10)

READ(5,1)N

1 FORMAT(I2)

READ(5,2)(S(J),H(J),J=1,N)

2 FORMAT(F9.2,F6.3)

READ(5,3)DEM

3 FORMAT(F7.0)

T01=0

T02=0

N1=N-1

DO 4 I=1,N1

RM(I)=S(I)*H(N)/(S(N)+H(I))

K(I)=0

5 K(I)=K(I)+1

KK=K(I)*(K(I)+1)

IF(KK.LT.RM(I))GO TO 5

T01=T01+S(I)/K(I)

T02=T02+K(I)+H(I)

4 CONTINUE

K(N)=1

Q1=SQRT(2*DEM*T01/T02)

IQ(N)=RED(Q1)

CT=0

DO 6 I=1,N

IQ(I)=K(I)+IQ(N)

CT=CT+S(I)*DEM/IQ(I)+H(I)+IQ(I)/2

6 CONTINUE

WRITE(6,7)CT

7 FORMAT(1H1,25X,"TOTAL COST = ",F12.2)

DO 8 I=1,N

WRITE(6,9)I,IQ(I)

9 FORMAT(1H5,40X,"Q(",I3,") = ",I5)

8 CONTINUE

STOP

END

C

C FUNCTION "RED"

C

FUNCTION RED(X)

IJ=X

```

RESI=X-IJ
P5=.5
IF (RESI.GE.P5)GO TO 97
RED=IJ
GO TO 98
97 RED=IJ+1
98 CONTINUE
RETURN
END

```

T H E S I S 8

PROGRAM THE8 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

```

C
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF SEVERAL RETAILERS
C AND JUST ONE WAREHOUSE, SINGLE ITEM, STATIONARY DEMANDS, CONTINUOUS
C REVIEW, INFINITE HORIZON, THE METHODOLOGY USED IN THE ALGORITHM IS
C SUGGESTED BY LEROY B. SCHWARZ
C

```

```

C INPUT VARIABLES
C N = NUMBER OF STAGES
C S = SET UP COST
C H = INVENTORY CARRYING COST
C D = DEMAND

```

```

C OUTPUTS
C TOTAL COST
C LOT SIZE AT STAGE 1 (WAREHOUSE)
C K = INTEGER MULTIPLIERS
C

```

```

DIMENSION S(10),H(10),D(10),T1(10),II(10),CRJMIN(10)
DIMENSION KK(10)
READ(5,1)N
1 FORMAT(I2)
READ(5,2)(S(J),H(J),J=1,N)
2 FORMAT(F9.2,F6.3)
READ(5,3)(D(J),J=2,N)
3 FORMAT(F7.0)
D(1)=.
DO 4 I=2,N
D(1)=D(1)+D(I)
4 CONTINUE
TW=SQRT(2*S(1)/(D(1)*H(1)))
J=0
DO 5 I=2,N
T1(I)=SQRT(2*S(I)/(D(I)*H(I)))
IF(T1(I).LE.TW)GO TO 5
J=J+1
II(J)=I
5 CONTINUE
IF(J.LE.0)GO TO 33
TS=TW
GO TO 44
33 TOT1=.
TOT2=.
DO 6 I=1,J
L=II(I)
TOT1=TOT1+S(L)
TOT2=TOT2+H(L)*D(L)
6 CONTINUE
TS=SQRT(2*(S(1)+TOT1)/(H(1)*D(1)+TOT2))
44 CT=0

```

```

DO 7 I=2,N
K=0
10 K=K+1
CRJ=K*S(I)/TS+H(I)*D(I)*TS/(2*K)
IF(K.EQ.1)GO TO 8
IF(CRJ.GE.CRJMIN(I))GO TO 9
8 CRJMIN(I)=CRJ
KK(I)=K
GO TO 10
9 CT=CT+CRJMIN(I)
7 CONTINUE
CT=CT+S(1)/TS+H(1)*D(1)*TS/2
IWLO=TS*D(1)
WRITE(6,12)CT
12 FORMAT(1H1,25X,"TOTAL COST = ",F12.2)
WRITE(6,80)IWLO
80 FORMAT(1H0,25X,"LOT SIZE AT WAREHOUSE = ",I6)
DO 13 I=2,N
WRITE(6,14)I,KK(I)
14 FORMAT(1H1,25X,"K( 1 ",I2," ) = ",I2)
13 CONTINUE
STOP
END

```

T H E S I S 9

PROGRAM THE9 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

```

C
C THIS PROGRAM OBTAINS OPTIMUM LOT SIZES FOR THE CASE OF SEVERAL
C RETAILERS AND JUST ONE WAREHOUSE, SINGLE ITEM, STATIONARY DEMANDS. CON
C TINUJUS REVIEW, INFINITE HORIZON, THE METHODOLOGY USED IN THE ALGORITHM
C IS THE BRANCH AND BOUND ALGORITHM SUGGESTED BY STEPHEN C. GRAVES
C AND LEROY B. SCHWARZ.

```

```

C
C INPUT VARIABLES
C N = NUMBER OF STAGES
C S = SET UP COST
C H = INVENTORY CARRYING COST
C D = DEMAND
C
C OUTPUTS
C LOT SIZE AT STAGE 1 (WAREHOUSE)
C TOTAL COST
C K = INTEGER MULTIPLIERS

```

```

C
DIMENSION S(10),H(10),D(10),JK(10),BMIN(10000),KG(10000),I2(10)
READ(5,1)N
1 FORMAT(I2)
READ(5,2)(S(J),H(J),J=1,N)
2 FORMAT(F9.2,F6.3)
READ(5,3)(D(J),J=2,N)
3 FORMAT(F7.0)
D(1)=I
DO 4 I=2,N
JK(I)=1
D(1)=C(1)+D(I)
4 CONTINUE
II=2
J=1
JS=0
KIMIN=0
39 K1=JK(II)

```

```

6  CONTINUE
   CALL DLSH(S,H,II,JK,BL,D,N)
   JS=JS+1
   BMIN(J)=BL
   KG(J)=K1MIN*10+K1
   K1=K1+1
   IF(JS.EQ.9)GO TO 5
   J=J+1
   JK(II)=K1
   GO TO 6

5  JS=0
   BMINMI=BMIN(J)
   KJMIN=KG(J)
   DO 7 KK=1,J
   IF(BMIN(KK).GE.BMINMI)GO TO 7
   BMINMI=BMIN(KK)
   KKK=KK
   KGMIN=KG(KK)

7  CONTINUE
   K1MIN=KGMIN
   DO 8 LL=1,J
   IF(LL-KKK)9,9,10

10 LLL=LL-1
   GO TO 11

9  LLL=LL
11 BMIN(LLL)=BMIN(LL)
   KG(LLL)=KG(LL)

8  CONTINUE
   J=J-1
   DO 12 I9=2,N
   IF(KGMIN.LT.1)GO TO 13
   RLC1=KGMIN
   RLC=RLC1/10.
   ILCV=RLC
   RRLC=RLC-ILCV
   RLC2=5*RLC*10.
   KLM=KGMIN-ILCV/10*10
   I2(I9)=KLM
   KGMIN=RLC

12 CONTINUE
   GO TO 310

13 CONTINUE
   DO 14 I9=2,N
   JK(I9)=1

14 CONTINUE
   IZ=1
   I91=I9-1
   DO 15 J3=2,I91
   J94=I9-I2
   JK(J3)=I2(J94)
   IZ=IZ+1

15 CONTINUE
   II=I9
   GO TO 30

310 CONTINUE
   DO 16 J99=2,N
   JK(J99)=1

16 CONTINUE
   IZ=0
   DO 17 J3=2,N
   J94=N-IZ
   JK(J3)=I2(J94)
   IZ=IZ+1

17 CONTINUE

```

```

JK(1)=1
TOT1=0
TOT2=0
DO 18 I=1,N
TOT1=TOT1+JK(I)*S(I)
TOT2=TOT2+H(I)*D(I)/(JK(I)*D(I))
18 CONTINUE
QNS=SQRT(2*D(1)*TOT1/TOT2)
QNST=AIHT(QNS)
CT=TOT1*D(1)/QNST+TOT2*QNST/2
IQNST=QNST
WRITE(6,19)IQNST,CT
19 FORMAT(1H1,20X,"WAREHOUSE LOT SIZE = ",I5,10X,"TOTAL COST = ",F12.
-2)
DO 20 I=2,N
WRITE(6,21)I,JK(I)
21 FORMAT(1H1,25X,"K( 1 ",I2," ) = ",I2)
20 CONTINUE
STOP
END

```

```

C
C      SUBROUTINE "BLSH"
C

```

```

SUBROUTINE BLSH(S,H,II,JK,BL,D,N)
DIMENSION S(10),H(10),JK(10),D(10)
SH=S(1)
HH=H(1)
DO 80 I=2,II
SH=SH+JK(I)*S(I)
HH=HH+H(I)*D(I)/(JK(I)*D(I))
80 CONTINUE
BL=SQRT(2*SH*D(1)*HH)
N12=II+1
DO 81 I=N12,N
BL=BL+SQRT(2*S(I)*D(I)*H(I))
81 CONTINUE
RETURN
END

```

T H E S I S 10

```
PROGRAM THE10 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
```

```

C
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF SEVERAL RETAILERS AND
C JUST ONE WAREHOUSE, SINGLE ITEM, STATIONARY DEMANDS, CONTINUOUS
C REVIEW, INFINITE HORIZON. THE METHODOLOGY USED IN THE ALGORITHM IS
C A GROUP OF MYOPIC POLICIES SUGGESTED BY STEPHEN D. GRAVES AND
C LEROY B. SCHWARZ.
C

```

```

C      INPUT VARIABLES
C      N = NUMBER OF STAGES
C      S = SET UP COST
C      H = INVENTORY CARRYING COST
C      D = DEMAND

```

```

C      OUTPUTS
C      LOT SIZE AT STAGE 1 (WAREHOUSE)
C      TOTAL COST
C      K = INTEGER MULTIPLIERS
C

```

```

DIMENSION S(10),H(10),D(10),K(10)
READ(5,1)N

```

```

1  FORMAT(I2)
   READ(5,2) (S(J),H(J),J=1,N)
2  FORMAT(F9.2,F6.3)
   READ(5,3) (D(J),J=2,N)
3  FORMAT(F7.0)
   D(1)=0
   DO 4 I=2,N
     D(1)=D(1)+D(I)
4  CONTINUE
   DO 5 I=2,N
     RM=S(1)*(D(I)/D(1))*H(I)/(S(I)*H(1))
     K(I)=1
6  KK=K(I)*(K(I)+1)
   IF(KK.GE.FM)GO TO 5
   K(I)=K(I)+1
   GO TO 6
5  CONTINUE
   TOT1=0
   TOT2=0
   K(1)=1
   DO 7 I=1,N
     TOT1=TOT1+K(I)*S(I)
     TOT2=TOT2+I(I)*D(I)/(K(I)*D(1))
7  CONTINUE
   QNS=SQRT(2*D(1)*TOT1/TOT2)
   QNST=AINT(QNS)
   CT=TOT1*D(1)/QNST+TOT2*QNST/2
   IQNST=QNST
   WRITE(6,19)IQNST,CT
19  FORMAT(14I,20X,"WAREHOUSE LOT SIZE = ",I9,10X,"TOTAL COST = ",F12.
   -2)
   DO 20 I=2,N
     WRITE(6,21)I,K(I)
21  FORMAT(14I,25X,"K( 1 ",I2," ) = ",I2)
20  CONTINUE
   STOP
   END

```

T H E S I S 11

PROGRAM THE11 (INPUT,OUTPUT,TAPES=INPUT,TAPE 6=OUTPUT)

```

C
C THIS PROGRAM OBTAIN LOT SIZES FOR THE CASE OF SEVERAL RETAILERS AND
C JUST ONE WAREHOUSE, SINGLE ITEM, STATIONARY DEMANDS, CONTINUOUS
C REVIEW, INFINITE HORIZON, THE METHODOLOGY USED IN THE ALGORITHM IS
C A GROUP OF MODIFIED MYOPIC POLICIES SUGGESTED BY STEPHEN C. GRAVES
C AND LEROY J. SCHARZ

```

```

C
C INPUT VARIABLES
C N = NUMBER OF STAGES
C S = SET UP COST
C H = INVENTORY CARRYING COST
C D = DEMAND

```

```

C OUTPUTS
C LOT SIZE AT STAGE 1 (WAREHOUSE)
C TOTAL COST
C K = INTEGER MULTIPLIERS

```

```

C DIMENSION S(10),H(10),D(10),K(10)
   READ(5,1)N
1  FORMAT(I2)

```

```

      READ(5,2) (S(J),H(J),J=1,N)
2     FORMAT(F9.2,F6.3)
      READ(5,3) (D(J),J=2,N)
3     FOPMAT(F7.0)
      CONT1=1
      D(1)=C
      DO 4 I=2,N
      D(1)=D(1)+D(I)
4     CONTINUE
      DO 5 I=2,N
      RM=S(1)*(D(I)/D(1))*H(I)/(S(I)*H(I))
      K(I)=1
6     KK=K(I)*(K(I)+1)
      IF(KK.GE.PM)GO TO 5
      K(I)=K(I)+1
      GO TO 6
5     CONTINUE
11    TOT1=0
      TOT2=0
      K(1)=1
      DO 7 I=1,N
      TOT1=TOT1+K(I)*S(I)
      TOT2=TOT2+H(I)*D(I)/(K(1)*D(1))
7     CONTINUE
      QNS=SQRT(2*D(1)*TOT1/TOT2)
      QNST=AIN(T,QNS)
      CT=TOT1*D(1)/QNST+TOT2*QNST/2
      IF(CT1.EQ.1)GO TO 9
      IF(CT.LT.CTMIN)GO TO 8
      GO TO 14
8     CTMIN=CT
      CONT1=CONT1+1
      DO 9 I=2,N
      RM=(H(I)*D(I)*QNST**2)/(2*S(I)*D(I)**2)
      K(I)=1
10    KK=K(I)*(K(I)+1)
      IF(KK.GE.PM)GO TO 9
      K(I)=K(I)+1
      GO TO 10
9     CONTINUE
      GO TO 11
14    CONTINUE
      IQNST=QNST
      WRITE(6,19)IQNST,CT
19    FORMAT(1H1,20X,"WAREHOUSE LOT SIZE = ",I9,10X,"TOTAL COST = ",F12.
-2)
      DO 20 I=2,N
      WRITE(6,21)I,K(I)
21    FORMAT(1H1,25X,"K( 1 ",I2," ) = ",I2)
20    CONTINUE
      STOP
      END

```

T H E S I S 12

PROGRAM THE12 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

C

C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF AN ASSEMBLY STRUCTURE
C SINGLE ITEM, STATIONARY DEMAND, CONTINUOUS REVIEW, INFINITE HORIZON.
C THE METHODOLOGY USED IN THE ALGORITHM IS A DYNAMIC PROGRAMMING
C APPROACH SUGGESTED BY RICARDO R. LOFFZ.

```

C
C      INPUT VARIABLES
C          N = NUMBER OF STAGES
C          S = SET UP COST
C          H = INVENTORY CARRYING COST
C          DEM = DEMAND
C      OUTPUTS
C          TOTAL COST
C          Q = LOT SIZE
C      DIMENSION S(10),H(10),K(10,1000),T(10,1000),IQ(10)
C      READ(5,1)N
1      FORMAT(I2)
C      READ(5,2)(S(J),H(J),J=1,N)
2      FORMAT(F9.2,F6.3)
C      READ(5,3)DEM
3      FORMAT(F7.0)
C      DO 4 I=1,N
C      QIU=SQRT(2*DEM*S(I)/H(I))
C      IF(I.EQ.1)GO TO 5
C      IF(QUU.GT.QIU)GO TO 5
C      IF(QUU.LT.QIU)GO TO 6
C      GO TO 4
5      QLU=QIU
C      IF(I.NE.1)GO TO 4
6      QUU=QIU
4      CONTINUE
C      IQLU=RED(QIU)
C      IQUU=RED(QUU)
47      WRITE(6,47)IQLU,IQUU
C      FORMAT(1H1,20X,16,I6)
C      IDIF=IQUU-IQLU
C      IQLU1=IQLU-.2*IDIF
C      IQUU1=IQUU+.1*IDIF
C      IQLU=MAX0(1,IQLU1)
C      J=L
10     J=J+1
C      JA=J-1
C      DO 7 I=IQLU,IQUU
C      RLC=SQRT(2*S(J)*DEM/(I**2*H(J)))
C      KRED=RED(RLC)
C      K(J,I)=MAX0(1,KRED)
C      IF(J.NE.1)GO TO 8
C      T(J,I)=S(J)*DEM/(K(J,I)*I)+K(J,I)*I*H(J)/2
C      GO TO 7
8      IF(J.EQ.N)GO TO 9
C      T(J,I)=S(J)*DEM/(K(J,I)*I)+K(J,I)*I*H(J)/2+T(JA,I)
C      GO TO 7
9      T(J,I)=S(J)*DEM/I+I*H(J)/2+T(JA,I)
7      CONTINUE
C      IF(J.NE.N)GO TO 10
C      TOP=T(N,IQUU)
C      DO 11 I=IQLU,IQUU
C      IF(TOP.LT.T(N,I))GO TO 11
C      TOP=T(N,I)
C      IQOP=I
11     CONTINUE
C      NN=N-1
C      DO 12 JJ=1,NN
C      IQ(JJ)=IQOP*K(JJ,IQOP)
12     CONTINUE
C      IQ(N)=IQOP
C      WRITE(6,32)TOP
32     FORMAT(1H1,20X,"TOTAL COST = ",F12.2)
C      DO 33 I=1,N

```

```

WRITE(6,34)I,IQ(I)
34 FORMAT(1H0,40X,"Q(",I3,") = ",I5)
33 CONTINUE
STOP
END

```

```

C
C   FUNCTION "RED"
C

```

```

FUNCTION RED(X)
IJ=X
RESI=X-IJ
P5=.5
IF(RESI.GE.P5)GO TO 97
RED=IJ
GO TO 95
97 RED=IJ+1
98 CONTINUE
RETURN
END

```

T H E S I S 13

PROGRAM THE13 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

```

C
C THIS PROGRAM MAKES A COMPARISON BETWEEN MULTIPLE AND DIVISOR
C POLICIES FOR THE CASE OF TWO LEVEL, TWO STAGE SYSTEM, SIMULTANEOUS
C AND NONSIMULTANEOUS PRODUCTION AND CONSUMPTION.
C BASICALLY THE DATA HAS BEEN TAKEN FROM AN SPECIFIC PROBLEM ASSIGNED
C AS A HOMEWORK IN SPRING-77, ISYE 6308 BY PROF. LYNNWOOD A. JOHNSON.
C THE RATIOS TESTED IN THIS PROGRAM ARE A2/A1, H2/H1, AND P2/D
C

```

```

C   INPUT VARIABLES
C   RDP = UTILIZATION RATIO
C   RH = INVENTORY RATIO
C   RA = SET UP COST RATIO
C

```

```

C   OUTPUTS
C   THE SPECIFIC VALUES OF THE RATIOS
C   TOTAL COST, USING MULTIPLE POLICY
C   TOTAL COST, USING DIVISOR POLICY
C   COST DIFFERENCES
C

```

```

C
DIMENSION RDP(7),RH(7),RA(7)
A1=28.
H1=2
D=10000
TAO=.008
READ(5,22)(RDP(I),RH(I),RA(I),I=1,7)
22 FORMAT(F3.2,F3.0,F5.2)
DO 25 I=1,7
WRITE(6,26)RDP(I)
26 FORMAT(1H0,15X,"UTILIZATION RATIO D/P2 = ",F3.2)
P2=D/RDP(I)
DO 27 J=1,7
WRITE(6,28)RH(J)
28 FORMAT(1H0,15X,"INVENTORY RATIO H2/H1 = ",F3.0)
H2=H1*RH(J)
DO 29 K=1,7
A2=A1/RA(K)
CALL MULT(A1,A2,H1,H2,D,P2,TAO,TM)
CALL DIVI(A1,A2,H1,H2,D,P2,TAO,TD)
RAT=RA(K)

```

```

DIFE=TM-TD
WRITE(6,1)RAT, TM, TD, DIFE
1  FORMAT(1H(,15X,"A1/A2 = ",F7.2,5X,"TOT MULT = ",F10.2,5X,"TOT DIV
   -= ",F10.2,5X,"DIFFERENCE = ",F10.2)
29  CONTINUE
27  CONTINUE
25  CONTINUE
    STOP
    END

C
C      SUBROUTINE "MULT" (SIMULTANEOUS PROD. & CONS.)
C
SUBROUTINE MULT(A1,A2,H1,H2,D,P2,TAG, TM)
R1=A1*(H2-H1)*(1-D/P2)/(A2*H1)
12  N=0
13  N=N+1
    NN=N*(N-1)
    NNN=N*(N+1)
    IF(NN.GT.R1)GO TO 12
    IF(NNN.LT.R1)GO TO 13
    NOP=N
    T1=2*(A1/NOP+A2)*D
    T2=H1*(NOP-1+D/P2)+H2*(1-D/P2)
    TM=SQRT(T1*T2)+H2*TAG*D
    RETURN
    END

C
C      SUBROUTINE "DIVI" (SIMULTANEOUS PROD. & CONS.)
C
SUBROUTINE DIVI(A1,A2,H1,H2,D,P2,TAG, TD)
R2=A2*H1*(D/P2)/(A1+H2*(1-D/P2))
22  N=0
23  N=N+1
    NN=N*(N-1)
    NNN=N*(N+1)
    IF(NN.GT.P2)GO TO 22
    IF(NNN.LT.R2)GO TO 23
    NOP=N
    T1=2*D*(NOP*A1+A2)
    T2=H1*D/(NOP*P2)+H2*(1-D/P2)
    TD=SQRT(T1*T2)+H2*TAG*D
    RETURN
    END

C
C      SUBROUTINE "MULT" (NONSIMULTANEOUS PROD. & CONS.)
C
SUBROUTINE MULT(A1,A2,H1,H2,D,P2,TAG, TM)
R1=A1*(H2*(1+D/P2)-H1*(1-D/P2))/(A2*H1)
12  N=0
13  N=N+1
    NN=N*(N-1)
    NNN=N*(N+1)
    IF(NN.GT.R1)GO TO 12
    IF(NNN.LT.R1)GO TO 13
    NOP=N
    T1=2*(A1/NOP+A2)*D
    T2=H1*(NOP-1+D/P2)+H2*(1+D/P2)
    TM=SQRT(T1*T2)+H2*TAG*D
    RETURN
    END

C
C      SUBROUTINE "DIVI" (NONSIMULTANEOUS PROD. & CONS.)
C
SUBROUTINE DIVI(A1,A2,H1,H2,D,P2,TAG, TJ)

```

```
R2=A2*H1*(D/P2)/(A1*H2*(1+D/P2))
22 N=C
23 N=N+1
   NN=N*(N-1)
   NNN=N*(N+1)
   IF(NN.GT.F2)GO TO 22
   IF(NNN.LT.R2)GO TO 23
   NCP=N
   T1=2*D*(NCP+A1+A2)
   T2=H1*D/(NCP*F2)+H2*(1+D/F2)
   TO=SQRT(T1*T2)+H2*TAO*C
   RETURN
   END
```

T H E S I S 14

PROGRAM THE14 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)

C
 C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF SEVERAL RETAILERS
 C AND JUST ONE WAREHOUSE. SINGLE ITEM, STATIONARY DEMANDS, CONTINUOUS
 C REVIEW, INFINITE HORIZON, THE METHODOLOGY USED IN THE ALGORITHM IS
 C A DYNAMIC PROGRAMMING APPROACH SUGGESTED BY RICARDO LOPEZ.

C
 C INPUT VARIABLES
 C N = NUMBER OF STAGES
 C S = SET UP COST
 C H = INVENTORY CARRYING COST
 C D = DEMAND
 C OUTPUTS
 C TOTAL COST
 C INTEGER MULTIPLIERS
 C LOT SIZES
 C

```

DIMENSION S(10),H(10),D(10)
DIMENSION K(400),T(2,400),KK(400),KDP(10),I0(10)
READ(5,1)N
1  FORMAT(I2)
WRITE(6,35)N
35  FORMAT(1H1,10X,"N",I3)
READ(5,2)(S(J),H(J),J=1,N)
2  FORMAT(F9.2,F6.3)
WRITE(6,36)(S(J),H(J),J=1,N)
36  FORMAT(1H1,10X,"S AND H",F9.2,F6.3)
READ(5,3)(D(J),J=2,N)
3  FORMAT(F7.0)
WRITE(6,37)(D(J),J=2,N)
37  FORMAT(1H1,10X,"D",F7.0)
K1=1
D(1)=0
DO 4 I=2,N
D(1)=D(1)+D(I)
4  CONTINUE
Q1=SQRT(2*S(1)*D(1)/H(1))
IQ1=RED(Q1)
IQ1LB=IQ1-.3*IQ1
IQ1UB=IQ1+.3*IQ1
WRITE(6,75)IQ1LB,IQ1UB
75  FORMAT(1H1,10X,"IQ1LB",I6,10X,"IQ1UB",I6)
DO 448 I=IQ1LB,IQ1UB
KK(I)=0
448 T(2,I)=0
3  K1=K1+1
DO 432 I=IQ1LB,IQ1UB
432 T(1,I)=T(2,I)
DO 5 I=IQ1LB,IQ1UB
IF(K1.GT.N)GO TO 6
RLC=SQRT(I**2*D(K1)+H(K1)/(2*D(1)**2*S(K1)))
KRLC=RED(RLC)
K(I)=MAX(1,KRLC,I,0)
IF(K1.NE.2)GO TO 7
T(2,I)=S(K1)+K(I)*D(1)/I+I*D(K1)+H(K1)/(2*K(I)*D(1))
KK(I)=KK(I)+I+K(I)
GO TO 5

```

```

7  T(2,I)=S(K1)*K(I)*D(1)/I+I*D(K1)*H(K1)/(2*K(I)*D(1))+T(1,I)
   KK(I)=KK(I)*1(+K(I))
   GO TO 5
6  T(1,I)=S(1)*D(1)/I+I*H(1)/2+T(1,I)
5  CONTINUE
   IF(K1.LE.N)GO TO 8
   TOP=T(1,IQ10B)
   DO 9 I=IQ1LB,IQ1UB
   IF(TOP.LT.T(1,I))GO TO 9
   TOP=T(1,I)
   IQ10P=I
9  CONTINUE
   WRITE(6,4)KK(IQ10P)
40  FORMAT(1H0,10X,"KK(IQ10P) = ",I12)
   NN=N
   DO 11 I=2,N
   KLM=KK(IQ10P)-KK(IQ10P)/1(+1(
   KOP(NN)=KLM
   WRITE(6,39)KOP(NN)
39  FORMAT(1H0,10X,"KOPTIMA",I6)
   KK(IQ10P)=KK(IQ10P)/10
110  NN=NN-1
   DO 10 I=2,N
10  IQ(I)=IQ10P*D(I)/(KOP(I)*D(1))
   IQ(1)=IQ10P
   WRITE(6,11)TOP
11  FORMAT(1H1,25X,"TOTAL COST = ",F12.2)
   KOP(1)=1
   DO 12 I=1,N
   WRITE(6,13)I,KOP(I),I,IQ(I)
13  FORMAT(1H1,25X,"K( 1,"",I3," ) = ",I3,5X,"Q( "",I2," ) = ",I5)
12  CONTINUE
   STOP
   END

```

```

C
C   FUNCTION "RED"
C

```

```

FUNCTION RED(X)
IJ=X
RESI=X-IJ
P5=.5
IF(RESI.GE.P5)GO TO 97
RED=IJ
GO TO 98
97  RED=IJ+1
98  CONTINUE
RETURN
END

```

T H E S I S 15

```
PROGRAM THE15 (INPUT,OUTPUT,TAPES=INPUT,TAPE6=OUTPUT,DEBUG=OUTPUT)
```

```

C
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF ACYCLIC STRUCTURE.
C SINGLE ITEM, STATIONARY DEMAND, CONTINUOUS REVIEW, INFINITE HORIZON.
C THE METHODOLOGY USED IN THE ALGORITHM IS PROPOSED BY RICARDO P.
C LOPEZ.

```

```

C
C   INPUT VARIABLES
C       M = STAGES AT LEVEL-2
C       N = NUMBER OF TOTAL STAGES

```

```

C          L = INCREMENTS (DELTA)
C          K = RELATIONSHIPS AMONG STAGES
C          S = SET UP COST
C          H = INVENTORY CARRYING COST
C          D = DEMAND
C          OUTPUTS
C          TOTAL COST
C          K = INTEGER MULTIPLIERS
C          OPTIMAL CYCLE TIME
          DIMENSION K(2,5),S(5),H(5),D(5),QC(5),KK(5)
          DIMENSION KKMI(5)
          READ(5,1)M,N,L
1         FORMAT(I2,I2,I5)
          MM=M+1
          MMM=N-M
          DO 2 I=1,M
          READ(5,3)(K(I,J),J=MM,N)
3         FORMAT(3(I1))
2         CONTINUE
          READ(5,4)(S(J),H(J),J=1,N)
4         FORMAT(F9.2,F6.3)
          READ(5,5)(D(J),J=MM,N)
5         FORMAT(F7.0)
          DO 6 I=1,M
          D(I)=1
          DO 7 J=MM,N
7         D(I)=D(I)+D(J)*K(I,J)
6         CONTINUE
          DO 8 I=1,N
          QC(I)=SQRT(2*D(I)*S(I)/H(I))
          QC(I)=RED(QC(I))
          IF(I.NE.1)GO TO 9
          DQCMIN=QC(I)/D(I)
          DQCMAX=QC(I)/D(I)
          GO TO 8
9         ADQC=QC(I)/D(I)
          IF(ADQC.GE.DQCMIN)GO TO 10
          DQCMIN=ADQC
          GO TO 8
10        IF(ADQC.LE.DQCMAX)GO TO 8
          DQCMAX=ADQC
8         CONTINUE
          DELTA=(DQCMAX-DQCMIN)/L
          WRITE(6,35)DQCMIN,DQCMAX,DELTA
35        FORMAT(1H,5X,"DQCMIN",F17.8,5X,"DQCMAX",F17.8,5X,"DELTA",F17.16)
          CLAVE=0
          ADQC=DQCMIN
24        COTD=J
          WRITE(6,36)ADQC
36        FORMAT(1H,5X,"ADQC",F17.8)
          DO 11 I=1,M
          CLAVE=1
          KK(I)=1
13        CT=S(I)/(KK(I)*ADQC)+ADQC*KK(I)*D(I)+H(I)/2
          WRITE(6,38)CT
38        FORMAT(1H,5X,"CT",F17.8)
          IF(CLAVE.NE.0)GO TO 12
          CLAVE=1
14        CTMIN=CT
          WRITE(6,37)I,KK(I)
37        FORMAT(1H,5X,"I",I5,"KK",I6)
          WRITE(6,39)CTMIN
39        FORMAT(1H,5X,"CTMIN",F17.8)
          KK(I)=KK(I)+1

```

```

      GO TO 13
12  IF(CTMIN.GE.CT)GO TO 14
      KK(I)=KK(I)-1
11  COTO=COTO+CTMIN
      DO 15 I=MM,N
      CLAVE=0
      KK(I)=1
17  CT=S(I)/(ADQC/KK(I))+(ADQC/KK(I))*D(I)*4(I)/2
      IF(CLAVE.NE.)GO TO 16
      CLAVE=1
18  CTMIN=CT
      WRITE(6,37)I, KK(I)
      KK(I)=KK(I)+1
      GO TO 17
16  IF(CTMIN.GE.CT)GO TO 18
      KK(I)=KK(I)-1
15  COTO=COTO+CTMIN
      IF(CLAVE1.NE.)GO TO 19
22  COTOMI=COTO
      DO 20 I=1,N
20  KKMI(I)=KK(I)
      ADQCOP=ADQC
      GO TO 21
19  IF(COTOMI.GE.COTO)GO TO 22
21  CLAVE1=1
      IF(ADQC.GE.DQCMA)GO TO 23
      ADQC=ADQC+DELTA
      GO TO 24
23  WRITE(6,25)COTOMI
25  FORMAT(1H1,20X,"TOTAL CCST = ",F12.2)
      DO 26 I=1,N
      WRITE(6,27)I, KKMI(I)
27  FORMAT(1H1,20X,"K( ",I2," ) = ",I4)
26  CONTINUE
      WRITE(6,28)ADQCOP
28  FORMAT(1H1,20X,"OPTIMAL CYCLE TIME = ",F13.6)
      STOP
      END

```

C
C
C

FUNCTION "RED"

```

FUNCTION RED(X)
  IJ=X
  RESI=X-IJ
  P5=.5
  IF(RESI.GE.P5)GO TO 97
  RED=IJ
  GO TO 98
97  RED=IJ+1
98  CONTINUE
  RETURN
  END

```

T H E S I S 16

PROGRAM THE16 (INPUT,OUTPUT,TAP15=INPUT,TAP16=OUTPUT)

C

C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF A SERIAL STRUCTURE,
C SINGLE ITEM, TIME VARYING DEMAND, FINITE HORIZON. THE METHODOLOGY
C USED IN THE ALGORITHM IS A DYNAMIC PROGRAMMING AND NETWORK APPROACH
C SUGGESTED BY WILLARD I. ZANGWILL.

```

C
C INPUT VARIABLES
C OP = SET UP COST
C OH = INVENTORY CARRYING COST
C N = NUMBER OF PERIODS
C R = DEMAND
C
C OUTPUTS
C LOT SIZES FOR STAGES AT ALL PERIODS
C
  DIMENSION OP(2),OH(2),F(5),C(100,5,5),I(100,5,5)
  DIMENSION P(100,5,5),GOP(100,5,5),X(5,5),GA(5,5)
  DIMENSION INV(5,5)
  INTEGER A,B,G,GG,AA,GA
  READ (5,1)(OP(J),OH(J),J=1,2)
1  FORMAT(F9.2,F6.3)
  READ(5,2)N
2  FORMAT(I2)
  READ(5,3)(R(J),J=1,N)
3  FORMAT(F7.0)
  J=3
  I=N
  LA1=N*10+J
  C(LA1,N,N)=0
6  I=I-1
  II=I+1
  B=N
5  LA2=I*10+J
  H(LA2,II,B)=HH(I,J,II,B,OH,F)
  IF(II.LE.B)GO TO 4
  LA3=II*10+J
  C(LA3,II,B)=0
4  LA4=I*10+J
  LA5=II*10+J
  C(LA4,I,B)=H(LA4,II,B)+C(LA5,II,B)
  B=B-1
  IF(B.GE.I)GO TO 5
  IF(I.GE.1)GO TO 6
  I=N
  J=2
  LA6=N*10+M
  P(LA6,N,N)=PP(N,M,N,N,OP)
  LA7=N*10+J
  C(LA7,N,N)=P(LA6,N,N)
15 I=I-1
  A=1
  B=I
  II=I+1
11 G=I
10 GG=G+1
  LA8=I*10+3
  P(LA8,I,G)=PP(I,3,I,G,OP)
  LA9=I*10+2
  H(LA9,GG,B)=HH(I,2,GG,B,OH,F)
  IF(GG.LE.B)GO TO 7
  LA10=II*10+2
  C(LA10,GG,B)=0
7  LA11=I*10+3
  LA12=I*10+2
  LA13=II*10+2
  CI2IB=P(LA11,I,G)+C(LA11,2,G)+H(LA12,GG,B)+C(LA13,GG,E)
  IF(G.EQ.I)GO TO 8
  LA14=I*10+2
  IF(CI2IB.GE.C(LA14,I,B))GO TO 9
8  LA15=I*10+2

```

```

C(LA15,I,6)=C12IB
GOP(LA15,I,3)=G
9  G=G+1
   IF(G.LE.8)GO TO 10
   B=B+1
   IF(B.LE.N)GO TO 11
   IF(I.EQ.1)GO TO 12
   A=I+1
16  B=A
14  LA16=I*10+2
   H(LA16,A,3)=HH(I,2,A,B,DH,R)
   IF(A.LE.8)GO TO 13
   LA17=II*10+2
   C(LA17,A,B)=0
13  LA18=I*10+2
   LA19=II*10+2
   C(LA18,A,B)=H(LA18,A,B)+C(LA19,A,B)
   GOP(LA18,A,3)=0
   B=B+1
   IF(B.LE.N)GO TO 14
   A=A+1
   IF(A.GT.N)GO TO 15
   GO TO 16
12  J=1
   I=N
   LA20=N*10+2
   P(LA20,N,N)=PP(N,2,N,N,DP)
   LA21=N*10+1
   P(LA21,N,N)=P(LA20,N,N)+C(LA20,N,N)
22  I=I-1
   A=I
   II=I+1
23  AA=A-1
   CLAVE=1
   G=MAX0(I,AA)
20  GG=G+1
   LA22=I*10+2
   P(LA22,A,G)=PP(I,2,A,G,DP)
   IF(GG.LE.N)GO TO 17
   LA23=II*10+1
   C(LA23,GG,N)=0
17  LA24=I*10+2
   LA25=II*10+1
   C11AN=P(LA24,A,G)+C(LA24,A,G)+C(LA25,GG,N)
   IF(CLAVE.EQ.1)GO TO 18
   LA26=I*10+1
   IF(C11AN.GE.C(LA26,A,N))GO TO 19
18  LA27=I*10+1
   C(LA27,A,N)=C11AN
   GOP(LA27,A,N)=G
   CLAVE=0
19  G=G+1
   IF(G.LE.N)GO TO 20
   IF(I.EQ.1)GO TO 21
   A=A+1
   IF(A.GT.N)GO TO 22
   GO TO 23
21  A=1
   JC 51 I7=1,N
   DO 52 I8=1,2
52  X(I7,I8)=I
51  CONTINUE
   I=1
   J=1

```

```

33 LA28=I*10+J
   GA(I,J)=GQP(LA28,A,N)
   X(I,J)=(
   LATA=GA(I,J)
   DO 24 J7=I,LATA
24  X(I,J)=X(I,J)+R(J7)
   J=J+1
   JJ=J-1
   LATO=GA(I,JJ)
   LATI=I*10+J
   GA(I,J)=GQP(LATI,A,LATO)
   INV(I,J)=GA(I,JJ)-GA(I,J)
   INA=INV(I,J)
   IF(INV(I,J).GT.0)GO TO 25
   CLAVE1=(
   GO TO 26
25  CLAVE1=1
26  X(I,J)=(
   LATE=GA(I,J)
   DO 27 J7=I,LATE
27  X(I,J)=X(I,J)+R(J7)
   A=GA(I,J)+1
31  I=I+1
   II=I-1
   IF(CLAVE1.NE.1)GO TO 32
   IF(A.GT.1)GO TO 31
   LJ=INA+II
   LA29=I*10+J
   GA(I,J)=GQP(LA29,A,LJ)
   INV(I,J)=INV(II,J)-GA(I,J)
   INA=INV(I,J)
   IF(INV(I,J).GT.0)GO TO 28
   CLAVE1=(
   GO TO 29
28  CLAVE1=1
29  X(I,J)=(
   LATU=GA(I,J)
   DO 30 J7=I,LATU
30  X(I,J)=X(I,J)+R(J7)
   A=GA(I,J)+1
   GO TO 31
32  J=1
   IF(I.LE.N)GO TO 33
   DO 34 I=1,N
   WRITE(6,35)I,X(I,1),I,X(I,2)
35  FORMAT(1H1,2JX,"X(",I2,", 1 ) = ",F8.0,10X,"X(",I2,", 2 ) = ",F8.0
   -)
34  CONTINUE
   STOP
   END

```

```

C
C   FUNCTION "DD"
C

```

```

FUNCTION PP(I,J,K,L,CA)
DIMENSION CA(5)
IF(L.GE.K)GO TO 62
PP=0
GO TO 63
62  PP=CA(J)
63  CONTINUE
RETURN
END

```

```

C
C   FUNCTION "HH"

```

```

C
FUNCTION HH(I,J,K,L,AMI,DEM)
DIMENSION AMI(5),DEM(5)
HH=0
IF(K.GT.L)GO TO 73
DO 72 I9=K,L
72 HH=HH+DEM(I9)*AMI(J)
73 CONTINUE
RETURN
END

```

T H E S I S 17

```

PROGRAM THE17 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
C
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF ASSEMBLY STRUCTURE,
C SINGLE ITEM, TIME VARYING DEMAND, FINITE HORIZON. THE METHODOLOGY
C USED IN THE ALGORITHM IS A DYNAMIC PROGRAMMING APPROACH SUGGESTED BY
C WALLACE B. CROWSTON AND MICHAEL H. WAGNER.
C
C INPUT VARIABLES
C N = NUMBER OF STAGES
C M = HORIZON PERIOD
C S = SET UP COST
C H = INVENTORY CARRYING COST
C V = VARIABLE COST
C D = DEMAND
C
C OUTPUTS
C TOTAL COST
C QUANTITY ORDERS FOR STAGES AT ALL PERIODS
C
C
DIMENSION S(10),H(10),V(10),D(5),PAY(5),NPAY(32,5)
DIMENSION NPAY1(32),NPAY2(32,5),ICA(5),ICA01(5),ICA02(5)
DIMENSION ICA03(10,5),CTK(10),ICA04(10,5),Q(10,5)
DIMENSION NORMA(32,5)
READ(5,1)N,M
1 FORMAT(I2,I2)
READ(5,2)(S(I),H(I),V(I),I=1,N)
2 FORMAT(F9.2,F6.3,F6.3)
READ(5,3)(D(J),J=1,M)
3 FORMAT(F7.0)
CLA3=1
CLAVIS=1
DO 4 J=1,M
4 PAY(J)=1
JUL=2
52 CT=F(N,M,D,S(M),V(M),H(M),PAY)
II=1
43 CT1=F(N,M,D,S(II),V(II),H(II),PAY)
JI=0
DO 5 J=2,M
IF(PAY(J).NE.1)GO TO 5
JI=JI+1
5 CONTINUE
IC0=-1
LIM1=2**JI
DO 6 J=1,M
IF(J.EQ.1)GO TO 7
IF(PAY(J).NE.1)GO TO 8
CLAVE=1

```

```

      ICO=ICO+1
      ICO2=1
      ICO3=1
      GO TO 8
7     CLAVE=0
8     CONTINUE
      DO 9 I=1,LIM1
      IF(J.NE.1)GO TO 1E
      NPAY(I,J)=1
      GO TO 9
10    IF(CLAVE.EQ.1)GO TO 11
      NPAY(I,J)=0
      GO TO 9
11    L2ICO=2**ICO
      IF(ICO2.GT.L2ICO)GO TO 12
14    NPAY(I,J)=1
      ICO2=ICO2+1
      GO TO 9
12    IF(ICO3.GT.L2ICO)GO TO 13
      NPAY(I,J)=0
      ICO3=ICO3+1
      GO TO 9
13    ICO2=1
      ICO3=1
      GO TO 14
9     CONTINUE
6     CONTINUE
      IF(CLAVIS.NE.1)GO TO 685
      CLAVIS=1
      LANDA=LIM1
      DO 686 INZ=1,LANDA
      DO 687 INY=1,M
687   NORMA(INZ,INY)=NPAY(INZ,INY)
686   CONTINUE
685   CONTINUE
      I2=0
      DO 15 I=2,LIM1
      TOT=0
      DO 16 J=1,M
16    TOT=TOT+NPAY(1,J)-NPAY(I,J)
      IF(TOT.NE.1)GO TO 15
      I2=I2+1
      NPAY1(I2)=I
15    CONTINUE
      CLA2=1
      DO 17 I=1,I2
      I3=NPAY1(I)
      JK=0
      DO 18 J=2,M
      IF(NPAY(I3,J).NE.1)GO TO 18
      JK=JK+1
18    CONTINUE
      LICO=-1
      DO 19 J=1,M
      IF(J.EQ.1)GO TO 20
      IF(NPAY(I3,J).EQ.1)GO TO 21
      CLAVE=0
      GO TO 20
21    CLAVE=1
      LICO=LICO+1
      LICO2=1
      LICO3=1
20    CONTINUE
      LIM2=2**JK

```

```

DO 22 I9=1,LIM2
IF(J.NE.1)GO TO 23
N2PAY(I9,J)=1
GO TO 22
23 IF(CLAVE.EQ.1)GO TO 24
N2PAY(I9,J)=1
GO TO 22
24 L2LICO=2**LICO
IF(LICO2.GT.L2LICO)GO TO 25
27 N2PAY(I9,J)=1
LICO2=LICO2+1
GO TO 22
25 IF(LICO3.GT.L2LICO)GO TO 26
N2PAY(I9,J)=0
LICO3=LICO3+1
GO TO 22
26 LICO2=1
LICO3=1
GO TO 27
22 CONTINUE
19 CONTINUE
CLA=1
DO 30 I8=1,LIM2
DO 31 J=1,M
31 ICA(J)=N2PAY(I8,J)
CT2=F(N,M,D,S(II),V(II),H(II),ICA)
IF(CLA.EQ.1)GO TO 32
IF(CTMIN.LT.CT2)GO TO 30
GO TO 33
32 CLA=0
33 CTMIN=CT2
DO 34 JCA=1,M
34 ICA01(JCA)=ICA(JCA)
30 CONTINUE
IF(CLA2.EQ.1)GO TO 35
IF(CTOTMI.LT.CTMIN)GO TO 17
GO TO 36
35 CLA2=?
36 CTOTMI=CTMIN
DO 37 JCA=1,M
37 ICA02(JCA)=ICA01(JCA)
17 CONTINUE
CTK(II)=AMIN1(CTOTMI,CT1)
IF(CTOTMI.LT.CT1)GO TO 38
DO 39 JCA=1,M
39 ICA03(II,JCA)=PAY(JCA)
GO TO 41
38 CONTINUE
DO 40 JCA=1,M
40 ICA03(II,JCA)=ICA02(JCA)
41 NN=N-1
IF(II.EQ.NN)GO TO 42
II=II+1
GO TO 43
42 CONTINUE
DO 44 JRI=1,NN
44 CT=CT+CTK(JRI)
IF(CLA3.EQ.1)GO TO 45
IF(CTPI.LE.CT)GO TO 47
GO TO 46
45 CLA3=0
46 CTPI=CT
DO 48 IR=1,NN
DO 49 JR=1,M

```

```

      ICA04(IR,JR)=ICA03(IR,JR)
49  CONTINUE
48  CONTINUE
      DO 50 JR=1,M
50  ICA04(N,JR)=PAY(JR)
47  IF(JUL.GE.LANDA)GO TO 51
      DO 536 JARA=1,M
536  PAY(JARA)=NORMA(JUL,JARA)
      JUL=JUL+1
      GO TO 52
51  I=1
58  J=1
      JJ=J
55  Q(I,J)=D(J)
56  JJ=JJ+1
      IF(JJ.GT.M)GO TO 53
      IF(ICA04(I,JJ).EQ.0)GO TO 54
      J=JJ
      GO TO 55
54  Q(I,J)=Q(I,J)+D(JJ)
      GO TO 56
53  I=I+1
      IF(I.GT.N)GO TO 57
      GO TO 58
57  WRITE(6,59)CTPI
59  FORMAT(1H1,15X,"THE TOTAL COST IS = ",F13.3)
      DO 60 I=1,N
      DO 61 J=1,M
      WRITE(6,62)I,J,Q(I,J)
62  FORMAT(1H0,15X,"STAGE  ",12,5X,"TIME  ",13,5X,"QUANTITY ORDER  ",F
-10.0)
61  CONTINUE
60  CONTINUE
      STOP
      END
C
C      FUNCTION "F"
C
      FUNCTION F(N,M,D,S,V,H,FAYR)
      DIMENSION Q(10),FAYR(10),D(10)
      ID=0
      HT=0
      J=1
      JJ=J
      ST=3
102  Q(J)=D(J)
103  JJ=JJ+1
      IF(JJ.GT.M)GO TO 100
      IF(FAYR(JJ).EQ.0)GO TO 101
      ST=ST+S
      J=JJ
      ID=0
      GO TO 102
101  ID=ID+1
      HT=HT+D(JJ)*ID*H
      Q(J)=Q(J)+D(JJ)
      GO TO 103
100  CONTINUE
      DO 109 I=1,M
109  DT=DT+D(I)
      F=ST+HT+DT*V
      RETURN
      END

```

T H E S I S 18

```
PROGRAM THE18 (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT)
```

```
C
```

```
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF AFFORESCENCE
```

```
C STRUCTURE, SINGLE ITEM, TIME VARYING DEMAND, FINITE HORIZON.
```

```
C THE METHODOLOGY USED IN THE ALGORITHM IS A DYNAMIC PROGRAMMING
```

```
C APPROACH SUGGESTED BY ARTHUR F. VEINOTT, JR.
```

```
C
```

```
C INPUT VARIABLES
```

```
C N = NUMBER OF STAGES
```

```
C M = HORIZON TIME
```

```
C S = SET UP COST
```

```
C H = INVENTORY CARRYING COST
```

```
C V = VARIABLE COST
```

```
C D = DEMAND
```

```
C
```

```
C OUTPUTS
```

```
C TOTAL COST
```

```
C QUANTITY ORDERS FOR STAGES AT ALL PERIODS
```

```
C
```

```
DIMENSION S(10),H(10),V(1),B(10,5),F(10,5,5),JOP(10,5,5)
```

```
DIMENSION Q(10,5)
```

```
READ(5,1)N,M
```

```
1 FORMAT(I2,I2)
```

```
READ(5,2)(S(I),H(I),V(I),I=1,H)
```

```
2 FORMAT(F9.2,F6.3,F6.3)
```

```
DO 4 I=2,N
```

```
READ(5,3)(D(I,J),J=1,M)
```

```
3 FORMAT(F7.0)
```

```
4 CONTINUE
```

```
DO 317 J=1,M
```

```
D(1,J)=V
```

```
DO 316 I=2,N
```

```
316 D(1,J)=D(1,J)+D(I,J)
```

```
317 CONTINUE
```

```
DO 5 K=1,N
```

```
DO 6 I=1,M
```

```
F(K,I,I)=V
```

```
JOP(K,I,I)=I
```

```
6 CONTINUE
```

```
5 CONTINUE
```

```
K=2
```

```
13 I=M-1
```

```
12 M1=M
```

```
11 CLA=1
```

```
J=I+1
```

```
JJ=J+1
```

```
714 IF(JJ.LT.M1)GO TO 10
```

```
F(K,JJ,M1)=0
```

```
10 FKIM1=C(K,I,J,S,V,H,D)+F(K,JJ,M1)
```

```
IF(CLA.EQ.2)GO TO 7
```

```
IF(F(K,I,M1).LT.FKIM1)GO TO 8
```

```
GO TO 3
```

```
7 CLA=0
```

```
9 F(K,I,M1)=FKIM1
```

```
JOP(K,I,M1)=J
```

```
8 J=J+1
```

```
JJ=J+1
```

```

IF(J.LE.M1)GO TO 714
M1=M1-1
IF(M1.GT.1)GO TO 11
I=I-1
IF(I.GT.J)GO TO 12
K=K+1
IF(K.LE.N)GO TO 13
K=1
I=M-1
19 M1=M
CLA=1
J=I+1
JJ=J+1
183 IF(JJ.LT.M1)GO TO 18
F(K,JJ,M1)=0
18 SUF=0
DO 14 K9=2,N
14 SUF=SUF+F(K9,I,J)
FKIM1=C(K,I,J,S,V,H,D)+SUF+F(K,JJ,M1)
IF(CLA.EQ.1)GO TO 15
IF(F(K,I,M1).LT.FKIM1)GO TO 16
GO TO 17
15 CLA=0
17 F(K,I,M1)=FKIM1
JOP(K,I,M1)=J
16 J=J+1
JJ=J+1
IF(J.LE.M1)GO TO 183
I=I-1
IF(I.GT.0)GO TO 19
DO 20 K=1,N
DO 21 J=1,M
Q(K,J)=0
21 CONTINUE
20 CONTINUE
I=1
28 J=M
K=1
I91=I
JENT=JOP(K,I,J)
DO 22 I9=I91,JENT
22 Q(K,I)=Q(K,I)+D(K,I9)
JJ=JOP(K,I,J)
WRITE(6,301)I91,JENT,JJ
301 FORMAT(1H0,10X,"I91 = ",I4,5X,"JENT = ",I4,5X,"JJ = ",I4)
K=2
26 I1=I
25 I81=I1
JENTO=JOP(K,I1,JJ)
WRITE(6,300)I81,JENTO
300 FORMAT(1H0,10X,"I81 = ",I4,10X,"JENTO = ",I4)
DO 23 I8=I81,JENTO
23 Q(K,I1)=Q(K,I1)+D(K,I8)
IF(JENTO.GE.JJ)GO TO 24
I1=JENTO+1
GO TO 25
24 K=K+1
IF(K.LE.N)GO TO 26
IF(JJ.GE.J)GO TO 27
I=JJ+1
GO TO 28
27 WRITE(6,29)F(1,1,M)
29 FORMAT(1H1,15X,"THE TOTAL COST IS = ",F13.3)
DO 60 I=1,N

```

```

DO 61 J=1,M
WRITE(6,62)I,J,Q(I,J)
62 FORMAT(1H6,15X,"STAGE  ",I2.5X,"TIME  ",I3.5X,"QUANTITY ORDER  ",F
-10.0)
61 CONTINUE
60 CONTINUE
STOP
END

```

```

C
C   FUNCTION "C"
C
FUNCTION C(K,I,J,S,V,H,D)
DIMENSION S(10),H(10),V(10),D(10,5)
CONT=0
HT=0
DT=0
IR=I+1
DO 101 I2=IR,J
HT=HT+H(K)*CONT*Q(K,I2)
DT=DT+D(K,I2)
100 CONT=CONT+1
C=S(K)+V(K)*DT+HT
RETURN
END

```

T H E S I S 19

```

PROGRAM THE19 (INPUT,OUTPUT,TAPES=INPUT,TAPE6=OUTPUT)
C
C THIS PROGRAM OBTAINS LOT SIZES FOR THE CASE OF ACYCLIC STRUCTURE,
C SINGLE ITEM, TIME VARYING DEMAND, FINITE HORIZON, THE METHODOLOGY
C USED IN THE ALGORITHM IS AN HEURISTIC APPROACH SUGGESTED BY MICHAEL
C R. LOPEZ.
C
C   INPUT VARIABLES
C       N = NUMBER OF STAGES
C       L = STAGES AT LEVEL=2
C       M = HORIZON TIME
C       S = SET UP COST
C       H = INVENTORY CARRYING COST
C       D = DEMAND
C       A = RELATIONSHIP MATRIX
C
C   OUTPUTS
C       TOTAL COST
C       QUANTITY ORDERS FOR STAGES AT ALL PERIODS
C
DIMENSION S(10),H(10),D(10,5),A(7,7),MABA(32,5),VABA(5)
DIMENSION MABA1(32,5),MABA2(32,5),DP(5),VP(5),VOP(5)
DIMENSION CTIN(10),VOCP(10,5),VOPT(10,5),Q(10,10)
INTEGER A
READ(5,1)N,L,M
1  FORMAT(12,12,12)
READ(5,2)(S(I),H(I),I=1,N)
2  FORMAT(F9.2,F6.3)
LL=L+1
DO 4 I=LL,N
READ(5,3)(D(I,J),J=1,M)
3  FORMAT(F7.0)
4  CONTINUE
DO 5 J=1,L
READ(5,6)(A(I,J),J=LL,N)

```

```

6  FORMAT(5(I1))
5  CONTINUE
   DO 7 I=1,L
   DO 9 K=1,M
   D(I,K)=0
   DO 9 J=LL,N
9  D(I,K)=D(I,K)+D(J,K)*A(I,J)
8  CONTINUE
7  CONTINUE
   DO 10 J=2,M
   LIC02=1
   LIC03=1
   LID0J=2** (J-2)
   LID0M=2** (M-1)
   DO 11 I=1,LID0M
   IF(LIC02.GT.LID0J)GO TO 12
14  MABA(I,J)=1
   LIC02=LIC02+1
   GO TO 11
12  IF(LIC03.GT.LID0J)GO TO 13
   MABA(I,J)=0
   LIC03=LIC03+1
   GO TO 11
13  LIC02=1
   LIC03=1
   GO TO 14
11  CONTINUE
10  CONTINUE
   DO 15 I=1,LID0M
15  MABA(I,1)=1
   C01=1
   I=1
53  CONTINUE
   DO 16 J=1,M
16  VEBA(J)=MABA(I,J)
   C01=0
   DO 17 J=2,M
   IF(VEBA(J).NE.1)GO TO 17
   C01=C01+1
17  CONTINUE
   LD0C01=2**C01
   DO 18 J=1,M
   IF(J.NE.1)GO TO 178
   DO 179 K=1,LD0C01
879  MABA1(K,J)=1
878  CONTINUE
   IF(VEBA(J).EQ.1)GO TO 19
   DO 20 K=1,LD0C01
20  MABA1(K,J)=0
   GO TO 18
19  LIC02=1
   LIC03=1
   LD0C01=2**C01
   DO 21 K=1,LD0C01
   IF(LIC02.GT.LD0C01)GO TO 22
24  MABA1(K,J)=1
   LIC02=LIC02+1
   GO TO 21
22  IF(LIC03.GT.LD0C01)GO TO 23
   MABA1(K,J)=0
   LIC03=LIC03+1
   GO TO 21
23  LIC02=1

```

```

      LICO3=1
      GO TO 24
21  CONTINUE
      CLA=CLA+1
18  CONTINUE
      CO=0
      DO 25 J=1,M
      IF(VEBA(J).NE.0)GO TO 25
      CO=CO+1
25  CONTINUE
      LIDOOO=2**CO
      CLA=0
      DO 26 J=1,M
      IF(VEBA(J).NE.1)GO TO 27
      DO 28 K=1,LIDOOO
28  MABA2(K,J)=1
      GO TO 26
27  LICO2=1
      LICO3=1
      LDOCLA=2**CLA
      DO 29 K=1,LIDOOO
      IF(LICO2.GT.LDOCLA)GO TO 30
32  MABA2(K,J)=1
      LICO2=LICO2+1
      GO TO 29
30  IF(LICO3.GT.LDOCLA)GO TO 31
      MABA2(K,J)=1
      LICO3=LICO3+1
      GO TO 29
31  LICO2=1
      LICO3=1
      GO TO 32
29  CONTINUE
      CLA=CLA+1
26  CONTINUE
      K=1
39  INT=1
      DO 33 J=1,LDOOO1
      DO 34 JK=1,M
      DP(JK)=D(K,JK)
34  VP(JK)=MABA1(J,JK)
      CT=COSTO(M,S(K),H(K),DP,VP)
      IF(INT.EQ.1)GO TO 35
      IF(CTMIN.GE.CT)GO TO 36
      GO TO 33
35  INT=0
36  CTMIN=CT
      DO 37 JK=1,M
37  VOP(JK)=VP(JK)
33  CONTINUE
      CTMIN(K)=CTMIN
      DO 38 JK=1,M
38  VOP(K,JK)=VOP(JK)
      K=K+1
      IF(K.LE.L)GO TO 39
46  INT=1
      DO 40 J=1,LIDOOO
      DO 41 JK=1,M
      DP(JK)=D(K,JK)
41  VP(JK)=MABA2(J,JK)
      CT=COSTO(M,S(K),H(K),DP,VP)
      IF(INT.EQ.1)GO TO 42
      IF(CTMIN.GE.CT)GO TO 43
      GO TO 40

```

```

42 INT=0
43 CTMIN=CT
   DO 44 JK=1,M
44 VOP(JK)=VP(JK)
40 CONTINUE
   CTIN(K)=CTMIN
   DO 45 JK=1,M
45 VOPP(K,JK)=VOP(JK)
   K=K+1
   IF(K.LE.N)GO TO 46
   TOT=0
   DO 47 K=1,N
47 TOT=TOT+CTIN(K)
   IF(CAM.EQ.1)GO TO 48
   IF(TOMI.GE.TOT)GO TO 49
   GO TO 50
48 CAM=0
49 TOMI=TOT
   DO 51 K=1,N
   DO 52 JK=1,M
   VOP1(K,JK)=VOPP(K,JK)
52 CONTINUE
51 CONTINUE
50 I=I+1
   IF(I.LE.LIDON)GO TO 53
   K=1
58 I=1
57 Q(K,I)=0(K,I)
   II=I
56 II=II+1
   IF(II.GT.M)GO TO 54
   IF(VOP1(K,II).EQ.1)GO TO 55
   Q(K,I)=Q(K,I)+D(K,II)
   GO TO 56
55 I=II
   GO TO 57
54 K=K+1
   IF(K.LE.N)GO TO 58
   WRITE(6,59)TOMI
59 FORMAT(1H1,15X,"THE TOTAL COST IS = ",F13.3)
   DO 60 I=1,N
   DO 61 J=1,M
   WRITE(6,62)I,J,Q(I,J)
62 FORMAT(1H1,15X,"STAGE  ",I2,5X,"TIME  ",I3,5X,"QUANTITY OF DR  ",F
-10.0)
61 CONTINUE
60 CONTINUE
   STOP
   END

```

C
C
C

FUNCTION "COSTO"

```

FUNCTION COSTO(M,S,H,D,VEC)
DIMENSION D(5),VEC(5)
ST=0
HT=0
CLA=1
I=1
II=I
ST=ST+S
102 II=II+1
   IF(II.GT.M)GO TO 100
   IF(VEC(II).EQ.1)GO TO 101
   HT=HT+D(II)*H*CLA

```

```
CLA=CLA+1  
GO TO 102  
101 CLA=1  
ST=ST+S  
I=I+1  
GO TO 102  
100 CCSTO=HT+ST  
RETURN  
END
```

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Blackburn, J. D., and H. Kureuther. "Planning Horizons for the Dynamic Lot Size Model with Backlogging," Management Science (21, 3, 1974), 251-255.
2. Clark, A. J., and H. Scarf. "Optimal Policies for a Multi-Echelon Inventory Problem," Management Science (6, 4, 1960), 475-490.
3. Clark, A. J., and H. Scarf. "Approximate Solution to a Simple Multi-Echelon Inventory Problem," Chapter 5 in Arrow, K. J., et al. (Eds.), Studies in Applied Probability and Management Science. Stanford, California: Stanford University Press, 1962.
4. Crowston, W. B., M. H. Wagner, and A. Henshaw. "A Comparison of Exact and Heuristic Routines for Lot-Size Determination in Multi-Stage Assembly Systems," AIIE Transactions (4, 4, 1972), 313-317.
5. Crowston, W. B., M. Wagner, and J. F. Williams. "Economic Lot Size Determination in Multi-Stage Assembly Systems," Management Science (19, 8, 1973), 517-527.
6. Crowston, W. B., W. H. Hausman and W. R. Kampe. "Multi-Stage Production for Stochastic Seasonal Demand," Management Science (19, 8, 1973), 924-935.
7. Crowston, W. B., M. H. Wagner. "Dynamic Lot Size Models for Multi-stage Assembly Systems," Management Science (20, 1, 1973), 14-21.
8. Crowston, W. B., M. H. Wagner. "The Form of the Optimal Solution for the Dynamic Lot-Size Model for Multi-Stage Assembly Systems," (Working paper 72-51) Graduate School of Business Administration, New York University.
9. Gorenstein, S. "Planning Tire Production," Management Science (17, 2, 1970), 372-382.
10. Graves, S. C., and L. B. Schwarz. "Single Cycle Continuous Review Policies for Arborescence Production/Inventory Systems," Management Science (23, 5, 1977), 529-540.
11. Johnson, L. A., and D. C. Montgomery. Operation Research in Production Planning, Scheduling and Inventory Control. New York: John Wiley & Sons, Inc., 1974.

12. Johnson, L. A., "Extension of Material Requirements Planning Systems Using Optimization Methodology," 6th Management Science Colloquium (Osaka, Japan), 1977.
13. Kalymon, B. A., "A Decomposition Algorithm for Arborescence Inventory Systems," Management Science (20, 4, 1972), 860-874.
14. Kortanek, K. O., D. Sodard, and A. L. Soyster. "Multi-Product Production Scheduling Via Extreme Point Properties of Linear Programming," Naval Research Logistics Quarterly (15, 1968), 287-300.
15. Kortanek, K. O., and A. L. Soyster. "On the Status of Some Multi-Product Multi-Period Production Scheduling Models," Management Science (17, 8, 1971), 8560-8561.
16. Love, Stephen F. "A Facilities in Series Inventory Model with Nested Schedules," Management Science (18, 5, 1972), 327-338.
17. Lundin, R. A., and T. E. Morton. "Planning Horizons for the Dynamic Lot Size Model: Zabel Vs. Protective Procedures and Computational Results," Operation Research (23, 4, 1975), 711-734.
18. Manne, A. S. "Programming of Economic Lot Sizes," Management Science (4, 2, 1958), 115-135.
19. May, J. G. "A Linear Program for Economic Lot Sizes Using Labor Priorities," Management Science (21, 3, 1974), 277-285.
20. Newson, E. F. P. "Multi-Item Lot Size Scheduling By Heuristic, Part I: With Fixed Resources," Management Science (21, 10, 1975), 1186-1193.
21. Newson, E. F. P. "Multi-Item Lot Size Scheduling By Heuristic, Part II: With Variable Resources," Management Science (21, 10, 1975), 1194-1203.
22. Schwarz, L. B. "Economic Order Quantities for Products with Finite Demand Horizons," AIIE Transactions (4, 3, 1972), 234-237.
23. Schwarz, L. B. "A Simple Continuous Review Deterministic One Warehouse N-Retailer Inventory Problem," Management Science (19, 5, 1973), 555-566.
24. Schwarz, L. B., and L. Schrage. "Optimal and System Myopic Policies for Multi-Echelon Production/Inventory Assembly Systems," Management Science (21, 11, 1975), 1285-1294.
25. Taha, H. A., and R. H. Skeith. "The Economic Lot Size in Multistage Production Systems," AIIE Transactions (2, 2, 1970), 157-162.

26. Veinott, A. F., Jr. "Minimum Concave Cost Solution of Leontief Substitution Models of Multi-Facility Inventory Systems," Operations Research (17, 2, 1969), 262-291.
27. Zabez, E. "Some Generalizations of an Inventory Planning Horizon Theorem," Management Science (10, 3, 1964), 465-471.
28. Zangwill, W. I. "Single and Multi-Commodity Minimum Concave Cost Flows in Certain Networks," working paper No. 154, Center for Research in Management Science. Berkeley, California: University of California, 1966.
29. Zangwill, W. I. "A Deterministic Multiproduct, Multi-Facility Production and Inventory Model," Operation Research (14, 3, 1966), 486-507.
30. Zangwill, W. I. "A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System - A Network Approach," Management Science (15, 9, 1969), 506-527.