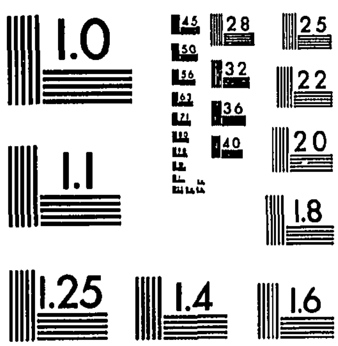
U·M·I



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No 2)

Kim, Seung Baum

SINGLE LEVEL AND MULTI-LEVEL, LOT SIZING STRATEGIES IN MATERIAL REQUIREMENTS PLANNING SYSTEMS

University of Illinois at Urbana-Champaign PH.D. 1986

PLEASE NOTE:

In all cases this material has been filmed in the best possible way from the available copy. Problems encountered with this document have been identified here with a check mark __✓__.

1. Glossy photographs or pages _____
2. Colored illustrations, paper or print _____
3. Photographs with dark background _____
4. Illustrations are poor copy _____
5. Pages with black marks, not original copy _____
6. Print shows through as there is text on both sides of page _____
7. Indistinct, broken or small print on several pages __✓__
8. Print exceeds margin requirements _____
9. Tightly bound copy with print lost in spine _____
10. Computer printout pages with indistinct print _____
11. Page(s) _____ lacking when material received, and not available from school or author.
12. Page(s) _____ seem to be missing in numbering only as text follows.
13. Two pages numbered _____. Text follows.
14. Curling and wrinkled pages _____
15. Dissertation contains pages with print at a slant, filmed as received __✓__
16. Other _____

_____

_____

SINGLE LEVEL AND MULTI-LEVEL LOT SIZING STRATEGIES IN MATERIAL REQUIREMENTS PLANNING SYSTEMS

BY

SEUNG BAUM KIM

B.B.A., Yonsei University, 1975
M.B.A., Washington University, 1978

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Business Administration
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1986

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

February 21, 1986

WE HEREBY RECOMMEND THAT THE THESIS BY

Seung Baum Kim

ENTITLED SINGLE LEVEL AND MULTI-LEVEL LOT SIZING STRATEGIES IN MATERIAL REQUIREMENTS PLANNING SYSTEMS

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Doctor of Philosophy

K Ravi Kumar
Director of Thesis Research

Louis R. Pondy
Head of Department

Committee on Final Examination†

K Ravi Kumar
Chairperson

Donald M. Roberts

Charles E. Blair

† Required for doctor's degree but not for master's

O 517

Abstract

The area of Material Requirements Planning (MRP) is receiving major emphasis in the management of operations due to the financial benefits of efficient deployment of inventory investment. The issue of lot sizing rule has been one of the most heavily addressed topics in the MRP area. Prior research in lot sizing has dealt with single level or multi-level problems with a single parent product structure. However, real world problems frequently involve both multiple levels and multiple parents.

To develop a multi-level lot sizing rule, the dependent demand relationship between parents and components and common usage of certain items in producing different finished products should be considered in some fashion. The objective of this research is to develop multi-level lot sizing rules that are simple to understand and can be implemented with ease on an MRP system. This dissertation proposes a new approach to multi-level lot sizing for MRP systems.

We formulate a mixed integer programming model for the multi-level lot sizing problem to facilitate the development of a multi-level lot sizing algorithm. An examination of sample solutions obtained by mixed integer programming with the ones obtained by a sequential application of any single level lot sizing rules can provide a pattern which would be useful in developing a recursion algorithm. Exploiting the characteristic differences among these solutions, a recursion algorithm is

developed.

Three popular lot sizing rules are modified for the single level, and multi-level problem as well. The three modified lot sizing rules operate on the part period accumulation principle besides their original mechanism.

To evaluate these proposed single level and multi-level lot sizing heuristics we develop a computer simulation model of an MRP system and design experiments designed to consider different demand patterns, different degrees of commonality and different numbers of levels in product structures. Evaluation criteria include inventory costs and computing time requirements.

Dedication

To my parents, Joonho and Germtai Kim, who taught me 'I can,'and to my wife, Sungnam and three children, Younjie, Harry Sangwoo, and Andrew Sangwon, who gave the support so that 'I could.'

Acknowledgements

I wish to extend my sincere appreciation to my chairman, Professor K. Ravi Kumar, who provided invaluable help and insight into this research. I also wish to thank the other members of my committee, Professors Charles Blair and Donald Roberts for their guidance and patience.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Material Requirements Planning

Material requirements planning (MRP) was originally seen as an effective method of ordering inventory. As it evolved, its major emphasis shifted to scheduling, i.e., establishing and maintaining valid due dates on orders.

Today, it has been expanded into manufacturing resource planning (MRP II) to include the effective planning of all the resources of a manufacturing firm. The challenge of 1980s has been to align diverse functions, which a typical manufacturing organization performs, such as, manufacturing, marketing, accounting, finance, engineering, so that their individual goals can be coordinated to meet the business plan through a communication network, an integrated data base. An integrated data base places into a common receptacle the data that is supplied from various areas. Each operating function has access to the total information available from each unit. The evolution of MRP to closed-loop MRP to MRP II results in a single game plan to meet the overall goals of an organization. This is possible because it ties together strategic, financial, and capacity planning areas.

Thus, the term MRP has meant different things to different people. Some think of it as an inventory system, other as a scheduling system, and still others as a complete closed-loop

production system. However, most would agree that MRP tends to become the cornerstone of the production system. MRP would reveal what items are needed, how many are needed, when they will be needed, and when they should be ordered.

Demand for an item may be classified as either independent or dependent depending on whether its demand depends on the demand for other items. The demand for the final product is independent in that it should be forecasted and is not dependent on the demand for other items. On the other hand, the demand for lower level components composing the end product tends to be dependent in that it is generated from the demand for other items. MRP works backward from the scheduled due dates of end items to determine the dates when dependent demand components are ordered. Dependent demand items are calculated by the MRP system from the master schedule. Except for lot sizing economics, dependent demand components should be available when needed, not before and not after. When work cannot be accomplished on time, MRP can reschedule planned orders so that priorities are realistic and meaningful.

The three major inputs of an MRP system are the master production schedule, the product structure records, also known as bills of materials (BOM) records, and the inventory status records. The master production schedule (MPS) outlines the production plan for all end items. The product structure records contain information on all materials, components, or subassemblies required for each end item. The inventory status records contain the on-hand and on-order status of inventory

items. MRP takes MPS for end items and determines the gross requirements for components from the BOM. Gross requirements are obtained by "exploding" the end item product structure record into its lower level requirements. The explosion identifies what components are required, as well as, how many, to produce a given quantity of end items. By referring to the inventory status records, the gross quantities will be netted by subtracting the available inventory items. When to order is determined by offsetting (setting back in time) the lead times for each component. Thus, the material requirements for each component are phased over time in a manner determined by lead times, parent requirements, and inventory status.

The planning horizon of the master production schedule should be large enough to cover the cumulative procurement and production lead times ("stacked" lead times) for all components composing the end products. One-week increments have been found to be the most practical.(Anderson, Schroeder, Tupy, and White (1982), Wemmerlov (1979))

The product structure usually contains several stages from raw materials to subassemblies to assemblies to end items. The end product is designated, by convention, as being at level 0, its immediate components at level 1, and so forth.(see Fig. 1.1) The numbers in the brackets indicate the usage factors, the quantity needed for producing one unit of immediate parent. The usage factor of {1} is usually not indicated.

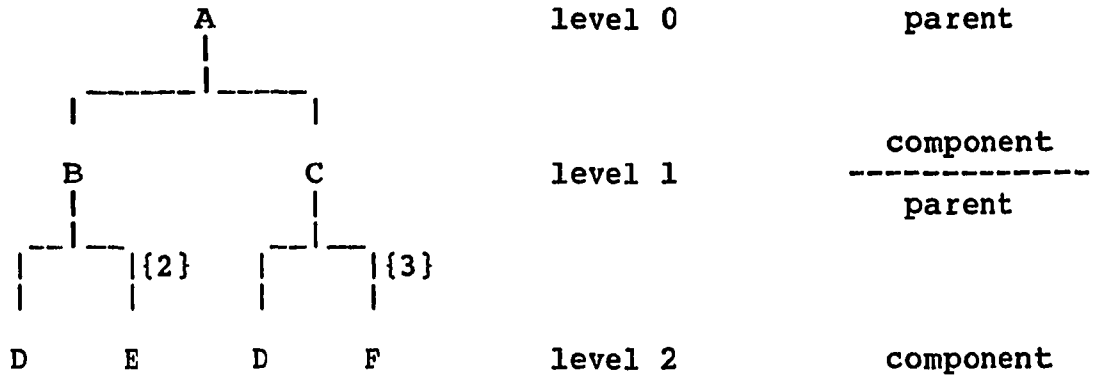


Fig. 1.1 Example Product Structure

1.2 Lot Sizing in the MRP System

In general, the lot sizing problem for discrete demand implies converting a vector of discrete demand for an item into another vector of planned orders by batching these demands into lots. The discrete nature of the demand is characterized by the instantaneous depletion of the item's inventory at certain points in time, e.g., at the beginning of a period, when the item's inventory is drawn for the assembly of higher-level subassembly or assembly. On the contrary, in the case of continuous demand, inventory is consumed gradually at a constant demand rate throughout the period and/or the entire planning horizon.

Generally speaking, there are two types of important costs that are involved in making lot size decisions in a manufacturing situation, namely the setup cost and inventory carrying/holding cost. The setup cost is the fixed cost that is independent of the

size of the replenishment and is associated with a replenishment. It usually includes the cost of processing of production orders, authorization, machine setup, tooling, and even interrupted production. The cost of carrying items in inventory includes the opportunity cost of the inventory investment, warehouse expenses, deterioration of stock, obsolescence, insurance, and taxes. Total inventory cost for an item's certain order schedule is naturally the sum of these two costs incurred for that particular schedule.

Given a demand vector, one possible way of lot sizing is to plan production orders whenever they are required and in exactly the same quantities as required (i.e., Lot for Lot). In this case, total inventory cost for the resultant order vector consists of setup costs only. As no inventory is carried, no carrying cost are incurred. Another possible way is one in which the demands for all the periods in the planning horizon is planned in one single order. This order vector would require only one setup while incurring large carrying costs. These are just two extreme possibilities and various possiblities exist between these two extremes. Figure 1.2 illustrates a trade-off between the two costs, setup and carrying costs for the case of continuous, constant demand. The same would be true for the discrete, constant demand case. Total inventory cost is represented as a funtion of the lot size and the optimal lot size is where the total costs are at their minimum. This occurs where the setup cost and carrying costs are equivalent.

In an MRP system, although gross requirements for components are obtained by exploding the end item product structure record

into its lower level requirements, we still need to decide order quantities for each item in the system. Thus in such system it is necessary to incorporate some lot sizing techniques (even if it is Lot for Lot) at all levels simply in order to generate lower level requirements in the correct time phased format. The application of a particular lot sizing rule to an item at a given level affects the gross requirements for components at subordinate levels. In a multiple level system like the MRP system, the lot sizing problem is to determine the set of order quantities for all items in the system over its planning horizon that minimizes the sum of inventory holding and setup cost incurred in the system.

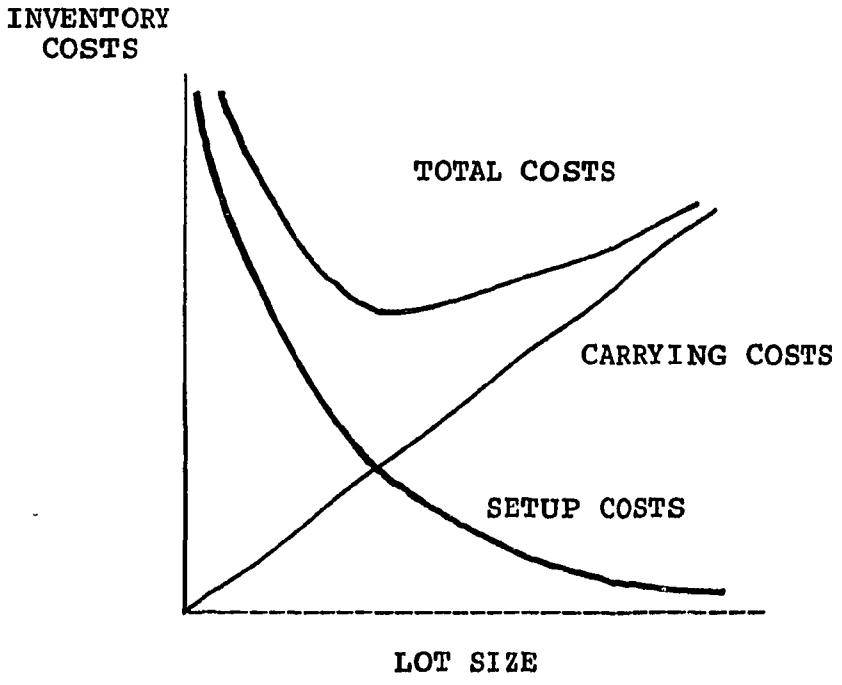


Fig. 1.2 Total Costs as a Function of the Lot Size for the Continuous, Constant Demand

As planned orders for higher level items will generate gross requirements for immediate lower level items, and so on, we should take this inter-level dependent demand relationship and commonnality relationship into account when developing lot sizing rules for the multi-level structure.

Furthermore, there is another factor to be considered before selecting the lot sizing strategy. An MRP system, in practice, would require a considerable amount of computer time to determine schedules of order quantities for items in the whole system. The computational effort depends on the number of items involved and frequency of reschedulings. Thus, time-efficient ordering procedures are of a high priority.

1.3 Objective of this Study

Few areas of management decision making offer more potential for theory development than problems of the design and operation of a multi-level production/inventory system. Furthermore, few areas of management would offer greater pay-offs for the application and execution of good management techniques than inventory control. In order to appreciate this potential, we need to consider the following facts. Approximately one third of the current assets of the average US business firm is devoted merely to inventory. This is about 90% of the same firm's working capital. Most of this inventory is currently being managed within multi-level (multi-echelon) production / inventory systems.

The motivation for this study arises from the realization

that although Material Requirements Planning has been available as a tool for managing manufacturing inventory for over two decades, little work has been published that addresses the problem of deciding how much to produce in a batch for items in multi-level MRP settings.

Most of the recent studies of heuristic lot sizing techniques for multiple-level material requirements planning systems have investigated the application of lot sizing rules derived in the context of a single level.

In reality, there exist vertical interdependencies between levels due to the demand at lower levels being derived by the lot sizing decisions made at immediate higher levels and horizontal commonalities due to existence of some components having two or more parents.

In this study, we aim to formulate a mathematical model of the multi-level lot sizing problem and further develop multi-level lot sizing heuristics that account for these two distinct characteristics (i.e., dependency and commonality). These heuristics should be easy to understand and simple to implement in practice. We also intend to test the proposed heuristics in various combination of factors (i.e., demand pattern, cost structures, degrees of commonality, number of levels, etc.) and compare their performance with other established heuristics and optimum solutions.

Chapter 2

Review of Literature

For the purpose of review, the taxonomy described by Krajewski and Ritzman (1977) is used. (See Figure 2.1.) We categorize lot sizing problems in terms of the number of levels in product structures, the number of products, and the number of parents and components.

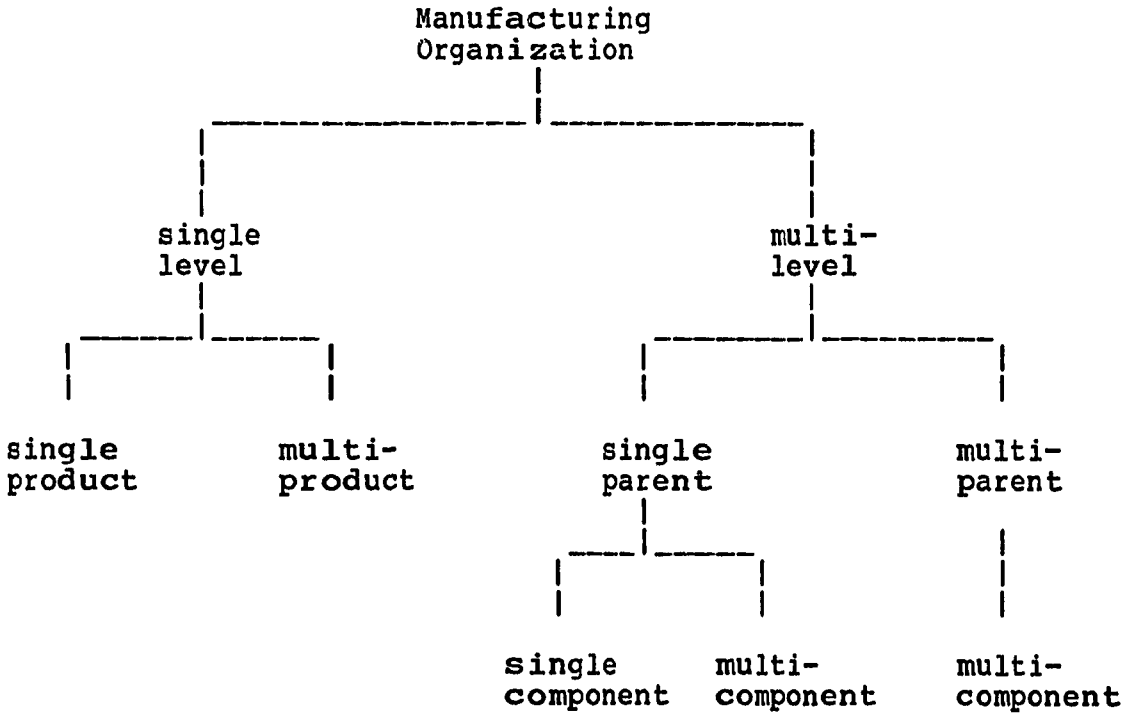


Fig. 2.1 A Taxonomy for Lot Sizing Problems

2.1 Single Level Research

Prout (1973) reports that multi-stage inventory analysis is frequently approached by using a set of single stage, single product inventory models. In practice, most of current MRP users are reportedly adopting single level lot sizing techniques. Thus it is useful to examine single level research.

Harris (1915) is credited with devising the classic Economic Order Quantity (EOQ) model for determining the optimal order quantities when demand is continuous and the steady-state demand rate is known. It is based on the reasoning that optimum, that is minimum inventory cost, is at the point where the inventory carrying cost and setup cost are equal. Although it was originally developed for continuous, constant demand environment, the model has been used in the discrete demand situations in which MRP systems are utilized. The EOQ model no longer provides optimal lot sizes for the discrete demand case.

Wagner and Whitin (1958) considered discrete, deterministic demands but with time varying demand rate over a fixed planning horizon of T periods. In their study, they dropped the assumption of invariable inventory cost from period to period. Allowing no stockouts, and assuming no initial inventory, they developed a dynamic version of the economic lot size model which yields the optimal order quantities for the single level single product problem. The Wagner-Whitin (W-W) algorithm is based on the dynamic programming formulation. Assuming an order must be placed in the first period, there are 2^{T-1} possible lot sizing

combinations, T being total number of periods in the planning horizon, where only those lots consisting of integral number of periods of demand are considered. They showed that it is sufficient to consider only extreme solutions: inventory need not be carried into a period in which a new replenishment order is scheduled to arrive. Further, they proved a planning horizon theorem that, given an optimal schedule for t periods with an order placed in period t, the schedule for the first t-1 periods is optimal.

Presumed computational difficulties with Wagner-Whitin's model have led to a number of heuristic approaches.

Gorham (1968) discussed a method called Least Total Cost (LTC) which goes through the product requirements step by step, accumulating a lot size until a future period t, where the cumulative inventory carrying costs through period t comes closest to the setup cost.

Orlicky (1975) presented the Periodic Order Quantity (POQ) and Lot for Lot (L4L) approaches. The POQ model is a variant of the EOQ model whereby the economic time interval (N^*) between replenishment orders is determined by dividing the economic order quantity obtained from the EOQ model by an average demand per period and rounding it off to the nearest integer. Thus, the order quantity is the sum of demand over the inteval (t, $t+N^*$). On the other hand, the L4L method simply makes the lot size equal to the demand each period.

DeMatteis and Mendoza (1968) developed Part Period Balancing (PPB) algorithm which is based on the same reasoning as the EOQ

model but for a discrete demand setting. PPB heuristic decides order quantities by accumulating requirements until when the cumulative inventory carrying cost nearly equals to, but does not exceed setup cost. The look-ahead and look-backward features are added to account for wide demand variations.

Silver and Meal (1973) outlined a heuristic which selects the order quantity Q so as to minimize the costs per unit time over the time period that Q lasts. Demand for period t is included in the order as long as this results in a reduction in the average cost per period over the interval up to period t.

Morton (1978) developed a dynamic programming algorithm for the model with backlogging. Barbosa and Friedman (1978) proposed a general model with no backorder and a known finite planning horizon. Groff (1979) designed a lot sizing heuristic based on marginal analysis. His heuristic accumulates the demands for consecutive periods as long as the increase in marginal costs is less than the marginal decrease in the setup costs.

All these single level algorithms do not appropriately solve the realistic MRP system problem which necessarily entails multi-level, multi-product situation. However, they are important in that they solve a relatively simple single level problem and may also be applied to the multi-level, multi-product problem.

When numerous items on a single level are subjected to a certain ordering prodecure, the optimal or near-optimal policy is the sum of the optimal or near-optimal policies for the individual items. However, as Krajewski and Ritzman assert, multiple products introduce the complexities of resources

constraints and non-trivial sequencing problems. Several researchers have looked at the case where there is no dependent demand but items are related in other ways. Doll and Whybark (1973) addressed the single machine, multi-product lot scheduling problem where there are several orders that need to be processed on the same machine with limited capacity. They present an iterative procedure for determining the production sequence and lot sizes for the items, using a joint EOQ approach for determining the number of production cycles for planning horizon.

Newson (1975) outlines a heuristic that initially structures the problem as a network of unlimited capacity. The heuristic is extended to include variable capacity constraints. Other studies dealing with single level lot sizing with capacity constraints include Eisenhut (1975), Swoveland (1975), Zangwill (1966). Lambrecht and Vander Eecken (1978) allowed capacity to vary in each period and derived an algorithm to determine dynamic lot sizes by formulating the problem as a minimum cost network flow problem. Silver (1979) presented a dynamic programming algorithm for coordinated replenishment of items, when there is a major setup cost incurred for a family of items and a minor one for items within the family. Other research in this category include Elmaghraby and Bawle (1972), and Simmons (1972).

For the current research, however, it will be assumed that the setup cost for each item is constant and independent of the processing sequence of open orders at a work center. It will also be assumed that there is sufficient capacity to process any

quantity of the items within the pre-specified lead time.

The comparative lot sizing studies (e.g., Berry (1972), Biggs (1975), Groff (1979), Kaiman (1969), Orlicky (1975), Silver et al. (1973), Theisen (1974)) generally confirm the relative total inventory cost results achieved by each lot sizing rule. For single level, single product problems under discrete, deterministic demand and with fixed planning horizons, the W-W rule will yield the lowest total cost solutions, followed closely by S-M heuristic; LTC and PPB heuristics usually rank next, followed by the POQ, EOQ, and L4L rules. The number of calculations, and hence the computational efforts required for the rules generally varies inversely with the quality of their solutions. The W-W rule is the most complex, and is most time consuming while the L4L rule is the simplest to implement.

Most MRP systems utilize one or more of those single level rules to make lot sizing decisions for all the items under their control. However, even Wagner-Whitin algorithm, the best among the single level lot sizing rules, cannot guarantee the best solution for the multi-level lot sizing problem and may not be entirely suited for multi-level MRP settings. Since these rules do not effectively take advantage of the additional information available for the MRP system, we need to design a multi-level lot sizing heuristic that accounts for two distinct characteristics, namely vertical dependency and horizontal commonality.

2.2 Multi-Level Research

There has not been much work published in the area of

multi-level lot sizing rules for MRP systems. Several studies have examined non-MRP based lot sizing in multi-stage, multi-level systems. Those studies, however, failed to address some of the key issues that characterize the lot sizing problem in MRP systems. First, the assumptions made on the product structure examined in those works do not address the typical product structure, namely multi-end product, multi-component, multi-level product structure, of the MRP system. One major issue is that several studies, such as Clark and Scarf (1960), Crowston and Wagner (1973), Love (1972), Schwarz and Schrage (1975), Taha and Skeith (1970), and Zangwill (1966), assume a serial production system, where each item (except end product and raw material) has exactly one predecessor and one successor. The solution procedures devised for the serial, multi-level lot sizing problems are generally not viable in the non-serial assembly structures. Another issue is that of computational efficiency. Optimum-seeking procedures presented in several works, such as Crowston and Wagner (1973), Crowston, Wagner and Henshaw (1972), Kalymon (1972), Love (1972), Pinkus (1975), and Zangwill (1966, 1969), simply require a huge amount of computation time to obtain the optimal lot schedules. Thus it may be impractical to use those optimizing procedures in real settings.

Although these studies failed to directly address our research problem, examining them may still be valid in that we may gain some insights into our problem.

The simplest multi-echelon structure is a serial one. For

the serial product structure, lot sizing problem with uncapacitated stages, concave production and holding costs, and time-varying demand, Zangwill (1969) represented this problem as a single source network. He showed that the optimal solution is contained in the set of extreme point solutions. An extreme point solution is one in which the lot schedule includes the lots consisting of an integral number of periods of demand. Using the extreme point solution concept, he proposed a dynamic programming procedure for deriving the optimal production schedule. Love (1972) also considered the serial structure problem with discrete demand and presented an alternative dynamic programming algorithm. He proved that in a serial system the lot sizes of the component will be an integral multiple of its parent's lot sizes. The amount of computation for both algorithms is bounded by a polynomial in the number of stages and the number of time periods. Lambrecht and Vander Eecken (1978) examined a single-item serial system with time-varying demand for which the last stage has capacity constraints. They characterized the optimal solution in terms of extreme network flows and proposed a decomposition solution procedure in which the problem is divided between the stage with capacity restrictions and the ones without them. Their procedure is to enumerate all extreme solutions for the last processing stage, which is capacitated, and utilize Zangwill's algorithms for solving the other uncapacitated stages. Other approaches in a serial system may be found in Taha and Skeith (1970), and Gabbay (1979). Taha and Skeith also used the integer multiplier concept for a serial system. They assumed a

constant demand rate over an infinite horizon and allowed backorders in the final stage, and delay between stages. Gabbay used linear programming to solve the aggregate problem and then a set of multi-stage problems over a short time horizon, assuming monotone costs for the aggregate problem and separability for disaggregation.

Some researchers considered an assembly type structure where the production of an item requires several components to be assembled in a hierarchical order and each component has at most one successor. The serial structure is a special case of this assembly network.

Crowston, Wagner, and Williams (1973) showed that for single parent multi-component systems, the lot sizes of the component will be an integral multiple of its parent's lot sizes. They give a dynamic programming algorithm for calculating optimal lot sizes when demand is continuous and constant over an infinite horizon. For the same problem, Schwarz and Schrage (1975) first determined an optimal branch and bound procedure and then showed a "system myopic" policy in which lot sizes are determined by integer multipliers that are obtained by considering the parent and its component residing in two adjacent levels in the product structure at a time. Once the lot size for an item is calculated, it will be fixed.

For an uncapacitated assembly system with time-varying, discrete demand over a finite planning horizon, concave production costs, and linear inventory holdling costs, Crowston and Wagner (1973) formulated the lot sizing problem as a dynamic

program by utilizing an extreme point characterization of the optimal solution. Their optimal algorithm is quite complex, with the solution time increasing linearly with the number of items but increasing exponentially with the number of periods in the scheduling horizon. For the more general assembly system, namely multi-parent, multi-component structure like in a typical MRP system, Steinberg and Napier (1980) produced an optimal procedure for the problem by modeling the system as a constrained generalized network with fixed charge arcs and constraints. Other optimizing models addressing the problem may be found in Haeling von Lanzenhauer (1970) and Kalymon (1972). Haeling von Lanzenhauer modeled the multi-parent multi-component lot sizing problem as a 0-1 integer programming problem. Kalymon used decomposition techniques to solve the multi-level lot sizing problem. In all of these optimum seeking algorithm cases for serial and a non-serial assembly systems, the computational inefficiency of obtaining optimal solutions on a recurring basis have prohibited their application in most MRP settings.

In order to reduce the immense computational efforts required with optimizing models, several researchers have taken a heuristic approach to this problem, lot sizing for a serial or assembly system. Their approach to this problem is mostly a sequential application of a single stage lot-sizing method with a set of modified costs that attempt to account for the dependence relationship between neighboring stages. New (1974) proposed a procedure which specifically acknowledges the dependency relationship. For the constant demand case, he uses the concept

of "value added" at each stage in the system to calculate an adjustment factor to be used in the EOQ calculations. Once the order quantity is determined, it wll not change. Based on this modified EOQ calculation, McLaren (1977), and McLaren and Whybark (1976) developed setup cost adjustment mechanism to account for the interdependencies among levels. Their upward adjustment of the setup cost is calculated by looking at each item and its immediate component items as a small system nested in an entire product structure. It is similar to Schwarz and Schrage (1975)'s methodology with respect to considering two adjacent levels as a separate system. This set of adjusted setup costs for each item in the system can be used with any single level lot sizing rules. They also presented both the Order Moment heuristic, which is a single level lot sizing rule, and the Wagner-Whitin model- based multi-level algorithm which does not guarrantee optimality. They conceived their adjustment mechanism from examining the lot sizing patterns of an end item. The product structures examined in their study are ones in which every item in the system has at most one parent. However, their technique can be used without any modification for the multi-parent situation. Blackburn and Millen (1982) propose several ways of modifying both the setup cost and carrying cost to take into account the dependency relationship between two neighboring levels. Their cost modifications are based on the assumptions of an infinite horizon and continuous, constant demand per period. They further assume that the lot size for an item is an integer multiple of the lot size for its parent, which is not valid for the product structure

with items having multiple parents. The setup and carrying cost are modified for each item by considering the item and all lower level items below it. This modification scheme, however, cannot be adapted for application to multi-parent, multi-component systems, as its basic assumption of the lot size for an item being an integral multiple of the lot for its parent is not valid in multi-parent, multi-component structures.

The other approach to the multi-level lot sizing problem offered in the previous literature is to directly apply single level lot sizing heuristics (for example, Economic Order Quantity, Least Total Cost as in Gorham (1968), Periodic Order Quantity as in Orlicky (1975) etc.) to every item in the system sequentially and compare total inventory cost performances of the rules. Collier (1978) chooses five lot sizing methods for use in his comparative study: EOQ, POQ, LTC, L4L, W-W. He classifies the product structure into three general levels: (1) the "top" level consisting of end items whose demand must be forecast; (2) the "intermediate" level consisting of all the manufactured parts, subassemblies, and assemblies; and (3) the "lower" level consisting of raw materials procured from outside. He then evaluates twenty-five combinations of the lot sizing rules applied at the two upper levels for different degrees of end item demand variability and item commonality. Performance was compared in terms of inventory cost and computational effectiveness. Yelle (1979) uses four rules: EOQ, POQ, LTC, L4L, for a single parent, single component structure. He then compared sixteen combinations of these techniques over the two levels under six different

demand patterns for the end item. Choi, Malstrom, Classen (1984) evaluates nine lot sizing rules, including LFL, EOQ, POQ, LTC, LUC, PPB, S-M, W-W, EEH(EOQ/EPQ hybrid algorithm), for a one-end item, 20-component, 3-level product structure. The performance of each lot sizing rule is simulated over nine different sets of market requirements patterns over a twelve time period.

Other comparative studies include Biggs, Goodman, and Hardy (1977), Biggs (1979), and Jacobs and Khumawala (1980).

Chapter 3

Mathematical Model and Heuristics Development for Single Level Environment

The recent growth in the use of material requirements planning (MRP) systems has resulted in increased interest in the topic of lot sizing strategies to be used for every item under the control of MRP system. Management has complete control over what lot size model to choose for each product item or product structure. Most of current MRP users are adopting single level lot sizing techniques. We first present a mathematical model for the single level, single product problem.

3.1 Mathematical model

Our problem is to find a set of values of lot sizes that optimizes our performance criteria, total inventory cost, which consists of the setup cost and carrying cost, for an item for all time periods in the planning horizon. It is constrained to meet demands at least. A mixed integer programming model is formulated for the problem. In the t-th period, t= 1, 2,..., T, we let

X_t = order quantity for the item for period t

$Z (X_t)$ = zero-one variable for setup

S = setup cost for the item

C = carrying cost for the item / unit / period

I_t = ending inventory of the item at the end of period t

D_t = demand for the item for period t

L = lead time for the item

We may write our objective function for planning horizon of T periods as

$$\text{Minimize} \quad \sum_{t=1}^{T} \left(S * Z(X_t) + C * I_t \right) \tag{3.1}$$

subject to

$$I_t = I_{t-1} + X_{t-L} - D_t \tag{3.2}$$

$$Z(X_t) = \begin{cases} 0 & \text{if } X_t = 0 \\ 1 & \text{if } X_t > 0 \end{cases} \tag{3.3}$$

$$X_t - M * Z(X_t) < 0 \tag{3.4}$$

$$I_t > 0 \quad \text{for all} \quad t \tag{3.5}$$

$$X_t > 0 \quad \text{for all} \quad t \tag{3.6}$$

Since the early work of Harris (1915), numerous articles and papers have been written in the past few years on the subject of lot sizing heuristics and their cost performance in single stage or multi-stage situation. The classic Economic Order Quantity (EOQ) model was developed for determining the optimal order quantities when demand is continuous and the steady-state demand rate is known. Wagner-Whitin (1958) developed a dynamic version of the economic lot size model which drops the assumption of a steady-state demand rate and invariable inventory cost from period to period, and considers discrete, deterministic demands but with time-varying demand rate. It is an optimum-seeking algorithm for the single level single product problem.

Presumed computational difficulties with Wagner-Whitin's model have led to a number of heuristic approaches. Those

approaches include Least Total Cost (LTC) (Gorham, 1968), Part Period Balancing (DeMatteis and Mendosa, 1968), Silver and Meal (1973), Groff (1979), Lot for Lot (LFL) and Periodic Order Quantity (POQ). Currently, MRP users are using single stage lot size models such as LFL, EOQ, POQ, LTC within the MRP system (American Production and Inventory Control Society (1974), Berry (1972), Biggs et al. (1977)).

The purpose of this chapter is to develop and test modifications to existing single stage lot sizing heuristics which produce enhanced cost performance for a single stage situation. In the next section, we present new modified versions of the widely used lot sizing rules of EOQ, POQ, and LTC, together with a numerical illustration of their application. This is followed by a cost comparison of the modified heuristics with their original version and W-W approach in the problem set of numerical examples presented in the paper of Berry (1972).

3.2 Lot Sizing Techniques and their Modifications

Economic Order Quantity (EOQ) vs. Modified EOQ (MEOQ)

EOQ model determines the optimal order quantity when demand is continuous, constant, and known, and where total inventory costs are used as the optimality criterion.

The EOQ lot sizing rule is to always order the economic order quantity (Q^*) calculated from :

$$Q^* = \sqrt{\frac{2\,D\,S}{C}} \qquad (3.7)$$

where D = Average demand / period

S = Setup cost

C = Carrying cost / unit / period

When the on-hand inventory at a period is less than demand for the next period, an order of Q^* units is placed. If an order of Q^* is not sufficient for meeting the demand for the period, the order is increased to the level of the gross requirement to prevent shortages.

It should be noted that as the total inventory cost formula assumes continuous, level demand, it is only an approximation to the total cost function for the case of discrete, time-varying demand. Furthermore, with the EOQ model, we cannot guarantee that inventory carryover will not occur. In most of the periods within the planning horizon, an excess inventory is carried into a period in which a new replenishment order is scheduled to arrive. Carrying costs are incurred for the units carried into that period. This unnecessary waste results from the static model ignoring actual time-varying demand pattern.

It is sufficient to consider rules in which inventory should never be carried in a period in which an order is scheduled to arrive. As all dynamic lot sizing rules do , an EOQ model for discrete demand environment needs to place an order covering an integral number of periods of demand. The modification to the original EOQ checks to see how many periods of demand should be included in each lot. Thus it is named Modified Economic Order Quantity (MEOQ).

When an order is to be placed (i.e., when on-hand inventory

reaches zero and current demand is positive), MEOQ determines each lot size according to the following algorithm :

Step 1. Accumulate demand period by period and check whether the cumulative demand equals or exceeds the EOQ (obtained from eq. (3.7)) for the first time. Suppose the test is being done in period K.

Is $\sum_{t=R}^{K} D_t > EOQ$?

where R is the period in which a replenishment order is to be placed.

If Yes and it equals EOQ, then order $\sum_{t=R}^{K} D_t$

If Yes and it exceeds EOQ, go to step 2.

If No, go to step 3.

Step 2. Compute part period of period K and compare this with EPP[1].

Is $(K - R) D_K > EPP$

If Yes, order $\sum_{t=R}^{K-1} D_t$ and go to step 1.

If No, order $\sum_{t=R}^{K} D_t$ and go to step 1.

---

[1] If the number of units of demand carried in inventory is multiplied by the number of periods carried, the result is called part period for the demand. EPP, the quantity of the inventory item which, if carried in inventory for one period, would result in a carrying cost equal to the setup cost, is computed by :

$$EPP = \frac{S}{C} \qquad (3.8)$$

where S : setup cost
C : carrying cost / unit / period

Step 3. Compute part period of period K and compare this with EPP.

Is $(K - R) D_k > EPP$?

If Yes, order $\sum_{t=R}^{K-1} D_t$

If No, move to next period by setting K = K + 1 and go to step 1.

Numerical Example.

Without loss of generality, we assume that shipments are received in the same period as the orders i.e., lead time is zero.

Setup cost = $ 206

Carrying cost / unit / period = $ 2

EOQ = 138 units

EPP = 103 units

When EOQ is applied,

DEM	80	100	125	100	50	50	100	125	125	100	50	100
INV	58	96	109	9	97	47	85	98	111	11	99	137
ORD	138	138	138	0	138	0	138	138	138	0	138	138

Carrying cost : $ 1914

Setup cost : 1854

Total cost : 3768

When MEOQ is applied,

DEM	80	100	125	100	50	50	100	125	125	100	50	100
INV	100	0	100	0	50	0	125	0	100	0	100	0
ORD	180		225		100		225		225		150	0

Carrying cost : $ 900

Setup cost : 1442

Total cost : 2342

Step 1. R = 1 and K = 1

Is D_1 (=80) > EOQ (=138) ?

No.

Go to step 3.

Step 3. Is (1 - 1) D_1 (=80) > EPP (=103) ?

No.

K = 2 and go to step 1

Step 1. Is D_1 (=80) + D_2 (=100) > EOQ (=138) ?

Yes, it exceeds.

Go to step 2.

Step 2. Is (2 - 1) D_2 (=100) > EPP (=103) ?

No.

Order $D_1 + D_2$ = 180 in period R (=1).

Step 1. R = 3 and K = 3

Is D_3 (=125) > EOQ (=138) ?

No.

Go to step 3.

Step 3. Is (3 - 3) D_3 (=125) > EPP (=103) ?

No.

K = 4 and go to step 1.

Step 1. R = 3 and K = 4.

Is D_3 (=125) + D_4 (=100) > EOQ (=138) ?

Yes, it exceeds.

Go to step 2.

Step 2. Is (4 - 3) D_4 (=100) > EPP (=103) ?

No.

Order $D_3 + D_4 = 225$ in period R (=3).

Continuing in this fashion, we will have MEOQ order schedule. It can be seen that EOQ results in a total cost which is 76% over MEOQ.

Periodic Order Quantity (POQ) and Modified POQ (MPOQ)

Another popular modification to the EOQ rule for use in an environment of discrete demand, termed Periodic Order Quantity, is obtained by converting the units given by EOQ to an equivalent number of periods of average demand.

The POQ algorithm first calculates the EOQ. Then this order quantity is divided by average demand per period and rounded off to the nearest integer value, N^* to determine ordering interval. An order is placed with lot size equal to N^* periods of positive demand. Since this method prevents inventory carryover, it is more effective than the EOQ in obtaining schedules with lower inventory carrying costs. However, like the EOQ model from which it is derived, it may be penalized for placing N^* periods of demand automatically without considering the actual demand pattern.

Thus a modification to the POQ algorithm checks to see whether there can be any possible improvement by moving original order points set by POQ rule either backward or forward depending on the actual time-varying demand pattern. The Modified Periodic Order Quantity (MPOQ) algorithm first computes economic

ordering interval, N^* as POQ does. Further steps in the algorithm are :

Step 1. Accumulate demand period by period spanning over N^* periods from an order point period R up to period K ($= R + N^* - 1$).

Step 2. Compute the part period for period K and compare this with EPP.

Is $(K - R) D_t >$ EPP ?

If Yes, order $\sum_{t=R}^{K-1} D_t$

go to step 1.

If No, go to step 3.

Step 3. Is $\sum_{t=R}^{K} D_t > EOQ$?

If Yes, order $\sum_{t=R}^{K} D_t$

go to step 1.

If No, move to next period by setting $K = K + 1$ and go to step 2.

Numerical Example.

Without loss of genarality, we assume that shipments are received in the same period as the orders, i.e., lead time is zero.

Setup cost = $ 200

Carrying cost = $ 2

EOQ = 110 units

Ordering Interval (N^*) = 2 periods

EPP = 100 units

When POQ is applied,

DEM	30	100	40	110	0	50	100	20	80	40	110	40
INV	100	0	110	0	0	100	0	80	0	110	0	0
ORD	130		150			150		100		150		40

Carrying cost : $ 1000

Setup cost : 1200

Total cost : 2200

When MPOQ is applied,

DEM	30	100	40	110	0	50	100	20	80	40	110	40
INV	0	40	0	0	0	0	20	0	40	0	40	0
ORD	30	140		110		50	120		120		150	

Carrying cost : $ 280

Setup cost : 1400

Total cost : 1680

Step 1. As $N^* = 2$ and $R = 1$, $K = 2$.

Step 2. Is $(2 - 1) D_2$ (=100) > EPP (=100) ?

Yes.

Order D_1 (=30) in period R (=1).

Step 1. As $N^* = 2$ and $O = 2$, $K = 3$.

Step 2. Is $(3 - 2) D_3$ (=40) > EPP (=100) ?

No.

Go to step 3.

Step 3. Is D_2 (=100) + D_3 (=40) > EOQ (=110) ?

Yes.

Order $D_2 + D_3$ (=140) in period R (=2).

Continuing computation in this fashion, it is shown that basic POQ results in a total cost which is 31% over MPOQ.

Least Total Cost (LTC) and Modified LTC (MLTC)

LTC determines lot sizes so that the carrying cost incurred for each lot is about equal to the setup cost. This rule first computes economic part period (EPP). The algorithm accumulates consecutive periods' demands until the cumulative part-periods exceeds the EPP. Suppose the cumulative part-periods exceed the EPP at the N th period. With this algorithm, an order is placed for the next N or N - 1 periods in period R, the order point, depending on whether either the cumulative part-periods up to period R + N or the ones up to R + N - 1 is closer to EPP.

LTC is an extension of the EOQ model for discrete demand environment in that it intends to minimize total inventory costs over the planning horizon by equalizing order costs and carrying costs. However, it also can be penalized for ignoring the given demand pattern. Experience with the basic LTC heuristic has indicated that its use can lead to non-trivial cost penalties if it handles demand pattern with some sporadic rapid seasonal increases over the planning horizon.

When, among the periods included in one lot, ending period's part period happen to exceed the EPP for the item, maintaining the period in the lot would incur carrying cost bigger than setup cost. Thus if the last period, whose part period is greater than EPP, is eliminated from the lot, carrying costs for the demand would be saved, while incurring smaller setup cost.

Our modified LTC algorithm is as follows,

Step 1. For each time period T,

calculate part period D_T (T - R), where R is the period in which a replenishment order is placed.

Is D_T (T - R) > EPP ?

If No, go to step 2.

If Yes, for the first time in period N,

then order $\sum_{t=R}^{N-1} D_t$

Step 2. Accumulate part period and check whether the cumulative part period exceeds the EPP.

Is $\sum_{t=R}^{T} D_t$ (t - R) > EPP ?

If No, set T = T + 1 and go to step 1.

If Yes, for the first time in period N,

$$\text{set } A = EPP - \sum_{t=R}^{N-1} D_t \ (t - R)$$

$$B = \sum_{t=R}^{N} D_t \ (t - R) - EPP$$

$$\text{If } A < B, \text{ then order } \sum_{t=R}^{N-1} D_t$$

$$A > B, \quad \text{order } \sum_{t=R}^{N} D_t$$

In this algorithm, Step 2 is the basic LTC algorithm while Step 1 is the modification.

Numerical example.

Without loss of generality, we assume that shipments are received in the same period as the orders, i.e., lead time is zero.

Setup cost = $ 200

Carrying cost/ unit / period = $ 2

EPP = 100 units

When LTC is applied,

DEM	20	30	80	0	50	10	90	40	0	70	60	100
INV	110	80	0	0	100	90	0	70	70	0	100	0
ORD	130				150			110			160	

Setup cost : $ 800

Carrying cost : 1240

Total cost : 2040

When MLTC is applied,

DEM	20	30	80	0	50	10	90	40	0	70	60	100
INV	30	0	0	0	10	0	40	0	0	60	0	0
ORD	50		80		60		130			130		100

Setup cost : $ 1200

Carrying cost : 280

Total cost : 1480

Step 1. R = 1

T = 1

D_1 (=20) (1 - 1)

Is D_1 (=20) (1 - 1) > EPP (=100) ?

No.

Go to step 2.

Step 2. Is D_1 (1 - 1) > EPP (=100) ?

No.

Set T = 2 and go to step 1.

Step 1. T = 2

D_2 (=30) (2-1)

Is D_2 (=30) (2 - 1) > EPP (=100) ?

No.

Go to step 2.

Step 2. Is D_1(=20) (1 - 1) + D_2(=30) (2 - 1) > EPP (=100) ?

No.

Set T = 3 and go to step 1.

Step 1. T = 3

D_3 (=80) (3-1)

D_3 (=80) (3 - 1) > EPP (=100) ?

Yes, for the first time N = 3.

Order D_1 + D_2 (=50) in period R (=1).

Continuing computation in the same way, it is shown that basic LTC results in a total cost which is 31% over MLTC.

3.3 Experimental Investigation

In order to compare the cost performances of the three modified heuristics with their original version and with the optimal Wagner-Whitin algorithm, we use the experimental framework presented in Berry's (1972) article. Berry has

suggested a general experimental framework within which to compare systematically the various lot sizing procedures that have been proposed. This problem set has been used by Silver and Meal (1973) and Groff (1979).

There are 25 sample problems which are made from a combination of five different demand patterns (see Table 3.1) and five different time between order (TBO) values (see Table 3.2). TBO is defined as the expected coverage duration of a lot (EOQ) and calculated as follows,

$$TBO = \frac{EOQ}{D} = \sqrt{\frac{2\,S}{D\,C}} \qquad (3.9)$$

Table 3.1

DEMAND PATTERNS FOR INVESTIGATION

Period	1	2	3	4	5
1	92	80	10	10	0
2	92	100	80	10	0
3	92	125	180	15	0
4	92	100	80	20	0
5	92	50	0	70	0
6	92	50	0	180	1105
7	92	100	180	250	0
8	92	125	150	270	0
9	92	125	10	230	0
10	92	100	100	40	0
11	92	50	180	0	0
12	93	100	95	10	0
Sum	1105	1105	1105	1105	1105
Coefficient of Variation	0	.293	.718	1.410	3.310

Table 3.2

INVENTORY COST PARAMETERS

TBO	EOQ	S	C	EPP
0.73	66	$ 48	$ 2	24
1.00	92	92	2	46
1.14	105	120	2	60
1.50	138	206	2	103
1.80	166	300	2	150

The percentage increases in total inventory costs over the optimal Wagner-Whitin method is presented in Table 3.3. The three improved heuristics outperform their basic counterparts. They also compare very favorably with the optimal Wagner-Whitin solution.

Their average percentage increases are reduced rather dramatically from their corresponding heuristics' performance,as seen in Table 3.4. For the case of MEOQ, it reduced from 44.54% to 0.78%, for MPOQ, from 7.61% to 1.41%, and for MLTC, from 6.42% to 0.53%. Table 3.5 presents the number of occasions in which the modified algorithms outperform their predecessors. For the case of MEOQ, it outperformed EOQ in 18 cases, and tied in 7. MPOQ surpassed POQ in 12 occasions and tied in 13 while MLTC outperformed LTC in 13 cases and tied in 12. On no occasion did the basic algorithms outperform their modified versions. All the ties are the same as the optimal in the Wagner-Whitin solution.

When excluding the cases of tie in which both modified and original heuristics provide optimal solutions, the reduction is more remarkable. With the MEOQ model, it is reduced from 61.86% to 1.09%, with the MPOQ, from 15.85% to 2.94%, and with the MLTC, from 12.35% to 1.02%. (see Table 3.6) Average reduction rate is 98%, 81%, and 92% respectively. The three modified rules provide the same solution as the optimal one in more than 84% of the sample problems, as seen in Table 3.7.

3.4 Conclusion

We generated modified versions of three lot sizing rules, namely EOQ, POQ and LTC, which are popularly used in current MRP settings. The common problem with EOQ and POQ is that they have rigid policies which have been instituted on the basis of average demand patterns. The modifications suggested improve the basic rules by looking at the demand varitions from period to period through Economic Part Period (EPP) computations. The LTC rule, while looking at EPP, heuristically decides on whether to choose the last period to be in the lot size. Our modification seems to enhance this rule by avoiding inclusion of periods whose part periods are larger than EPP in the lot size.

Each modified heuristic dominates its predecessor in a cost comparison in our experimental setup. Average cost reduction rates from their predecessors' are very high ranging from 81% to 98%. The chances that they provide optimal solutions are also very high ranging from 84% to 88%. An average cost penalty of

Table 3.3

PERCENTAGE INCREASES (OVER W-W METHOD IN TOTAL COSTS OF SETUP AND CARRYING INVENTORY).

Coefficient of Variation (C_v)

TBO	RULE	0	.293	.718	1.410	3.310
	EOQ	0.0	22.22	48.67	59.92	0.0
	MEOQ	0.0	0.0	0.0	0.0	0.0
0.73	POQ	0.0	0.0	6.19	9.09	0.0
	MPOQ	0.0	0.0	0.0	0.0	0.0
	LTC	0.0	0.0	0.0	8.26	0.0
	MLTC	0.0	0.0	0.0	0.0	0.0
	EOQ	0.0	63.95	76.42	85.41	0.0
	MEOQ	0.0	0.0	0.0	3.35	0.0
1.00	POQ	0.0	0.0	8.49	21.05	0.0
	MPOQ	0.0	0.0	0.0	7.17	0.0
	LTC	0.0	1.45	16.03	12.92	0.0
	MLTC	0.0	0.0	0.0	7.17	0.0
	EOQ	74.03	41.43	97.27	92.31	0.0
	MEOQ	0.0	0.0	0.0	1.92	0.0
1.14	POQ	0.0	2.86	9.09	26.92	0.0
	MPOQ	0.0	0.0	0.0	7.69	0.0
	LTC	26.81	24.29	13.64	3.85	0.0
	MLTC	0.0	0.0	0.0	1.92	0.0
	EOQ	38.00	67.62	49.60	70.94	0.0
	MEOQ	0.0	4.18	10.19	0.0	0.0
1.50	POQ	0.0	6.14	15.51	44.42	0.0
	MPOQ	0.0	0.0	0.0	9.26	0.0
	LTC	0.0	6.14	15.51	9.77	0.0
	MLTC	0.0	4.18	0.0	0.0	0.0
	EOQ	38.13	46.78	65.06	75.70	0.0
	MEOQ	0.0	0.0	0.0	0.0	0.0
1.82	POQ	0.0	0.0	7.73	32.71	0.0
	MPOQ	0.0	0.0	0.0	11.21	0.0
	LTC	0.0	0.0	7.73	14.01	0.0
	MLTC	0.0	0.0	0.0	0.0	0.0

Table 3.4

AVERAGE PERCENTAGE INCREASES OVER W-W

ORIGINAL		IMPROVED	
EOQ	44.54%	MEOQ	0.78%
POQ	7.61	MPOQ	1.41
LTC	6.42	MLTC	0.53

Table 3.5

MODIFIED VS. ORIGINAL

	WIN	LOSE	TIE*
MEOQ vs. EOQ	18	0	7
MPOQ vs. POQ	12	0	13
MLTC vs. LTC	13	0	12

* All the ties are the same as the optimal W-W solution.

Table 3.6

AVERAGE PERCENTAGE INCREASES OVER W-W METHOD WHEN THE CASES OF TIE ARE EXCLUDED

Original		Improved	
EOQ	61.86%	MEOQ	1.09%
POQ	15.85	MPOQ	2.94
LTC	12.35	MLTC	1.02

Table 3.7

FREQUENCIES THAT EACH RULE PROVIDES THE SAME OPTIMAL SOLUTION AS THE W-W

MEOQ : 21 out of 25 (84%)

MPOQ : 21 out of 25 (84%)

MLTC : 22 out of 25 (88%)

0.53%, 0.78%, and 1.41%, respectively, over the optimal solution is an extremely encouraging result. Considering the high cost of computation, these modified heuristics seem worthy of extensive usage in practice. They can be applied sequentially to the items in the multi-level product structures to solve multi-level lot sizing problems and will be used together with our proposed multi-level heuristic to examine their performance in multi-level setting.

Chapter 4

Mathematical Model and Heuristic Generation for Multi-Level Environment

Chapter 1 presented a description of the problem of lot sizing for multi-level product structures such as in MRP systems, and Chapter 2 reviewed the literature related to the problem. The main purpose of this chapter is: (1) the development of a simple multi-level lot sizing algorithm; and (2) the design of experiments to test the proposed algorithm.

In the first section, the mathematical model is formulated and introduced to indicate the necessity for the development of simple heuristics. This is followed by sections on the development of the proposed heuristic and on a preliminary experimental investigation to find how it performs. The last section includes the research methodlogy and the specific hypotheses to be tested in the formal experiments.

4.1 Mathematical Model

Our problem is to find a set of values for lot sizes that optimizes our performance criteria, which is total inventory cost, consisting of the setup cost and carrying cost, for all items in the system and for all time periods in the planning horizon. It is constrained to at least meet demands.

A mixed integer programming model is formulated for the problem. In the t-th period, t= 1, 2,...,T, we let

N = number of items in the system

T = number of time periods in the planning horizon

X_{it} = order quantity for item i for period t

$Z (X_{it})$ = zero-one variable for setup

S_i = setup cost for item i

C_i = carrying cost for item i / unit / period

I_{it} = ending inventory of item i at the end of period t

L_i = lead time for item i

U_{ij} = usage factor (i.e., quantity of item i needed for producing one unit of item j) which can be found from the bill of materials information

D_{it} = independent demand for item i in period t

$P(i)$ = set of the immediate parents of item i

We may write our objective function for N items and for planning horizon of T periods as

$$\text{Minimize} \sum_{i=1}^{N} \sum_{t=1}^{T} (S_i \, Z (X_{it}) + C_i \, I_{it}) \qquad (4.1)$$

subject to

$$I_{it} = I_{it-1} + X_{i(t-L)} - \sum_{j \in P(i)} U_{ij} \, X_{jt} - D_{it} \qquad (4.2)$$

$$Z (X_{it}) = \begin{cases} 0 & \text{if } X_{it} = 0 \\ 1 & \text{if } X_{it} > 0 \end{cases} \qquad \text{for all i} \qquad (4.3)$$

$$X_{it} - M * Z (X_{it}) < 0 \qquad (4.4)$$

where M is a large number greater than $\max.(\sum_{t=1}^{T} D_{it})$

$$I_{it} > 0 \qquad \text{for all i, t} \qquad (4.5)$$

$$X_{it} > 0 \qquad \text{for all i, t} \qquad (4.6)$$

The objective is to minimize the total inventory cost, which consists of the setup cost and carrying cost as represented by two terms in the objective function (4.1). A setup cost is incurred whenever an order is placed and is independent of the size of order. Inventory carrying costs are directly proportional to the ending inventory in each period. No production cost term is included for it is not a function of lot size.

First constraint (4.2) describes how inventory level changes each period for each item. In (4.2), for an item i, the beginning inventory plus current production less current demand equals the ending inventory. Current production consists of $X_{i(t-L)}$, the order quantity whose order was placed L_i (lead time for item i) periods prior to t and was planned to be added to the inventory for the period t. On the other hand, usage includes the demand for the item to satisfy the order release from immediate parents and independent demand for item i as service parts. For the end items in particular, the relation (4.2) can be stated as follows,

$$D_{i,t} = I_{i,t-1} + X_{i,t} - I_{i,t}$$

where i represents end items only.

The demands for the end items, $D_{i,t}$, are obtained directly from the known master production schedule and can be treated as constants.

Relation (4.4) forces the 0-1 variable, Z (X_{it}), to be 1 whenever an order is placed. Z(X_{it}) will ordinarily have value 0 due to its positive coefficient in the objective function, which is to be minimized. When X_{it} is positive, however, relation

(4.4) is violated unless $Z(X_{it})$ is set to value 1. Since the largest possible value for X_{it} arises when a single order placed in the first period satisfies the entire T periods of demand included in the planning horizon, the number M must be at least as large as the total sum of demand over T periods regardless of item, so that relations (4.3) and (4.4) are not violated.

The non-negativity relations (4.5) and (4.6) are used to ensure that all demand is to be met while maintaining a feasible production schedule. And there will be no stockouts and backlogs. This is consistent with current MRP systems.

The problem represented by relations (4.1) - (4.6) can be solved using mixed integer programming. The size of the problem depends on both the number of items (N) in a system and the number of periods (T) in the planning horizon. The problem has N*T binary integer variables ($Z(X_{it})$), 2*N*T continuous variables (X_{it} and I_{it}). Assuming that the lot sizes and inventories will be large enough to be approximated by rounding off their real number values to obtain integer values, there still are N*T integer variables. This makes integer programming computationally infeasible for calculating optimal lot sizes for a practical situation, where N is in the thousands, and T is typically 52 weeks or more.

Solution time for mixed integer programing problem tends to be very high. Our previous computational experience with the MIP supports this observation. A set of problems whose N is 6 and T is 6 was solved by using MIP. Since obtaining the optimal solution of an MIP problem about the same size as our problems

would require unexpectedly huge computing time and costs, we had to stop at some intermediate stage of computation, that is, the third integer solution as in our preliminary investigation. Computing time needed to obtain the third integer solution of MIP with APEX III package on CDC CYBER ranged from 4.9 to 105 CPU seconds of execution time (average time 33.88 CPU seconds of execution time per problem).

4.2 The Proposed Heuristic

The direction taken in our study is toward the development of simple rules that yield near optimal lot sizes, with regard to inventory costs, and yet are computationally practical for MRP systems. Heuristic lot sizing rules may be well suited for use in MRP systems in that they may offer a great possibility of computational efficiency without great sacrifice of solution quality.

Several papers have pursued this direction in previous research. The most popular approach to the multi-level lot sizing problem is a sequential application of a single stage lot sizing rule with a set of modified costs that attempt to account for the dependence relationship between neighboring stages. New (1974) has presented a procedure which specifically acknowledges the dependency relationship. For the constant demand case, he uses the notion of "value added" at each stage in the system to calculate an adjustment factor to be used in the EOQ calculations.

McLaren (1977), and McLaren and Whybark (1976) developed

setup cost adjustment mechanism to account for the interdependencies among levels. The setup cost adjustment mechanism allocates a proportion of all immediate components' setup cost to the production of the parent part. The proportionality constants are ratios of the time between orders (TBOs). Then the weighted setup cost is the sum of the parent setup cost plus the allocated portions of its immediate components setup costs. It can be shown as

$$S_i' = S_i + \sum_{j \in B(i)} \left(\frac{TBO_i}{TBO_j} S_j \right) \qquad (4.7)$$

where $$TBO_j = \sqrt{\frac{2\ S_j}{D_j\ C_j}} \qquad (4.8)$$

B(i) is the set of immediate components of i

These adjusted setup costs are used with single level lot sizing techniques. As the adjustment mechanism was developed for the multi-component single parent type of product structure, problems arise when it is applied to an item that has multiple parents as in a typical product structure in the MRP system.

Consider a case in which an item k has two parents, items i and j. Suppose an average demand for parent j, D_j, increases, other things being equal. It causes the average demand for its child k, D_k, to increase and the time between orders for the item, TBO_k, to shrink.

We note from equation (4.7) that the adjusted setup

cost for parent i, S_i', in turn, increases as TBO_k decreases. The inflated S_i' tends to increase the calculated lot size for the parent i, even though nothing has happened to the item itself. When the planned orders for one or more of the other parents of item k are changed, S_i' would change causing lot sizes for the parent i, even when no changes have occurred to parent i.

Consider a case in which item i is the only parent in the system with multiple components (e.g. k,l,m,...,etc.) just like the situation for which the setup cost adjustment mechanism has been developed. Suppose EPP ratios for the items in the system are all equal or even close to each other.

$$\frac{S_i}{C_i} = \frac{S_k}{C_k} = \frac{S_l}{C_l} = \frac{S_m}{C_m} = \ldots \text{ etc.}$$

where B(i) is set of (k, l, m, ...etc.).

In this case, as $D_i = D_k = D_l = D_m = \ldots$ etc., from the single parent explosion into multiple components, the ratios between the TBOs of parent i and one component item are equal to or closer to 1. This fact causes the adjusted setup cost for parent i, S_i', to be the sum of all the components' setup cost and its own. S_i' is most likely inflated so that only a single or a few number of large order(s) may be planned without checking whether this order plan is economical systemwise at all.

These problems with McLaren-Whybark's setup cost adjustment mechanism result from the way it was developed. It was originally generated from product structures in which every

item in the system has only one parent at most. Further, the adjustment mechanism was devised having examined the lot sizing patterns of an end item only. They have not checked their adjustment mechanism with the lot sizing patterns of component items. Thus, these problems imply that this technique is inappropriate for multi-parent, multi-component product structures which are usual in real MRP settings.

Blackburn and Millen (1982) proposed the modification based on the assumptions of an infinite horizon and constant demand per period. They further assumed that the lot size for an item is an integer multiple of the lot size for its parent, which is not valid for the product structure with common items having multiple parents.

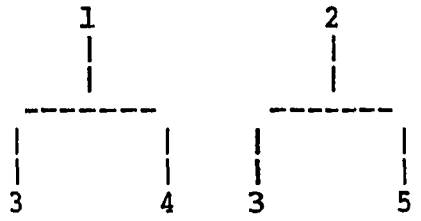
Instead of adjusting cost parameters involved in the problem, our proposed study explores ways to revise current schedules obtained from applying lot sizing rules to the system.

A common practice in the production of complex, assembled products is to subdivide the final products into assemblies, subassemblies, parts, and raw materials, maintaining parent-child relationship between adjacent two levels. These assemblies and subassemblies are produced on separate production orders determined by an MRP system.

To facilitate our understanding of the proposed heuristic's development, consider the following sample problem.

Numerical Example

Product Structure



Cost Structure

item cost	1	2	3	4	5
setup	100	100	100	100	100
carrying	2.5	2.5	1.0	1.0	1.0

Demand Pattern

item \ period	1	2	3	4	5	6
1	50	30	60	40	0	80
2	20	70	10	90	40	30

Given these input data, we can develop a set of MRP schedules applying a single level lot sizing rule sequentially to all the items residing in the system.

When our Modififed Economic Order Quantity model is chosen as a rule for the sequential application to the items in the system, the MRP schedules for the system would look as given below.

ITEM 1

period	1	2	3	4	5	6
GR	50	30	60	40	0	80
OHI	30					
POR	80		60	40		80

ITEM 2

GR	20	70	10	90	40	30
OHI		10			30	
POR	20	80		90	70	

ITEM 3

GR	100	80	60	130	70	80
OHI	80			70		
POR	180		60	200		80

ITEM 4

GR	80		60	40		80
OHI			40			
POR	80		100			80

ITEM 5

GR	20	80		90	70	
OHI	80			70		
POR	100			160		

Legend:

GR : Gross requirements

OHI: On hand inventory

POR: Planned order release

Carrying Cost = $ 515

Setup Cost = 1700

Total Cost = 2215

Examine the schedule for item 3 for a moment. In period 1, a planned order of 180 units of item is released and is scheduled to arrive in the same period (as we assume that lead time is zero). Out of these 180 units, 100 units are assembled together with item 4 and 5 into item 1 and 2 respectively, and 80 units are stored for later use in assemblying item 2 in period 2. The gross requirement of 180 units for item 3 is a combination of orders of 100 units from both items 1 and 2, and 80 units from single source, item 2. If we combine these two separate orders and assemble all 180 units into items 1 and 2 in period 1, 80 units of item 3 inventory and the same amount of item 5 inventory, will be moved up to the level of item 2 and stored as item 2 inventory in period 1. Furthermore, we release one combined order of 100 units for item 2 in period 1.

Then we could save $160 of inventory carrying cost ($ 80 for item 3 and $ 80 for item 5) and $ 100 of setup cost for item 2 while incurring $ 200 of new inventory carrying cost (80 units X $ 2.5 for item 2). Net savings from this combination of separate orders run $ 60. A series of similar decision making on combination of orders can be made in period 4 for item 3 and in period 3 for item 4.

After we finish revising the current MEOQ schedule, a new schedule is generated as follows,

ITEM 1

period	1	2	3	4	5	6
GR	50	30	60	40	0	80
OHI	30		40			
POR	80		100			80

ITEM 2

GR	20	70	10	90	40	30
OHI	80	10		70	30	
POR	100			160		

ITEM 3

GR	180		100	160		80
OHI	80					
POR	180		100	160		80

ITEM 4

GR	80		100			80
OHI						
POR	80		100			80

ITEM 5

GR	100			160		
OHI						
POR	100			160		

Legend:

GR : Gross requirements

OHI: On hand inventory

POR: Planned order release

Carrying Cost = $ 650

Setup Cost = 1400

Total Cost = 2050

Total inventory cost for the revised schedule is $ 2050, while total cost for MEOQ model is $ 2215. Net savings from this modification are $ 165. This modified schedule coincides with the ninth integer solution from mixed integer programming formulation which took more than 20 CPU seconds of execution time.

We may note that when an order is partially assembled and partially stored, making decisions on whether or not to combine this entire order offers a great opportunity for savings in total inventory cost. For a particular order, if we combine the entire order, we can save holding cost for the units which otherwise would be stored for later use in parent assemblies and the setup cost for its parent item while incurring holding cost for the units as its parents. Here the trade-off is between the setup cost for the parent and the echelon holding cost between the parent and child, and the ratio between the two costs is shown as follows,

$$\text{EPP-ML} = \frac{S_p}{C_p - C_i} \qquad (4.9)$$

where S_p : Setup cost for the parent

C_p : Carrying cost for the parent

C_i : Carrying cost for the child i.

Echelon holding cost is defined in Clark and Scarf (1960) as incremental costs that are added to the component inventory carrying costs, as the manufacturing process moves to successively higher levels. Thus, for any item, the inventory

carrying costs will be the sum of the carrying costs for its immediate component items and its echelon cost. This implies that the carrying costs for an item are at least equal to the sum of the carrying costs for its immediate components.

The EPP-ML ratio indicates the quantity of the inventory item which, if we process inventory stored on a component level to its immediate parent level and save one setup for the parent, would result in net savings equal to zero.

Therefore, if a schedule includes an inventory of which the quantity is less than EPP-ML ratio, we can save some inventory costs by making a decision on the combination of the orders. Positive net savings can be obtained in the range of quantity from one up to EPP-ML value for the items involved.

If savings from combination is greater than additional carrying cost incurred, we can combine separate subassembly orders and can save the difference. Implementing the logic of our recursive algorithm, combination of orders saves one or more setups for immediate parent(s) and eliminates stocking of child items while causing stocking of parent items instead, with additional echelon holding cost incurred. Besides the savings in inventory cost, combining assemblies and/or subassemblies orders will reduce or simplify paperwork as it reduces or eliminates the preparation, tracking and costing of production orders for those assemblies or subassemblies.

Our proposed recursion algorithm mainly consists of two phases, one in which a single item lot sizing rule sequentially schedules each stage and the other in which the current schedule

is revised following the logic described above. This process is executed stage by stage, and level by level. Detailed description of our heuristic is presented below.

Step 1. Scheduling

Schedule i th level items (i = 1,2,...,L) using a single item lot sizing rule chosen.

Step 2. Explosion

Explode these into the immediate lower level items (i.e., i+1 st level)

Step 3. Recursion

3.1 Decide the sequence, according to rules below, by which recursion (decision making for combination) procedures are executed in the level.

3.1.1 The most commonly used item, i.e., the item which has the most parents in the level has a higher priority.

Revising the schedule of the most commonly used item tends to have a larger impact on the schedules of the other items in the system.

3.1.2 The item whose requirement schedule contains more periods of positive requirement, which is less than the item's EPP, has a higher priority.

The more periods of positive requirement, which is less than the item's EPP, the schedule of a certain item has, the more periods the schedule is likely to carry items in inventory. Thus, there are more time periods in which we can make a decision to

combine orders.

3.1.3 On the occasion of tie, compare their EPP-ML ratio vectors. For each item, EPP-ML ratio vector is generated by listing EPP-ML ratios in decreasing order. For the purpose of comparison, we may need to enlarge vectors which contain less elements by filling with as many zeroes as needed.
The item whose vector is lexicographically greater than the other vectors has higher priority.
A bigger element (EPP-ML ratio) in the vector indicates that the range, in which net savings are realized from revising the current schedule, is rather wide and the chance to achieve positive net savings is higher in a wider range than in a narrower one. Furthermore, in the case of equal quantities in inventory, net savings are greater with a bigger EPP-ML ratio than with a smaller one.

3.1.4 In the case of a tie, the highest numbered item has higher priority. (This is an arbitrary rule.)

3.2 Lot sizing is performed on the k th item in the sequence decided in step 3.1. (k = 1,2,...,n, where n is number of items on the level.)

3.3 Make a set of decisions, according to steps below, on the schedule for the k th item whether to combine or not throughout planning horizon.

3.3.1 Search for a period, t, into which inventory is carried.

3.3.2 For the t th period, compute net savings that can be realized due to combining orders by examining the t th period schedules for its immediate parents and/or these parents' immediate components (i.e., items residing on the same level, i, as the k th item.

3.3.3 If net savings are positive, reschedule items involved accordingly.

If net savings are negative, go back to step 3.3.1.

3.4 Set k = k + 1.

Is k greater than n ?

No. Go to Step 3.5.

Yes. Set i = i + 1.

Is i greater than L ?

No. Return to Step 1.

Yes. Stop.

3.5 Explode the immediate parents (level i) into the k th item in the sequence and return to Step 3.2.

This recursion mechanism captures both the product structure, demand pattern and cost information contained in the multi-level problem. Furthermore, since our recursion algorithm can be used with all single level lot sizing rules, the logic and computational efficiency of current MRP systems can be maintained.

4.3 Experimental Investigation

Research Methodology

Assumptions to be made during this study are as follows:
(1) Since the component requirements are aggregated by time period for planning purposes, we assume that all of the requirements for each period must be available at the beginning period.
(2) All of the requirements for a given period must be met and cannot be backordered.
(3) The ordering decisions are assumed to occur at regular time intervals, namely, weekly.
(4) The orders which are placed at the beginning of a period, are assumed to be available in time to meet the requirements for that period. This assumption of zero production lead time is not very restrictive, however, since once the ordering decisions are made, they can be offset to allow for the production lead time.
(5) We assume that the components are withdrawn from inventory at once. Therefore the ending inventory level will be used in computing the inventory carrying cost.

First we formulated a mathematical model for the multi-level lot sizing problem. The model is in the form of mixed integer programming and will yield an optimal solution. Although it may produce an optimal solution, it may require huge amount of computational time as well as central memory. Therefore, use of the model in practice may be considered computationally infeasible, but provides a benchmark with which proposed heuristics can be compared.

In order to evaluate our proposed heuristic over as broad a set of sample problems as possible, the following major factors are varied: (1) product structure in terms of number of levels; (2) demand patterns (master schedule) for the finished products in the structure; and (3) degree of commonality (as defined in Collier (1981)) of the structure.

Lot Sizing Techniques

The first phase of the study will examine the cost performances between our proposed recursive algorithm operated together with single level lot sizing heuristics and when our recursive heuristic is not operated. The following five lot sizing techniques, MEOQ, MPOQ, MLTC, Silver-Meal, Wagner-Whitin, will be operated.

Product Structure (Number of Levels)

Product structure refers to the hierarchical processing pattern of parts and components from raw materials to the finished products. In the experiment, we create and use several product structures which are different mainly in terms of number of levels in the structure. This study will cover problems ranging from three to four levels (see appendix I.)

Demand Pattern (Master Schedule)

One of the problem characteristics is the demand pattern, that is, the master schedule for the finished products, which

that is, the master schedule for the finished products, which gives the scheduled production for each period. The variability of master schedule demand is measured by the coefficient of variation (C_v) which is defined as demand standard deviation divided by mean demand. Previous studies have adopted different sets with different values for C_v. Hoo Gon Choi et al. (1984) used several values of C_v ranging from 0 (constant demand) to 0.61. Wemmerlov (1982) and Biggs et al. (1976) set the theoretical C_v values to 0.15, 0.58, and 1.14 in order to represent three different demand patterns. McLaren (1977) chose five values for master schedule variability with C_v values of 0.3, 0.7, 1.0, 2.0, 3.0. The latter two high values represent extreme variations or "lumpiness" in the master schedule in that their expected mean time between demands are 3.75 and 7.50, respectively. The theoretical C_v values for this study will be set to 0.3, and 1.0 in order to represent two different, but relatively realistic demand patterns.

Degree of Commonality

It is common practice that an item (assembly, subassembly, parts, raw material) is used in differnt places to be assembled into higher level items. Common usage of an item by several parent items, complicates the explosion process (the computation of requirements for the lower level items) in MRP systems. Common usage of items in a manufacturing environment implies definitely a reduction in the number of items in the inventory system, and possibly a reduction in the total investment in

inventory.

A measure used for different levels of common usage is the degree of commonality as defined by Collier.

$$DC = \frac{\sum_{j=i+1}^{d+i} P_j}{d} \qquad (4.8)$$

where P_j = the number of immediate parents that component j has over a set of end items or product structure levels

d = the number of distinctive components in the set of end items or product structure levels

i = the number of end items or the number of highest level parent items for the product structure levels

E = the total number of immediate parents for all distinct component parts over a set of end items or product structure levels

The lower bound on degree of commonality is one, which occurs in the case where there is no common item used in the whole system. The upper bound, on the other hand, is E, which occurs when only one component is used for the production of all end products in the system.

$$E = \sum_{j=i+1}^{d+i} P_j$$

DC reflects the average number of common parent items per distinct component part and characterizes different

structures of products. Collier classified DC of one as zero commonality, DC of 1.5 as low, DC of 2.0 as medium, and finally DC of 2.5 as high commonality. Thus, several different degrees of commonality, ranging from zero to high commonality (1.0 (= zero commonality), 1.5, 2.0, and 2.5) will be examined in the experiment.

Cost Parameter

This study uses the concept of echelon stock introduced by Clark and Scarf (1960), where the echelon stock for an item is defined as the total inventory in stock for the item, regardless of its location, that is, whether it exists as itself and is identified as the particular item, or as a part assembled into higher level assemblies. Corresponding to this concept, the echelon holding costs are defined as incremental costs that are added to the component inventory carrying costs, as the manufacturing process moves to successively higher levels. Thus, for any item, the inventory holding costs are the sum of the holding costs for its immediate component items and its echelon holding costs.

In generating the test problems, we set C (carrying cost per unit per period) = .3 for all raw materials; for the higher level items, we set $C_i = e_i + \sum_{j \in B(i)} C_j$ where B(i) is the set of immediate components of i and e_i, the echelon holding cost for item i, is selected from a uniform distribution with values e_i= 0.1, 0.2, 0.3 for the next higher level, e_i= 0.2, 0.3, 0.4 for the next higher level, and e_i= 0.2, 0.4, 0.6 for the highest

level, respectively. We computed S_i (setup cost) from $S_i = EPP_i * C_i$ with EPP_i selected from a uniform distribution with values EPP_i= 50, 70, 90, 120.

The major objectives of the experiment are to evaluate the solution quality and computational efficiency of the proposed lot sizing heuristic, i.e., the recursion algorithm.

Solution quality for the comparison of treatments, i.e., our proposed recursion algorithm and McLaren-Whybark setup cost adjustment algorithm, is measured in terms of the percent deviation from control treatment, that is, original single level rule solution cost. Each lot sizing rule under a treatment, different from the control treatment, is compared to its counterpart under the control treatment.

$$\frac{\text{Heuristic solution cost under treatment} - \text{Control group heuristic solution cost}}{\text{Control group heuristic solution cost}} \times 100$$

The five measures of solution quality from the five lot sizing rules with each treatment are averaged out and the average represents the overall solution quality for a treatment group for a test problem.

Solution quality for the comparison of heuristic rules is measured in terms of the percent deviation from a certain base solution cost, i.e., basic single level W-W solution.

$$\frac{\text{Heuristic solution cost} - \text{Basic W-W solution cost}}{\text{Basic W-W solution cost}} \times 100$$

Computational efficiency for the comparison of treatments is measured in terms of the percent deviation from the base solution time (control treatment). Each lot sizing rule with a treatment, different from the control treatment, is compared to its counterpart with the control treatment.

$$\frac{\text{Heuristic solution time under treatment} - \text{Control group heuristic solution time}}{\text{Control group heuristic solution time}} \times 100$$

The five measures of solution efficiency from the five lot sizing rules are averaged out and the average represents overall solution efficiency for a treatment group for a test problem.

Computing efficiency for the comparison of lot sizing rules is measured in terms of average elapsed computing time for a sample problem.

The vehicle for the experiment is the computer simulation. This simulation program is designed to duplicate the major functions of an MRP system, and can allow us to systematically vary the experimental factors (see Appendix II).

The MRP simulation model is designed to perform similar major functions of an MRP system, such as input of cost data, the bills of materials, generation of master schedule, exploding process, lot sizing, and cost accounting. Since our primary concern, however, is lot sizing, the elapsed computing time measure is for the lot sizing and recursion function only.

The simulation model developed here can be classified as a terminating simulation (even though the real system is

non-terminating), since the lot sizing will stop at the end of the planning horizon. Two replications of 52 period demand pattern will be made for each C_v level. Furthermore, to provide identical conditions in terms of demands under which all lot sizing rules operate for each combination of factor levels, we will use common random number (CRN) strings. This approach is probably the most widely used variance reduction technique in practice, due to its simplicity and intuitive appeal.

Since we are concerned with the relative overall solution quality and computational efficiency among the three treatments, i.e., recursion algorithm, McLaren-Whybark algorithm (M-W) and original single level rule (control group), two primary hypotheses are made which may be tested statistically.

The first hypothesis would test whether there are no differences among the three treatments in terms of solution quality, i.e., cost performance. The second one would test the differences among the treatments in terms of computational efficiency. Each treatment is applied together with the same five different single level lot sizing rules, namely MEOQ, MPOQ, MLTC, S-M, and W-W. The average of the five rules' performance represents the overall performance of each treatment for one test problem.

Also, as we are concerned with the relative overall solution quality and computational efficiency among the fifteen heuristic procedures that are combination of the three treatments and the five single level lot sizing rules chosen for this study, two primary hypotheses are made which may be tested statistically.

They would examine whether there are any differences among the fifteen lot sizing techniques in terms of cost performance and computing time, respectively. When compared in terms of cost performance, each rule's performance is measured in terms of the percent deviation from the basic W-W solution cost. On the other hand, when compared in terms of computational efficiency, each rule's performance is measured in terms of elapsed computing time for a test problem.

To examine the above four hypotheses, multiple comparison tests are used to rank the cost performances or computational efforts required of the treatments and the lot sizing heuristics. They will also group treatments or rules for which there is no significant differences in cost performances or computational efficiency.

Another area for testing involves the factor effects on cost performance for a given treatment. Three specific hypotheses are made which may be tested statistically. The three hypotheses test whether the number of levels in a product structure, the degree of commonality in a product structure, and the master schedule C_v have any effects on the treatment in terms of cost performance.

The computational results and analysis of the formal simulation experiment are presented in the next chapter.

Chapter 5

Experimental Results

Chapter 4 introduced a methodlogy for evaluating the solution quality and computational efficiency of heuristic multi-level lot sizing techniques. The vehicle for this experimental comparison study is a computerized MRP simulation model. The model was designed to perform the major functions of a typical MRP system, and is flexible with respect to the input of various product cost data, product structure configuration, and master schedule demands, which determine a test problem. Therefore, lot sizing heuristics can be evaluated across a wide variety of sample problems.

This chapter presents the results of the computer simulation experiment. The procedure used for analysis is described in the first section. The second section presents the test results of proposed hypotheses. Finally, conclusions from the research experiment are presented in the last section.

5.1 Analysis Methodology

Our experimental design is full factorial design with the four problem factors, i.e., product structure levels, degrees of commonality, master schedule C_v, and treatments (single level rules equipped with recursion algorithm, original single level rules, single level rules equipped with McLaren-Whybark setup cost adjustment mechanism) taking on two, four, two, and three

levels, respectively. Each combination of factor levels has two replications. Futhermore, one sample problem is solved by five single level lot sizing procedures (MEOQ, MPOQ, MLTC, S-M, W-W). Thus, there are a total of 2 x 4 x 2 x 3 x 2 x 5 = 480 observations.

Since our major concern for this study is solution quality and computational efficiency, two criteria were used to measure the performance of the treatments and lot sizing rules: (1) total inventory costs; and (2) the computational time requirements. However, since the number of items varied for different problems, ranging from a low of 7 distinctive items to 30 items, the resulting total inventory costs also varied from $ 145,036 to $ 656,456. Hence, in order to give an equal weight to each observation, when comparing the treatments and the lot sizing techiniques, the total inventory costs had to be normalized. When comparing the treatments, each basic single level rule under control treatment is used as the basis for comparing its counterparts under other treatments. When comparing the lot sizing heuristics, the single level Wagner-Whitin algorithm (WW) is used as the basis for comparing the lot sizing techniques for one test problem.

The other criterion, measured in CPU seconds, is the computing time required for determining the order schedules for all items in the system. However, the computational time varies drastically with different lot sizing rules, especially between W-W and other heuristics, ranging from a low of .003 seconds with simple rules such as basic MEOQ, MLTC to .227 seconds with W-W

equipped with recursion algorithm for the same sample problem. Thus, when comparing the treatments, the computing time has to be normalized. Computing time for each rule under control treatment is used as the basis for comparing the solution efficiency of its counterparts under different treatments. When comparing the individual lot sizing techniques, the elapsed computing time in CPU seconds for the techniques are used.

The experimental hypotheses of Chapter 4 examine the overall and factor level effects on the cost performances and computational efficiency, and are summarized in Fig. 5.1. Since the replication runs varying the demands in the master schedules introduce a random effect to the experiment, the experimental hypotheses may be statistically tested.

The use of percentage value as a measure for cost performance enables us to use standard parametric ANOVA tests on the cost performance data, as its values are normally distributed and have generally equal variances.[1] However, computational efficiency measures, regardless of whether they are presented in absolute or relative terms, are genarally not normally distributed and are bimodal. Subsequent transformation of efficiency data could not correct the situation. Therefore, the nonparametric procedure suggested by Taylor (McLaren (1977)) is used for the test on efficiency data.

To examine the hypotheses H_1 and H_3, Tukey's studentized range test, which is multiple group means comparison test

---

[1] The SAS statistical package was used for applying ANOVA procedures.

H_1 : There is no difference among the three treatments;
treatment 1: our proposed recursion algorithm;
treatment 2: McLaren-Whybark's setup cost adjustment;
treatment 3: the basic single level rules without recursion algorithm;
in terms of solution quality (cost performance).

H_2 : There is no difference among the three treatments;
treatment 1: our proposed recursion algorithm;
treatment 2: McLaren-Whybark's setup cost adjustment;
treatment 3: the basic single level rules without recursion algorithm;
in terms of computational efficiency (computing time).

H_3 : There is no difference between the 15 lot sizing heuristics;
treatment 1 rules: 5 lot sizing rules (MEOQ, MPOQ, MLTC, S-M, WW) equipped with our proposed recursion algorithm;
treatment 2 rules: the same 5 lot sizing rules with McLaren-Whybark's setup cost adjustment mechanism;
treatment 3 rules: the same 5 lot sizing rules without recursion algorithm;
in terms of solution quality (cost performance).

H_4 : There is no difference between the 15 lot sizing heuristics;
treatment 1 rules: 5 lot sizing rules (MEOQ, MPOQ, MLTC, S-M, WW) equipped with our proposed recursion algorithm;
treatment 2 rules: the same 5 lot sizing rules with McLaren-Whybark's setup cost adjustment mechanism;
treatment 3 rules: the same 5 lot sizing rules without recursion algorithm;
in terms of computational efficiency (computing time).

H_5 : Number of levels in the product structure has no effect on the treatment in terms of cost performance.

H_6 : Degrees of commonality in the product structure have no effect on the treatment in terms of cost performance.

H_7 : Master schedule variability (C_v) has no effect on the treatment in terms of cost performance.

Figure 5.1 Experimental Hypotheses

provided as an option in ANOVA procedure, is used to rank the cost performances of the treatment and the lot sizing heuristics and to group those rules for which there is no significant difference in cost performances. To examine the hypotheses H_2, and H_4, a posteriori contrast multiple comparison test -- devised for a data set which does not fit for ANOVA tests -- is used to rank the computational efforts required of the treatments and the lot sizing heuristics and to group those rules for which there is no significant difference in computational efficiency. To examine the hypotheses H_5, H_6, and H_7, standard parametric ANOVA tests are used.

5.2 Overall Results and Testing of Experimental Hypotheses

For each of the three treatments, there are solution quality and computational efficiency measures for each test problem. Table 5.1 and 5.2 show the overall and marginal cost performance means and computing time means for the treatments, respectively. Marginal means are obtained for each factor level by holding that factor level constant and calculating the mean cost performance for all other observations. Detailed analyses of these two tables are presented in subsequent sections.

H_1 Test

Table 5.3 highlights the ranked overall solution quality for all three treatments from Table 5.1. Due to the usage of percentage value, the sample means are normally distributed, and

Table 5.1

OVERALL AND MARGINAL MEAN COST PERFORMANCE
TREATMENT COMPARISON

		Treatment 1		Treatment 2		Treatment 3 (Control Group)	
	n	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Overall	32	-7.612	4.843	-3.808	4.606	0	0
Product Structure							
3levels	16	-6.366	4.232	-1.792	5.713	0	0
4levels	16	-8.858	5.220	-5.826	2.899	0	0
Degree of Commonality							
Zero	8	-3.778	1.303	-7.195	1.653	0	0
Low	8	-7.296	4.407	-6.130	3.267	0	0
Medium	8	-7.913	4.560	-1.150	2.070	0	0
High	8	-11.460	5.293	-0.758	6.190	0	0
Master Schedule C_v							
0.3	16	-10.335	4.756	-5.470	4.446	0	0
1.0	16	-4.888	3.166	-2.147	4.266	0	0

Treatment 1 : Recursion Algorithm

Treatment 2 : McLaren-Whybark Setup Cost Adjustment

Treatment 3 : Basic Single Level Rule Only

n = number of observations

Table 5.2

OVERALL AND MARGINAL MEAN COMPUTATIONAL EFFICIENCY
TREATMENT COMPARISON

		Treatment 1		Treatment 2		Treatment 3 (control group)	
	n	Mean	Standard Deviation	Mean	Standard Deviation	Mean	Standard Deviation
Overall	32	165.43	40.47	-3.83	10.62	0	0
Product Structure							
3 Level	16	152.17	39.48	-4.55	11.92	0	0
4 Level	16	178.69	38.10	-3.10	9.48	0	0
Degree of Commonality							
Zero	8	127.25	23.76	4.33	8.70	0	0
Low	8	163.51	35.66	-6.71	10.14	0	0
Medium	8	182.70	42.06	-6.82	5.56	0	0
High	8	188.27	32.28	-6.11	13.71	0	0
Master Schedule C_v							
0.3	16	181.56	35.99	-3.36	10.03	0	0
1.0	16	149.30	39.17	-4.30	11.49	0	0

Treatment 1 : Recursion Algorithm

Treatment 2 : McLaren-Whybark Setup Cost Adjustment

Treatment 3 : Basic Single Level Rule Only

n = number of observations

fit an ANOVA model well. Thus, Tukey's studentized range test, which is provided as an option for mean comparison test in the ANOVA procedure, is used to rank the treatments in terms of cost performance. This test controls the type I experimentwise error rate, instead of controlling the type I comparisonwise error rate like the t test does. It should be noted that there is no significant difference in means between any two groups belonging to the same subset at some prescribed significance level, α.

Table 5.3

TUKEY'S STUDENTIZED RANGE TEST
FOR COST PERFORMANCE
TREATMENT COMPARISON

Subset	Treatments	Mean	Standard Deviation	Minimum Value	Maximum Value
1	Treatment 1	-7.612	4.843	-18.19	-2.01
2	Treatment 2	-3.808	4.606	-10.32	7.66
3	Treatment 3	0.0	0.0	0.0	0.0

Table 5.3 also shows the results of the Tukey's studentized range test for the treatments for $\alpha = 0.05$. We note that H_1 must be rejected. There is a significant difference among the three treatments' cost performance at the $\alpha = 0.05$ level. These values are based on an average of cost performances of five lot sizing rules for one sample problem. Actual t statistics and p-values obtained from a set of pairwise t test is shown in Table 5.4.

Table 5.4

T STATISTICS AND P-VALUES

Treatment Comparison	t statistic	p-value
2 vs. 1	6.505	< 0.001
3 vs. 2	6.514	< 0.001

Generally speaking, Treatment 1 appears to outperform Treatment 3 most of the times; the maximum value of treatment 1 is still negative, while the maximum value of Treatment 2 is well above 0.

H_2 Test

Table 5.5 highlights the ranked overall solution efficiency for all three treatments from Table 5.2. The data set of solution efficiency has bimodal distribution and could not fit an ANOVA model even with transformation. Thus, Tukey's studentized range test cannot be used to rank the treatments in terms of computing efficiency. Instead, Taylor's multiple comparison test, which was designed to perform comparison tests on data with unequal group variances, may be used. The Taylor procedure is based on the concept of multiple confidence intervals for ranked group means, using a significance level, α', modified to provide an experimentwise error rate α. (1 - α') confidence intervals are constructed around each group mean. All those groups having overlaps in confidence intervals are classified into a subset.

The interpretation of a subset is that the group means within the subset do not differ by more than a prescribed level of confidence, and that the group means between subsets do differ by more than the prescribed level. The level α' is obtained as α / k, where k is the number of the group means and $(1 - \alpha)$ is the confidence level for the entire experiment.

Table 5.5

TUKEY'S STUDENTIZED RANGE TEST
FOR COMPUTING TIME
TREATMENT COMPARISON

Subset	Treatments	Mean	Standard Deviation	Minimum Value	Maximum Value
1	Treatment 2	-3.8	10.62	-22.80	15.30
	Treatment 3	0.0	0.0	0.0	0.0
2	Treatment 1	165.4	40.47	87.20	235.10

We see that H_2 must also be rejected; there is a significant difference in relative computational efficiency between Treatments 2, 3 and Treatment 1. Between Treatments 2 and 3, the difference in efficiency is not statistically different at $\alpha=0.05$ level. Generally speaking, the recursion algorithm seems to take more than twice as much as the original rule or setup cost adjustment mechanism. Table 5.6 shows t statistics and corresponding p-values for a set of pairwise t test.

Table 5.6

T STATISTICS AND P-VALUES

Treatment Comparison	t statistic	p-value
2 vs. 1	34.160	< 0.001
3 vs. 1	33.387	< 0.001
3 vs. 2	0.773	< 0.5

H_3 Test

Table 5.7 shows the ranked overall solution quality for all fifteen lot sizing rules, which is a combination of five rules and three treatments. Due to the usage of percentage value, the sample means are generally normally distributed, and fit for the use of an ANOVA model. Thus, Tukey's studentized range test is used to rank the lot sizing rules in terms of cost performance.

Table 5.8 shows the results of the Tukey's studentized range test for the fifteen lot sizing rules for $\alpha = 0.05$. The basic W-W rule was used as basis here in this comparison. We find that H_3 must be rejected; there are significant differences among the cost perfomances of the lot sizing rules at the $\alpha = 0.05$ level.

All the rules equipped with the recursion algorithm rank high. All the rules but MPOQ equipped with McLaren-Whybark setup cost adjustment mechanism take middle ranks, while all the simple rules generally take bottom ranks. McLaren (1977) reported that W-W rule with setup cost adjustment mechanism dominated all other

Table 5.7

COST PERFORMANCE CONTRAST

Rank	Rules[2]	COST PERFORMANCE (%) Mean	Standard Deviation	Minimum Value	Maximum Value
1	R/MLTC	-7.83	7.15	-23.82	0.76
2	R/MEOQ	-7.79	6.85	-25.52	1.22
3	R/W-W	-5.84	4.07	-16.69	0.20
4	M/W-W	-5.83	4.01	-11.76	5.30
5	R/MPOQ	-4.65	7.31	-23.85	6.76
6	R/S-M	-3.94	5.35	-15.32	2.91
7	M/S-M	-3.46	2.56	-8.82	0.61
8	M/MLTC	-2.75	5.48	-10.29	10.12
9	M/MEOQ	-1.35	5.95	-10.86	11.81
10	B/W-W	0.0	0.0	0.0	0.0
11	B/MLTC	1.33	1.99	-1.65	8.90
12	B/MPOQ	2.37	2.50	-0.89	7.80
13	B/S-M	2.58	2.20	-0.30	9.44
14	M/MPOQ	2.91	8.72	-10.37	24.99
15	B/MEOQ	3.03	1.82	-0.63	9.84

[2] Prefix R, M, and B represents the three different treatments, i.e., recursion algorithm, McLaren-Whybark's setup cost adjustment, and basic rule, respectively.

combination rules examined in his study and performed quite well relative to his multi-level dynamic programming model. The multi-level dynamic programming model does not guarantee optimal solution but in his earlier (1975) study allegedly produced the same solutions as did the optimal MIP algorithm for the 32 six- and eight-period planning horizon sample problems.

H_4 Test

Table 5.9 shows the ranked overall solution efficiency for all fifteen lot sizing rules. The data set of solution efficiency for lot sizing rules shows a skewed, bimodal distribution and could not fit an ANOVA model even with tranformation. Thus, Taylor's multiple comparison test may be used.

Table 5.10 shows the results of the Taylor multiple comparison test for the fifteen lot sizing rules at $\alpha = 0.05$ level. We see that H_4 should be rejected; there are significant differences in computational time among the lot sizing rules at $\alpha = 0.05$. All the Treatment 2 and Treatment 3 rules except W-W algorithm are in the first subset; they are not significantly different in terms of computing time. All the Treament 1 rules except W-W constitute the second subset. The third subset is composed of W-W algorithms across all the treatments. There are a few rules that outperform W-W with setup cost adjustment consuming less time, i.e., R/MLTC and R/MEOQ.

Table 5.9

COMPUTING TIME PERFORMANCE

Rank	Rules[3]	TIME PERFORMANCE (CPU sec.) Mean	Standard Deviation	Confidence Intervals $\alpha = 0.05$	
1	B/MLTC	0.0073	0.0034	0.0056	0.0091
2	M/LTC	0.0075	0.0037	0.0055	0.0094
3	B/S-M	0.0085	0.0041	0.0063	0.0107
4	M/MPOQ	0.0087	0.0047	0.0062	0.0112
5	M/S-M	0.0088	0.0046	0.0063	0.0112
6	M/MEOQ	0.0094	0.0046	0.0070	0.0118
7	B/MPOQ	0.0105	0.0044	0.0082	0.0128
8	B/MEOQ	0.0112	0.0055	0.0083	0.0140
9	R/S-M	0.0246	0.0115	0.0185	0.0306
10	R/MLTC	0.0255	0.0121	0.0191	0.0319
11	R/MEOQ	0.0284	0.0119	0.0222	0.0347
12	R/S-M	0.0298	0.0137	0.0226	0.0371
13	M/W-W	0.3840	0.1848	0.2885	0.4820
14	B/W-W	0.3853	0.1833	0.3068	0.5003
15	R/W-W	0.4035	0.1920	0.3022	0.5048

[3] Prefix R, M, and B represents the three different treatments, i.e., recursion algorithm, McLaren-Whybark's setup cost adjustment, and basic rule, respectively.

Table 5.10

TAYLOR MULTIPLE COMPARISON TEST
FOR COMPUTING TIME
($\alpha = 0.05$)

Subset[4]	Lot Sizing Rules[5]
1	B/MLTC M/MLTC M/POQ B/S-M M/S-M M/MEOQ B/MPOQ B/MEOQ
2	R/MEOQ R/MPOQ R/MLTC R/S-M
3	M/W-W B/W-W R/W-W

[4] The difference between any two group means within a subset is not statistically significant at the $\alpha = 0.05$ level.

[5] Prefix R, M, and B represents the three different treatments, i.e., recursion algorithm, McLaren-Whybark's setup cost adjustment, and basic rule, respectively.

H_5, H_6, H_7 Tests

Since, we set the cost performance of Treatment 3 as the base, the measure is always zero. Thus, the problem factors have no effect on this measure so that treatment 3 is excluded from this analysis. Table 5.11 summarizes the test results for the three effects of problem factor on cost performance hypotheses. An X in a cell indicates that the hypothesis for the factor and treatment is rejected at the 0.05 level. Rejection of the null hypothesis implies that the cost performance of the treatment varies significantly for different values of the problem factor.

Table 5.12 shows marginal mean cost performance for each factor level when compared treatmentwise.

Table 5.11

ANOVA RESULTS
FACTOR EFFECTS ON COST PERFORMANCE

	TREATMENT 1	TREATMENT 2
1-Way ANOVA		
H_5: Number of levels	X	X
H_6: Degree of Commonality	X	X
H_7: Master Schedule C_v	X	X
2-Way ANOVA		
No. of Levels-Commonality		X

Table 5.12

MARGINAL MEAN COST PERFORMANCE
TREATMENT COMPARISON

	Treatment 1		Treatment 2	
	Mean	Standard Deviation	Mean	Standard Deviation
Product Structure				
3 Level	-6.366	4.232	-1.792	5.713
4 Level	-8.858	5.220	-5.826	2.899
Degree of Commonality				
Zero	-3.778	1.303	-7.195	1.653
Low	-7.296	4.407	-6.130	3.267
Medium	-7.913	4.560	-1.150	2.070
High	-11.460	5.293	-0.758	6.190
Master Schedule C_v				
0.3	-10.335	4.756	-5.470	4.446
1.0	-4.888	3.166	-2.147	4.266

	Treatment 1		Treatment 2	
	3-level Mean	4-level Mean	3-level Mean	4-level Mean
Degree of Commonality				
Zero	-3.800	-3.755	-6.662	-7.727
Low	-6.050	-8.543	-5.238	-7.023
Medium	-6.633	-9.193	0.513	-2.813
High	-8.980	-13.94	4.225	-5.743

Treatment 1 : Recursion Algorithm
Treatment 2 : McLaren-Whybark Setup Cost Adjustment
Treatment 3 : Basic Single Level Rule Only

These results in the Table 5.12 can be diagrammed as in Figures 5.2, 5.3, 5.4, and 5.5. Fig. 5.2 shows the effects of number of levels on the treatment's cost performance. Both Treatments 1 and 2 move in the same direction and they appear to perform better on the product structure with 4 levels than with 3 levels. The mean difference between the two treatments are significant at both levels at 0.05 level.

Fig. 5.3 reveals the effects of degrees of commonality on the treatment's cost performance. Treatments 1 and 2 take opposite directions. Treatment 1 performed better with product structures with a low to high degree of commonality. On the other hand, Treatment 2 performed better with product structures with zero degree of commonality. The mean differences in cost performance between the treatments are significant at all levels but one at low degree of commonality.

Fig 5.4 shows the effects of master schedule variability on the treatment's performance. Both treatments move in the same direction; they perform better on rather low demand C_v. Treatment 1 is performing better than Treatment 2 at both C_v levels and the mean difference between the two treatments are significant across the levels.

Fig. 5.5 depicts the interaction effects of number of levels and degree of commonality on the treatment's cost performance. Generally speaking, both treatments obtain better results on the 4-level structure than on the 3-level one, and the margin gets wider as they move on the high end of degree of commonality. It should also be noted that Treatment 1 generally performs better

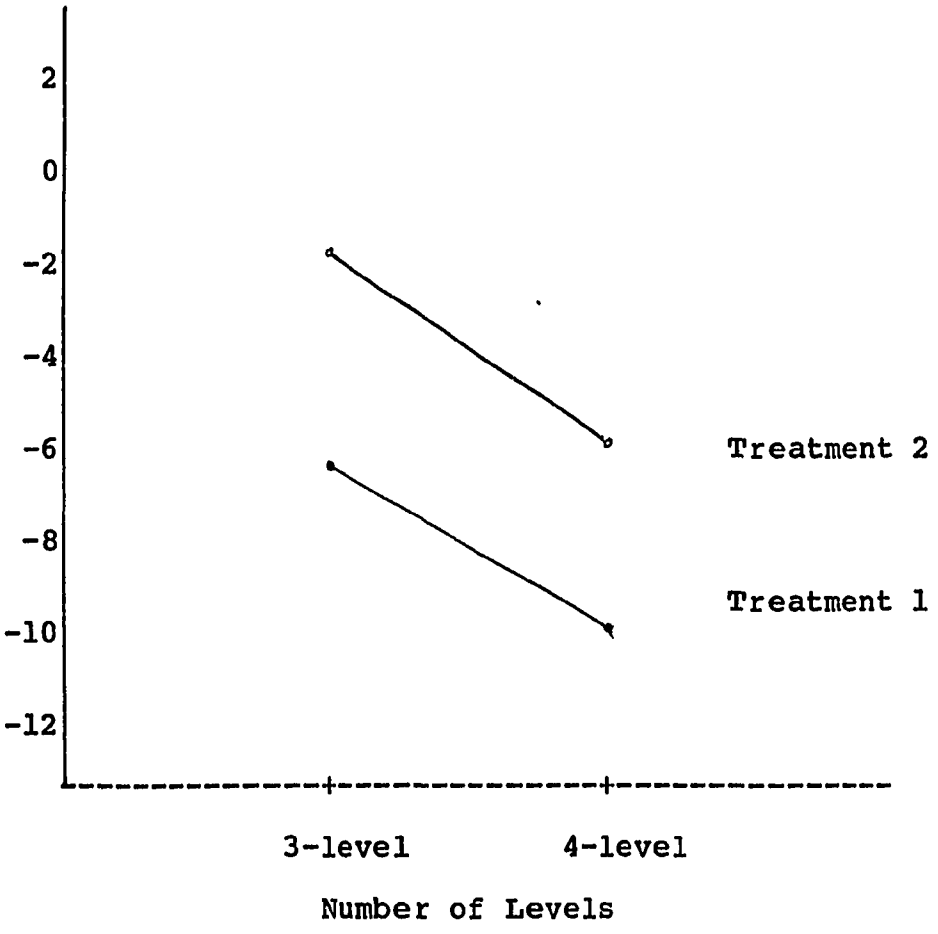


Fig. 5.2 Effects of Number of Level in Product Structure on the Treatment's Cost Performance 2

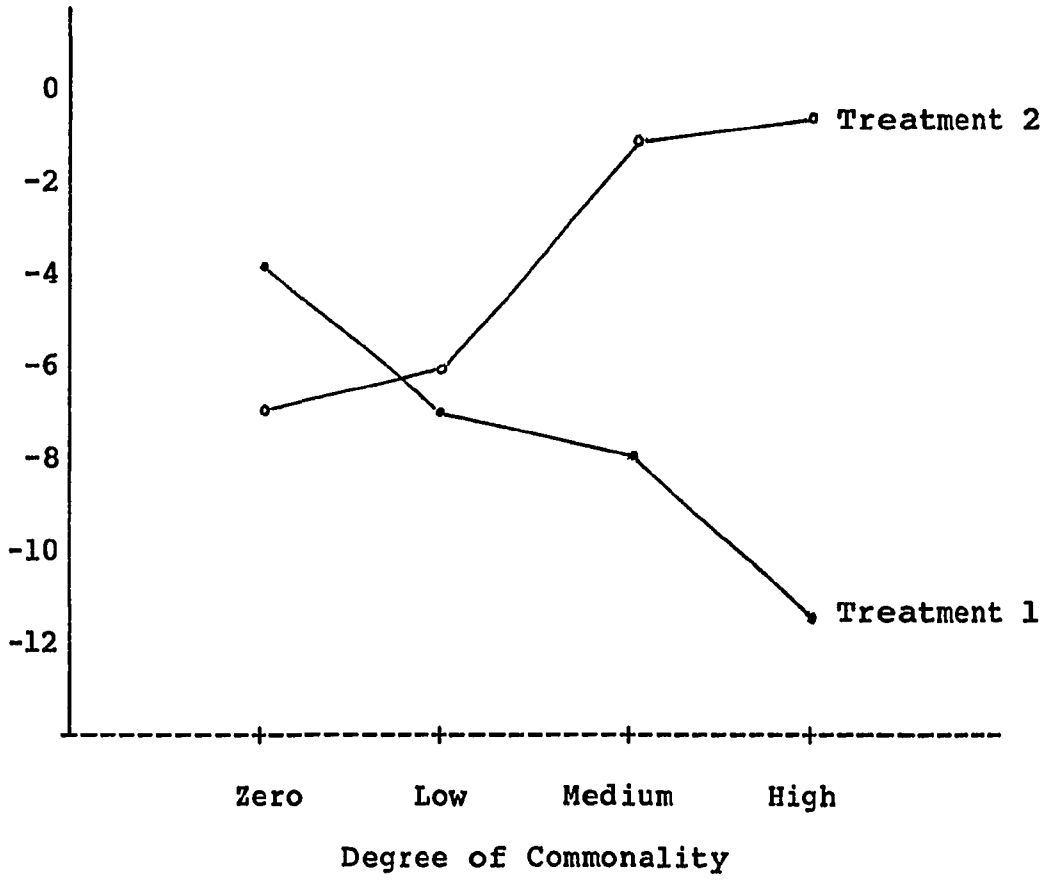


Fig. 5.3 Effects of Degree of Commonnality on the Treatment's Cost Performance

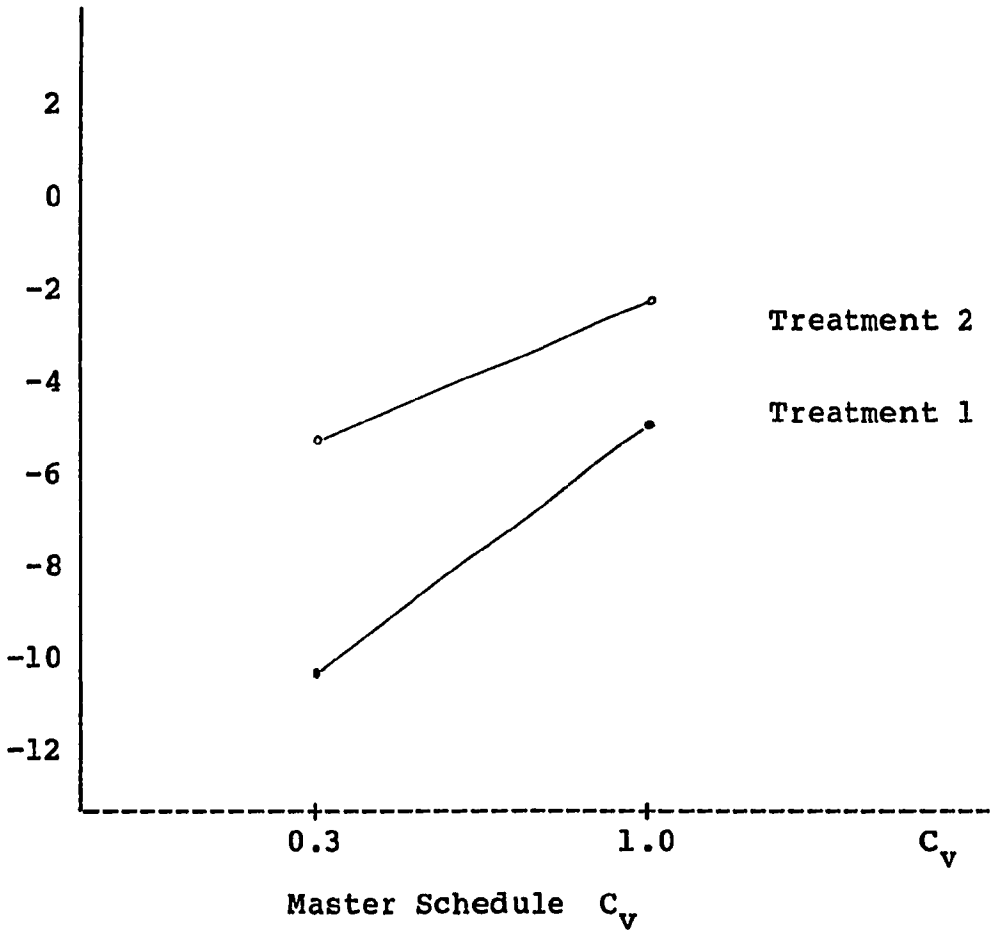


Fig. 5.4 Effects of Master schedule C_v on the Treatment's Cost Performance

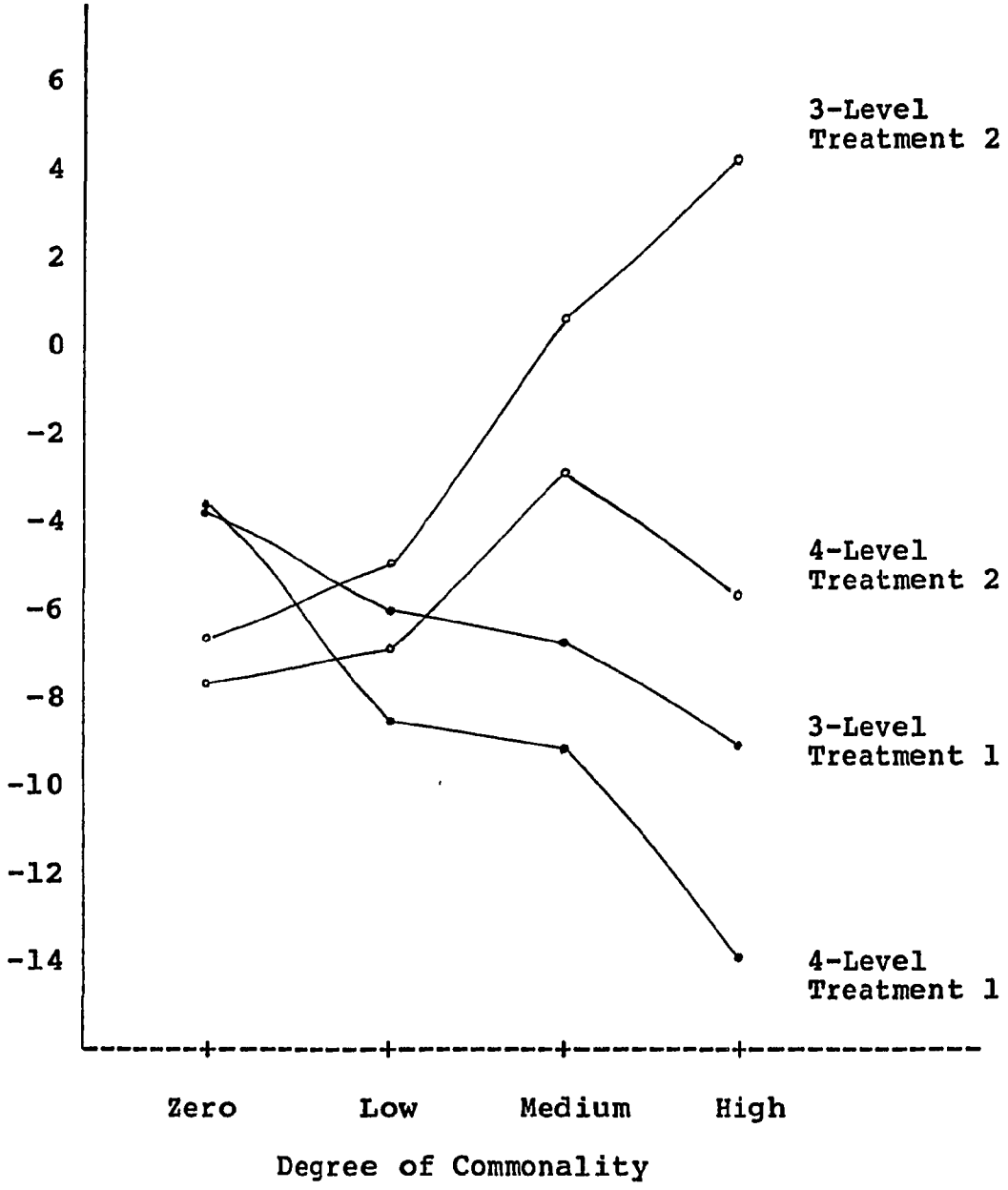


Fig. 5.5 Combined Effects of Number of Levels and Degree of Commonality on the Treatment's Cost Performance

as it +oves on the high end of degree of commonality, while Treatment 2 generally performs worse as it moves on the high end of degree of commonality except one case, i.e., 4-level structure with high degree of commonality.

5.3 Conclusions

In this chapter we have examined the major experimental hypotheses using a formal, statistically designed simulation experiment.

We have seen that Treatment 1, i.e., the recursion algorithm, provides lower cost solutions, on average, than the McLaren-Whybark setup cost adjustment mechanism and the sequential application of basic single level lot sizing rules. However, the lack of optimal solutions for the test problems prohibited us from describing the general solution quality of the rules equipped with the recursion algorithm. Two heuristics (MLTC and MEOQ) with recursion procedure significantly outperform the W-W with setup cost adjustment which was used as the basis in McLaren (1977) because of its superior performance in cost savings. Furthermore, they require less computing time.

However, the computing times of the rules with this recursion algorithm are generally two to three times larger than the computing times for their counterparts under control treatment or the setup cost adjustment treatment with the exception of W-W algorithm. In the case of W-W, the recursion algorithm is only 4 to 5% larger than its counterpart under two other treatments, namely W-W with McLaren-Whybark adjustment and basic W-W.

Considering the two criteria, i.e., cost and time, for a small penalty in computational requirements, compared to control treatment and McLaren-Whybark setup cost adjustment treatment, the four lot sizing heuristics (MLTC, MEOQ, MPOQ, S-M) with recursion procedure present solutions that offer significant cost savings. They outperform or compare very favorably with W-W with setup cost adjustment solution, but have much lower computational requirements.

Another point of significance was that as the degree of commonality increased, our recursion algorithm yielded solutions that were increasingly superior to those obtained with McLaren-Whybark methods or basic rules. The same is true with the factor of number of levels in a product structure. The more levels a product structure has, the better the solutions produced by the Treatments 1 and 2, with Treatment 1 outperforming Treatment 2.

For the effect of demand variability (C_v) on performance, it was found that the master schedule with lower variability in terms of coefficient of variation resulted in lower costs across the treatments. For rather highly lumpy demand (higher C_v) pattern, both treatments could not perform as well as they did in the low end of C_v. A master schedule having large expected TBO (time between order) value may not offer as many opportunities or occasions for Treatment 1 to make improvements on the order schedule obtained by the original rules. This judgement is supported by the fact that Treatment 1 required less computational time for the higher C_v demand pattern. Percentage increase value reduced from 181.6 to 149.3 (see Table 5.2).

The next chapter presents brief conclusions and contributions from the study and suggests future extensions for further research.

Chapter 6

Conclusions, Contributions, and Extensions

The primary objective of this study was the development and evaluation of simple lot sizing techniques that can be applied effectively and efficiently to existing MRP systems dealing with multi-level product structure. Most current MRP system users are choosing simple single level rules (EOQ, POQ, LTC etc.,) which do not utilize the vertical dependency and horizontal commonality relationships, let alone useful and valuable informations such as other relevant items' order schedules made available by MRP.

This chapter summarizes the research and presents the conclusions from the study. Contributions of this study are cited. This chapter concludes with suggested directions for further research.

6.1 Conclusions

The three popular lot sizing rules (EOQ, POQ, LTC) were modified for the single level, and multi-level problem as well. The three modified lot sizing rules operate on the part period accumulation principle besides their original mechanism. The modifications to three lot sizing rules suggested improve the basic rules by examining the demand variations from period to period through Economic Part Period computations.

Each modified heuristic dominates its predecessor in a cost comparison in our experiments. Average cost reduction rates

from their counterparts are very high, more than 80% across the rules. The chances that the three modified heuristics produce the optimal solutions are also very high ranging from 84% to 88%. Furthermore, the three heuristics did not outperform their modified versions on any occasion.

These modified heuristics may not appropriately solve the realistic MRP system problem which necessarily entails multi-level, multi-product situation. However, they may be applied to the multi-level lot sizing problem. When they are used for the multi-level problem, they must be the most time-efficient heuristics which provide good quality solutions.

It was shown that the vertical dependency and horizontal commonality can be taken into account. The recursion procedure reschedules following the logic of the proposed recursion procedure, the original schedule obtained by applying basic single level rules, which are generally used among current MRP users.

The recursion algorithm was developed in this study noting that when an order is partially assembled and partially stored, making decisions on whether or not to combine this entire order offers a great opportunity for savings in total inventory cost.

For the study in this dissertation, two other approaches, besides the recursion algorithm, are analyzed to examine and compare effectiveness and efficiency of our proposed recursion algorithm. One is McLaren-Whybark's setup cost adjustment mechanism which was devised for the multi-level lot sizing problem. The other is a sequential application of single level

lot sizing rules to all the items residing in the multi- level Both the recursion algorithm and McLaren-Whybark's adjustment method obtain cost benefits over the single level rules by either modifying the lot schedule obtained by single level rules or modifying cost information and using the adjusted cost data in single level rules.

The three treatments, i.e., recursion algorithm, McLaren-Whybark setup cost adjustment and a sequential application of single level lot sizing rules, were applied to the five selected single level lot sizing rules of which three are the modified rules in this study. The treatments and individual rules were compared with respect to cost performance and computational efficiency. Factors such as the number of levels in the product structure, the degree of commonality and the master schedule variability were varied so as to study the performance of the treatments and lot sizing techniques over the various operating conditions. The limitations of the experiment were the assumptions of unlimited production capacity and no uncertainty in demand.

All the lot sizing rules with recursion algorithm performed very well with respect to cost performance. In particular, the MLTC and MEOQ rules with recursion algorithm provided the lowest total cost solutions, on average, and outperformed W-W with McLaren-Whybark adjustment, which was used as the base procedure because of its superior cost performance. Furthermore, they have much less time requirements. The other rules with recursion algorithm also performed significantly better than other

treatment rules. However, tne recursion algorithm generally had two to three times larger time requirements compared to the other two treatments.

Other obsevations on the performance of recursion algorithm were also made from the computational results. The first was that the inventory costs decreased rapidly as the degree of commonality increased. The second was that the inventory costs performed increasingly better as the number of the levels in the product structure increased. The third was that the recursion algorithm worked better in the lower end of master schedule variability rather than in the high end of variability. However, in both variability levels, it outperformed the other two treatments rules.

6.2 Contributions

The research in this dissertation has addressed one important problem area of MRP system: multi-level lot sizing, and suggests heuristic methods to obtain better solutions.

There are two specific areas in which contributions have been made:

(1) A recursion algorithm has been developed and tested. It exploits useful and valuable information, such as other items' order schedules, especially relevant items in terms of dependency and commonality relationships. Other approaches have neglected or failed to utilize them. The recursion procedure may be used with any existing lot sizing technique for multi-level lot sizing in an MRP system.

(2) Three popular single level lot sizing rules have been modified and documented. These rules can also be used as multi-level lot sizing rules. In the case of single level, they performed very well and outperformed their predecessors and compared very favorably with optimal W-W for single level lot sizing problem.

They performed very well with the recursion algorithm for multi-level lot sizing problems as well. Results from the simulation experiment indicate that although the modified rules have larger computational requirements, they provide very high quality solutions across the wide variety of operating conditions. The question is the trade-off between extra-time requirements and cost savings in inventory investment resulting from using the recursion algorithm. The impressive advances in computer technology would make more feasible and practical the introduction of the recursion algorithm to the real world.

6.3 Suggestions for Further Research

There are several directions in which the research in this dissertation may be extended. The first direction involves improving some drawbacks, in the proposed algorithm. Another area involves relaxing some of the limiting assumptions made in the study. A third area involves the field testing of the proposed recursion algorithm.

Perhaps the most important and urgent area for further research is the area of time reduction efforts in the recursion algorithm as it usually has relatively larger time requirements.

The extended study may determine the economic number of levels in the high end of product structure hierarchy to which the recursion procedures are applied that will provide more time-efficient solutions but not sacrificing solution quality in terms of cost. This could be done by varying the number of top levels to which the procedures are applied in a series of experiments and comparing the performances of variations.

The study assumed that rough-cut capacity considerations were taken into account in obtaining the master schedule. It also assumed that there were no capacity limitations for lower level items as well. There arises some questions as to whether the larger lot sizes, due to combining the orders following the logic of recursion procedure, could cause any difficulties or conflicts with respect to capacity. As most production systems have limited capacity and limited resources, an extension to this study would be the inclusion of capacity limitations.

For this study, demand was assumed to be deterministic and the study did not allow for backorders. Whybark and Williams (1976) have examined the effects of uncertainty in the MRP system. Kumar (1983) analyzed the efficacy of buffering techniques such as safety stock and safety lead times when uncertainty is present using a simulation study. Thus a useful extension to this research is to study the effects of stochastic demand on the performance of the treatments and lot sizing techniques.

Besides relaxing those assumptions of the study, there is yet another direction in which the study may be extended, namely

field test of the recursion algorithm proposed in the dissertation. This may be done after extensive tests of the recursion algorithm with actual data from practice.

REFERENCES

American Production and Inventory Control Society, "State of the Art Survey : A Preliminary Analysis," Production and Inventory Management, Vol. 15, No. 4, 1974.

Anderson, R., R. Schroeder, S. Tupy, and E. White, "Material Requirements Planning Systems: the State of the Art," Production and Inventory Management, Vol. 23, No. 4, 1982.

Barbosa, L. C., and M. Friedman, "Deterministic Inventory Lot Size Models - A General Root Law," Management Science, Vol. 24, No. 8, 1978, 819-826.

Biggs, J. R., S. H. Goodman, and S. T. Hardy, "Lot Sizing Rules in a Hierarchical Multi-Stage Inventory System," Production and Inventory Management, Vol. 18, No. 1, 1977, 104-115.

Biggs, J. R., "An analysis of Heuristic Lot Sizing and Sequencing Rules on the Performance of a Multi-Stage Production - Inventory System Utilizing Material Planning Techniques," Unpublished Ph.D. thesis, The Ohio State University, 1975.

Biggs, J. R., "Heuristic Lot Sizing and Sequencing Rules in a Hierarchical Multi-Stage Inventory System," Decision Sciences, Vol. 10, No. 1, 1979, 96-115.

Blackburn, J. D., and R. A. Millen, "Selecting a Lot-Sizing Technique for a Single-Level Assembly Process: Part I Analytical Results," Production and Inventory Management, Vol. 20, No. 3, 1979a, 42-48.

Blackburn, J. D., and R. A. Millen, "Selecting a Lot-Sizing Technique for a Single-Level Assembly Process: Part II Empirical

Results," Production and Inventory Management, Vol. 20, No. 4, 1979b, 41-52.

Blackburn, J. D., and R. A. Millen, "Improved Heuristics for Multi-Stage Requirements Planning Systems," Management Science, Vol. 28, No. 1, 1982.

Choi, H. G., E. M. Malstrom, and R. J. Classen, "Computer Simulation of Lot-Sizing Algorithms in Three-Stage Multi-Echelon Inventory Systems," Journal of Operations Management, Vol. 4, No. 3, 1984, 259-277.

Clark, A. J., and H. Scarf, "Optimal Policies for a Multi-Echelon Inventory Problem," Management Science, Vol. 6, No. 4, 1960, 475-490.

Collier, D. A., " The Measurement and Operating Benefits of Component Part Commonality," Decision Sciences, Vol. 12, No. 1, 1981.

Collier, D. A., " Multi-Level Lot-Size Strategy Performance and Selection in a Material Requirements Planning System," Unpublished Ph. D. Thesis, The Ohio State University, 1978.

Crowston, W. B., W. H. Hausman, and W. R. Kampe, "Multi-Stage Production for Stochastic Seasonal Demand," Managemaent Science, Vol. 19, No. 8, 1973, 924-935.

Crowston, W. B. and M. H. Wagner, " Dynamic Lot Size Models for Multi-Stage Assembly Systems," Management Science, Vol. 20, No. 1, 1973, 14-21.

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 Transaction, Vol. 4, No.

4, 1972, 313-317.

Crowston, W. B., M. H. Wagner, and J. F. Williams, "Economic Lot Size Determination in Multi-Stage Assembly Systems," Management Science, Vol. 19, No. 5, 1973, 517-527.

DeMattteis, J. J., and A. G. Mendoza, "An Economic Lot-Sizing Technique," IBM Systems Journal, Vol. 7, No. 1, 1968, 30-46.

Doll, C. L., and D. C. Whybark, "An Iterative Procedure for the Single Machine Multi-Product Lot Scheduling Problem," Management Science, Vol. 20, No. 1, 1973, 50-55.

Eisenhut, P. S., "A Dynamic Lot Sizing Algorithm with Capacity Constraints," AIIE Transactions, Vol. 7, No. 2, 1975, 170-176.

Elmaghraby, S. E. and V. Y. Bawle, "Optimization of Batch Ordering Under Deterministic Variable Demand," Management Science, Vol. 18, No. 9, 1972, 508-517.

Gabbay, H., "Multi-Stage Production Planning," Management Science, Vol. 25, No. 11, 1979, 1045-1174.

Gorham, T., "Dynamic Order Quantities," Production and Inventory Management, Vol. 10, No. 1, 1968, 75-81.

Groff, G.,"A Lot Sizing Rule for Time-Phased Component Demand," Production and Inventory Management, Vol. 20, No 1, 1979, 47-53.

Harris, F. W., "Operations and Cost," Factory Management Series, A. W. Shaw Co., Chicago, Illinois, 1915, 48-52.

Jacobs, F. R., and B. M. Khumawala, " Multi-Level Lot Sizing in MRP: An Empirical Investigation," CBA Working Paper Series 1980-44, University of Houston, 1980.

Kaiman, R. A., "EOQ vs. Dynamic Programming-Which One to Use for Inventory Ordering," Production and Inventory Management, Vol. 10, No. 4, 1969, 66-74.

Kalymon, B. A., "A DecompositionAlgorithm for Arborescent Inventory Systems," Management Science, Vol. 20, No. 4, 1972, 860-874.

Krajewski, L., and L. P. Ritzman, "Disaggregation in Manufacturing and Service Organizations - Survey of Problems and Research," Decision Sciences, Vol. 8, No.1, 1977, 1-18.

Kumar, K. R., "Hybrid Buffering Policies for Material Requirements Planning Systems," BEBR Working Paper No. 965, University of Illinois at Urbana-Champaign, 1983.

Lambrecht, M. R., and J. Vander Eecken, "A Facilities in Series Capacity Constrained Dynamic Lot Size Model," European Journal of Operational Research, Vol. 2, No. 1, 1978, 42-49.

Lambrecht, M. R., and J.Vander Eecken, "Capacity Constrained Single Facility Dynamic Lot-Size Model," Eurorean Journal of Operational Research, Vol. 2, No. 2, 1978, 132-136.

Love, S. F., "A Facilities in Series Inventory Model with Nested Schedules," Management Science, Vol. 18, No. 5, 1972, 327-338.

McLaren, B. J., and D. C. Whybark, "Multi-Level Lot Sizing Procedures in an MRP Environment," Discussion Paper No. 64, Division of Research, School of Business, Purdue University, 1976.

McLaren, B. J., "A Study of Multiple Level Lot Sizing Procedures for Material Requirements Planning Systems," Unpublished

Ph. D. Thesis, Purdue University, 1977.

Morton, T. E., "An Improved Algorithm for the Stationary Cost Dynamic Lot Size Model with Backlogging," Management Science, Vol. 24, No. 8, 1978, 869-873.

New, C. C., " Lot-Sizing in Multi-Level Requirements Planning Systems," Production and Inventory Management, Vol. 15, No.4, 57-72.

Newson, E. F. P., "Multi-Item Lot Size Scheduling by Heuristic-Part I: With Fixed Resources and Part II: With Variable Resources," Management Science, Vol. 21, No. 10, 1975, 1186-1203.

Orlicky, J., Material Requirements Planning. New York: McGraw Hill, 1975.

Pinkus, C. E., "Optimal Design of Multi-Product, Multi-Echelon Inventory Systems," Decision Sciences, Vol. 6, No. 3, 1975, 492-507.

Prout, H. W., "A Comparison of Some Inventory Models," Working Paper Series, Waterloo, Ontario: Waterloo Lutheran University 1973.

Schwarz, L. G., and L. Schrage, " Optimal and System Myopic Policies for Multi-Echelon Production / Inventory Assembly Systems," Management Science, Vol. 21, No. 11, 1975, 1285-1294.

Silver, E. A., "Coordinated Replenishments of Items Under Time-Varying Demand: Dynamic Programming Formulation," Naval Research Logistics Quarterly, Vol. 26, No. 1, 1979, 141-151.

Silver, E. A., and H. C. Meal, " A Heuristic for Selecting Lot

Size Quantities for the Case of a Deterministic Time-Varying Demand Rate and Discrete Opportunities for Replenishment," Production and Inventory Management, Vol. 14, No. 2, 1973, 64-74.

Simmons, D. M., "Optimal Inventory Policies Under a Hierarchy of Setup Costs," Management Science, Vol. 18, No. 10, 1972, 591-599.

Steinberg, E., and H. A. Napier, "Optimal Multi-Level Lot Sizing for Requirements Planning Systems," Management Science, Vol. 26, No. 12, 1980, 1258-1271.

Swoveland, C., "A Deterministic Multi-Period Production Planning Model with Piecewise Concave Production and Holding-Backorder Costs," Management Science, Vol. 21, No. 9, 1975, 1007-1013.

Taha, H. A., and R. W. Skeith, " The Economic Lot Sizes in Multi-Stage Production Systems," AIIE Transactions, Vol. 2, No. 2, 1970, 157-162.

Theisen, E. C., "New Game in Town - The MRP Lot Size," Production and Inventory Management, Vol. 15, No. 2, 1974, 1-13.

Wagner, H. M., and T. M. Whitin, " Dynamic Version of the Economic Lot Size Model," Management Science, Vol. 5, No. 1, 1958, 89-96.

Wemmerlov, U., "Design Factors in MRP Systems: A Limited Survey," Production and Inventory Management, Vol. 20, No. 4, 1979, 15-34.

Yelle, L., " Material Requirements Lot Sizing : A Multi-Level Approach," International Journal of Production and Research, Vol. 17, No. 3, 1979, 223-232.

Zangwill, W. I., "A Deterministic Multi-Period Production Scheduling Model with Backlogging," Management Science, Vol. 10, No. 1, 1966, 105-119.

Zangwill, W. I., "A Deterministic Multi-Product, Multi-Facility Production and Inventory Model," Operations Research, Vol. 14, No. 3, 1966, 486-507.

Zangwill, W. I., " A Backlogging Model and a Multi-Echelon Model of a Dynamic Economic Lot Size Production System - A Network Approach," Management Science, Vol. 15, No. 9, 1969, 506-527.

APPENDICES

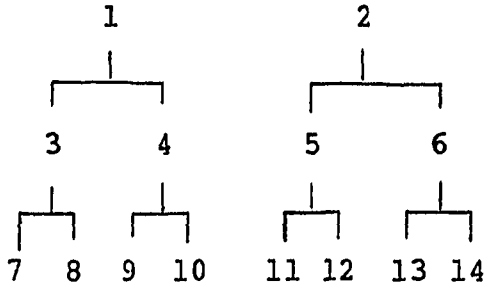
APPENDIX I

PRODUCT STRUCTURES

APPENDIX I

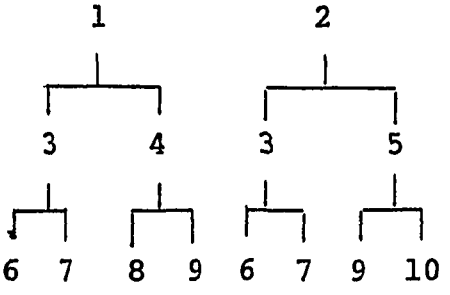
Product Structures

1) 3 levels



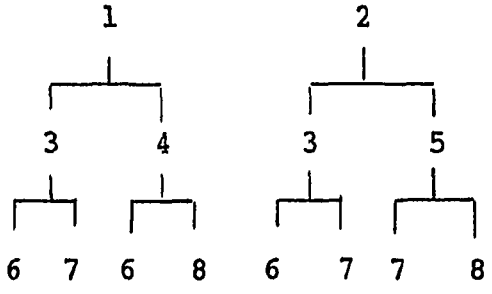
DC = 12 /12 = 1.0

Zero Commonality



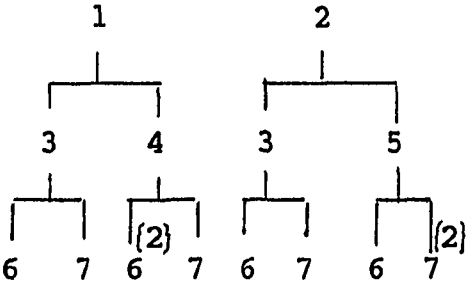
DC = 12 / 8 = 1.5

Low Commonality



DC = 12 / 6 = 2.0

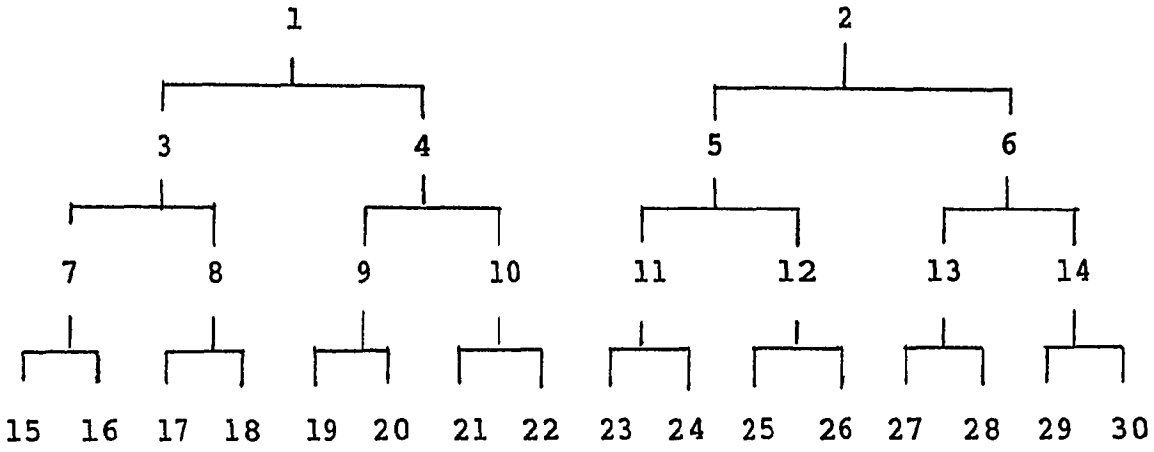
Medium Commonality



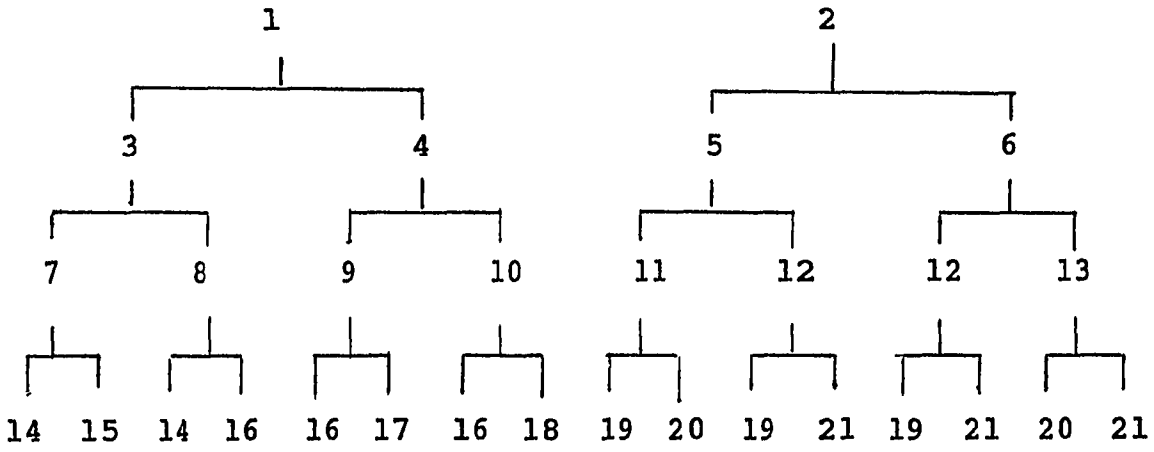
DC = 12 / 5 = 2.4

High Commonality

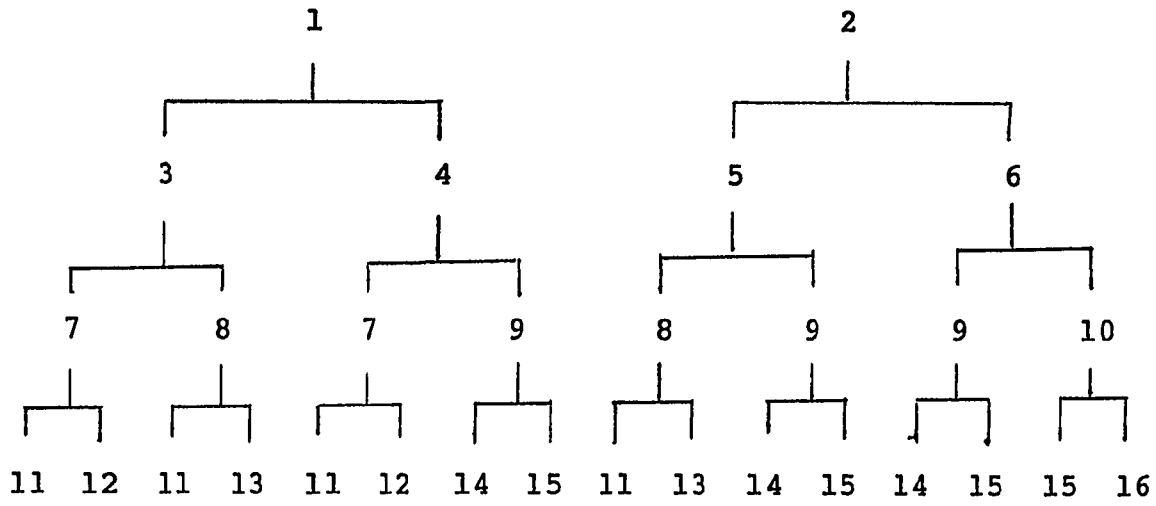
2) 4 levels



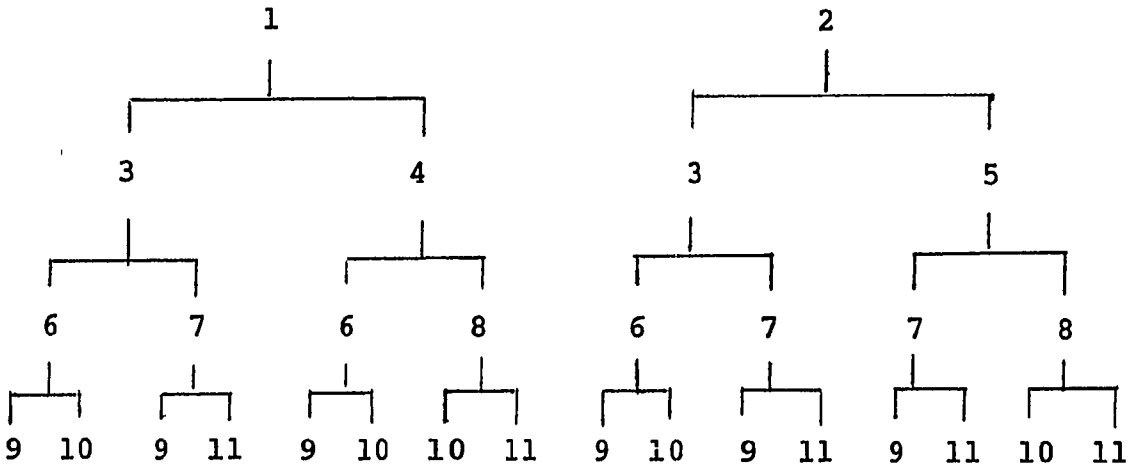
DC = 28 / 28 = 1.0 Zero Commonality



DC = 28 / 19 = 1.47 Low Commonality



DC = 28 / 14 = 2.0 Medium Commonality



DC = 28 / 11 = 2.5 High Commonality

APPENDIX II

COMPUTER PROGRAMS

```
      PROGRAM MRP3Z (INPUT,OUTPUT)
C
C THIS IS FOR 3-LEVEL PRODUCT STRUCTURE
C MULTI-LEVEL LOT-SIZING PROBLEM : MAIN PROGRAM
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C LOCAL VARIABLES
C     INTEGER IPAR(L+L1), MUSE(L+L1)
C READ PRODUCT STRUCTURE MATRIX
C
C     PRINT*,' PRODUCT STRUCTURE MATRIX FOR THIS CASE'
C     PRINT*,' '
      DO 10 IR= 1, S-L2
C       PRINT*, ' ROW ', IR
        READ 11,( STRUCT(IR,IC), IC= 1,S-L )
11      FORMAT ( 12(I1) )
10    CONTINUE
C
C
C VARY DEMAND PATTERN TWICE, IN TERMS OF DEMAND CV
      DO 200 IVARI= 1, 2
C RUN 2 REPLICATION RUNS
      DO 300 IRUN= 1, 2
C GENERATE DEMAND PATTERN
C
      DO 40 I= 1,L
        CALL DEMAND ( I, IRUN, IVARI )
40    CONTINUE
C
C READ COST DATA IN MATRIX COST
C
C     PRINT*,' COST STRUCTURE FOR THIS CASE'
C     PRINT*,' '
C     PRINT*,'_____________________________________________'
      CALL COSTIN
```

```
C      DO 20 IR= 1, 4
C        READ 21,( COST(IR,IC), IC= 1,S )
C1       FORMAT (10(F5.1,1X) )
C0     CONTINUE
C
C ESTABLISH COST AND EPP INFORMATION
C
       DO 80 IR = 1, 3
         DO 80 JC = 1, S
           MAT(IR,JC) = 0.0
80     CONTINUE
C
       DO 81 I = 1, S
         CALL COSDATA ( I )
81     CONTINUE
C
       CUMMT= 0.0
       DO 82 I = 1, S
         CALL MATRIX ( I )
         CUMMT= CUMMT+ MTIME
82     CONTINUE
C
C
       DO 84 1 = 1, S
         CALL MCLAREN ( I )
         CUMMT= CUMMT+ MTIME
84     CONTINUE
C
C
       IU= L
       IV= IU+ L1
       IW= IV+ L2
       IX= IW+ L3
       IY= IX+ L4
C
C USING LOT-SIZING RULES, OBTAIN LOT SCHEDULE AND SYSTEM COST
C
C USE THREE DIFFERENT TREATMENTS
C FIRST, ORIGINAL SINGLE LEVEL RULES WITH PROPOSED RECURSION PROC.
C SECOND, ORIGINAL SINGLE LEVEL RULES ( CONTROL GROUP )
C THIRD, SINGLE LEVEL RULES WITH MCLAREN SETUP COST ADJUST. MECH.
C
      DO 1000  ILOOP = 1, 3
        GO TO ( 1001, 1002, 1003) ILOOP
C001    PRINT*,'SINGLE LEVEL RULES USED WITH RECURSION'
1001    A = 1
        GO TO 1900
C002    PRINT*,'SINGLE LEVEL RULES ONLY USED'
1002    A = 1
        GO TO 1900
C003    PRINT*,'SETUP COST ADJUSTMENT USED-MCLAREN'
1003    A = 2
C
1900  CONTINUE
```

```
C
      DO 100 IFLAG = 1, NR
C
C
C INITIALIZE MRP TABLEAU
      DO 29 I= 1, L
        DO 29 J= 2,3
          DO 29 T= 1, PD
            TAB(I,J,T)= 0
29    CONTINUE
        DO 30 I= L+1, S
          DO 30 J= 1, 3
            DO 30 T= 1, PD
              TAB (I,J,T) = 0
30      CONTINUE
C
C
C
      IF( ILOOP .GT. 1 ) GO TO 240
      CUMSEC= 0.0
      DO 41 I= 1,L
        CALL LSR ( I, A, IFLAG )
        CALL SUMARY ( I )
        CUMSEC= CUMSEC+ TIME
41    CONTINUE
      DO 42 I = 1, S
        PROD(I) = .FALSE.
        NEED(I)= .FALSE.
42    CONTINUE
C
      ILEVEL = 1
59    GO TO ( 51, 52, 53 ,54 ) ILEVEL
51    LEVELI = L1
      LEVELS = IV+ 1
      GO TO 60
52    LEVELI = L2
      LEVELS = IW+ 1
      GO TO 60
53    LEVELI = L3
      LEVELS = IX+ 1
      GO TO 60
54    LEVELI = L4
      LEVELS = IY+ 1
C
60    DO 61 K= 1, LEVELI
        I= LEVELS- K
        DO 62 J = 1, S-L2
          IF( STRUCT(J,I-L) .NE. 0 )THEN
            IUSE = STRUCT(J,I-L)
            DO 63 T = 1, PD
              TAB(I,1,T)= TAB(I,1,T)+ IUSE* TAB(J,3,T)
63          CONTINUE
          ENDIF
62      CONTINUE
```

```
C
        CALL LSR ( I, A, IFLAG )
        CALL SUMARY ( I )
        CUMSEC= CUMSEC+ TIME
C
       DNWARD = .FALSE.
        CALL RECUR ( I )
        CUMSEC= CUMSEC+ TIME
61     CONTINUE
C
       DO 70 ITER= 1,ILEVEL
        IF(ILEVEL.GE.2 .AND. ITER.LT.ILEVEL) DNWARD = .TRUE.
        GO TO ( 71, 72, 73, 74 ) ITER
71      IBEGIN= IU+ 1
        IEND= IV
        GO TO 79
72      IBEGIN= IV+ 1
        IEND= IW
        GO TO 79
73      IBEGIN= IW+ 1
        IEND= IX
        GO TO 79
74      IBEGIN= IX+ 1
        IEND= IY
C
79      DO 78 I = IBEGIN, IEND
C         IF ( NEED(I) ) THEN
C           CALL RECUR(I)
C           CUMSEC= CUMSEC+ TIME
C         ENDIF
78      CONTINUE
        DNWARD = .FALSE.
70     CONTINUE
C
69     IF(ILEVEL .EQ. NLEVEL ) GO TO 109
       ILEVEL= ILEVEL+ 1
       GO TO 59
C
C
240    CUMSEC= 0.0
       DO 241 I= 1, S
         CALL LSR( I, A, IFLAG )
         CUMSEC= CUMSEC+ TIME
         IF( I .EQ. S ) GO TO 109
         INEXT= I+ 1
         IF( INEXT .LE. L )GO TO 241
         DO 245 J= 1, S-L2
           IF( STRUCT(J,INEXT-L) .NE. 0 )THEN
             IUSE= STRUCT(J,INEXT-L)
             DO 247 T=1, PD
               TAB(INEXT,1,T)=TAB(INEXT,1,T)+IUSE*TAB(J,3,T)
247          CONTINUE
           ENDIF
245      CONTINUE
241    CONTINUE
```

```
C
      IF( ILOOP .EQ. 3 )THEN
        CUMSEC= CUMSEC+ CUMMT
      ENDIF
109   CPU( IFLAG, ILOOP )= CUMSEC
      DO 110 I= 1, S
        CALL PRINT ( I )
110   CONTINUE
C
100   CONTINUE
C
1000  CONTINUE
C
C
      CALL PRINOUT
300   CONTINUE
C
200   CONTINUE
      STOP
      END
C
      SUBROUTINE LSR ( I, A, JFLAG )
C
C CALLS VARIOUS KINDS OF LOT SIZING RULE
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
780     GO TO (701,702,703,704,705) JFLAG
701       CALL MEOQ ( I, A )
          GO TO 790
702       CALL MPOQ ( I, A )
          GO TO 790
703       CALL MLTC ( I, A )
          GO TO 790
704       CALL  SM ( I, A )
          GO TO 790
705       CALL  WW ( I, A )
C
C
790   RETURN
      END
```

```
      SUBROUTINE  MPOQ ( I, A )
C
C MPOQ : MODIFIED PERIODIC ORDER QUANTITY
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C LOCAL VARIABLES
      INTEGER SUMDEM, QSTAR, NSTAR, START, K, DEM
      REAL AVGDEM
C
      T1 = SECOND ()
C
      SUMDEM = 0
      DO 500 T = 1, PD
        SUMDEM = SUMDEM + TAB(I,1,T)
500   CONTINUE
      AVGDEM = SUMDEM / PD
C
      QSTAR = NINT(SQRT(2*AVGDEM*SETC(A,I)/HOLDC(A,I)))
      NSTAR = NINT( QSTAR / AVGDEM )
      PP = EPP ( A, I )
C
C
      START = 1
      K = 1
      DEM = 0
C
C
510   IF ( TAB(I,1,START) .EQ. 0 ) THEN
        START = START + 1
        GO TO 510
      ENDIF
      T = START
540   DEM = DEM + TAB(I,1,T)
      IF ( K .GE. NSTAR ) THEN
550     IF ( DEM .GT. QSTAR ) THEN
          IF ( T .EQ. PD ) THEN
            TAB(I,3,START)= DEM
            GO TO 590
          ENDIF
```

```
            T= T+ 1
            K= K+ 1
            IF( (K-1)* TAB(I,1,T) .GT. PP )THEN
              TAB(I,3,START)= DEM
              T= T- 1
              K= K- 1
            ELSE
              TAB(I,3,START)= DEM+ TAB(I,1,T)
            ENDIF
          ELSE
            IF ( T .EQ. PD )THEN
              TAB(I,3,START)= DEM
              GO TO 590
            ENDIF
            K= K+ 1
            T= T+ 1
            IF( (K-1)* TAB(I,1,T) .GT. PP )THEN
              TAB(I,3,START)= DEM
              T= T- 1
            ELSE
              DEM= DEM+ TAB(I,1,T)
              GO TO 550
            ENDIF
          ENDIF
          IF( T .EQ. PD )GO TO 590
530       T= T+ 1
          IF ( T .EQ. PD ) THEN
            TAB(I,3,T) = TAB(I,1,T)
            GO TO 590
          ELSE
            START = T
            IF ( TAB(I,1,START) .EQ. 0 ) THEN
              GO TO 530
            ELSE
              DEM = 0
              K = 1
              GO TO 540
            ENDIF
          ENDIF
        ELSE
          K = K+ 1
          T = T+ 1
          IF ( T .EQ. PD ) THEN
            DEM = DEM + TAB( I,1,T)
            TAB(I,3,START) = DEM
            GO TO 590
          ELSE
            GO TO 540
          ENDIF
        ENDIF
590     CONTINUE
C
        T2 = SECOND ()
        TIME= T2- T1
        END
```

```
-
      SUBROUTINE  SM ( I, A )
C
C S-M : SILVER-MEAL HEURISTIC
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C LOCAL VARIABLES
      INTEGER Z, ORDER, K, X
      REAL G(PD)
C
      T1 = SECOND ()
C
      Z = 1
      ORDER = TAB(I,1,1)
      K = 1
      G(1) = EPP(A,I)
      DO 810 T = 1, PD-1
         IF ( K .EQ. 1 .AND. TAB(I,1,T) .EQ. 0 ) THEN
            Z = T+ .
            ORDER = TAB(I,1,Z)
            GO TO 810
         ELSEIF(K*K*TAB(I,1,T+1).GE.G(K).AND.T.NE.PD-1)THEN
            TAB(I,3,Z) = ORDER
            Z = T+ 1
            K = 1
            ORDER = TAB(I,1,Z)
         ELSEIF(K*K*TAB(I,1,T+1).GE.G(K).AND.T.EQ.PD-1) THEN
            TAB(I,3,Z)= ORDER
            TAB(I,3,PD)= TAB(I,1,PD)
         ELSEIF(K*K*TAB(I,1,T+1).LT.G(K).AND.T.NE.PD-1)THEN
            K = K+1
            X = T+ 1
            ORDER = ORDER+ TAB(I,1,X)
            G(K) = G(K-1)+ (K-1)* TAB(I,1,X)
         ELSEIF(K*K*TAB(I,1,T+1).LT.G(K).AND.T.EQ.PD-1)THEN
            ORDER = ORDER+ TAB(I,1,PD)
            TAB(I,3,Z) = ORDER
         ENDIF
810   CONTINUE
C
      T2 = SECOND ()
      TIME= T2- T1
C     PRINT*,'CPU TIME (SM)= ', TIME
C
      RETURN
      END
```

```
      SUBROUTINE  WW ( I, A )
C
C W-W : WAGNER-WHITIN ALGORITHM
C
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C
C LOCAL VARIABLES
      INTEGER K, N, M, H, START, PLACE(PD)
      REAL WA(PD,PD), SMALL, MIN
C
      T1 = SECOND ()
C
      DO 700 J = 1, PD
        DO 710 K = 1, PD
          IF ( J.EQ.K .AND. TAB(I,1,K).NE.0 ) THEN
            WA(J,K) = SETC(A,I)
          ELSEIF ( J.EQ.K .AND. TAB(I,1,K).EQ.0 ) THEN
            WA(J,K) = 9999999.
          ELSE
            WA(J,K) = 0.
          ENDIF
710     CONTINUE
700   CONTINUE
CARRYING COST ACUMMULATION AND SETUP COST ADDITION
C
      DO 720 J = 1, PD
        DO 730 K = J+1, PD
          IF (WA(J,J) .EQ. 9999999) THEN
            WA(J,K)= WA(J,J)+ K* 9999999
          ELSE
            WA(J,K) = WA(J,K-1)+ (K-J)*HOLDC(A,I)* TAB(I,1,K)
          ENDIF
730     CONTINUE
720   CONTINUE
C
C  FIND MIN. IN EACH COLUMN AND ADD THIS TO NEXT LOWER ROW
C
      DO 750 ICOL = 2, PD
        WA(2,ICOL) = WA(2,ICOL)+ WA(1,1)
750   CONTINUE
```

```
      DO 740 K = 2, PD
        SMALL = WA(1,K)
        DO 760 H = 2, K
          IF ( SMALL .GT. WA(H,K) ) THEN
            SMALL = WA(H,K)
          ENDIF
760     CONTINUE
        N = K+ 1
        DO 770 M = N, PD
          WA(N,M) = WA(N,M) + SMALL
770     CONTINUE
740   CONTINUE
C
C STORE SETUP PERIOD IN ARRAY PLACE (PD)
C
      DO 780 J = 1,PD
        N = PD+ 1- J
        MIN = WA(1,N)
        PLACE(N) = 1
        DO 790 H = 2, N
          IF ( MIN .GE. WA(H,N) ) THEN
            MIN = WA(H,N)
            PLACE(N) = H
          ENDIF
790     CONTINUE
780   CONTINUE
C
C
      DO 781 J = 1, PD
        IPD = PD+ 1- J
        ISET = PLACE(IPD)
        IF ( ISET .LT. IPD ) THEN
          IDUR = IPD- ISET
          DO 782  MPD = 1, IDUR
            PLACE(IPD-MPD) = ISET
782       CONTINUE
        ENDIF
781   CONTINUE
C
C
      START = 1
      TAB(I,3,1) = TAB(I,1,1)
      DO 791 J = 2, PD
        K = PLACE(START)
          IF ( K .EQ. PLACE (J) ) THEN
            TAB(I,3,START) = TAB(I,3,START)+ TAB(I,1,J)
          ELSE
            IF ( J .EQ. PD ) THEN
              TAB(I,3,J) = TAB(I,1,J)
            ELSE
              START = J
            TAB(I,3,START) = TAB(I,1,START)
            ENDIF
          ENDIF
791   CONTINUE
```

```
C
      T2 = SECOND ()
      TIME= T2- T1
C     PRINT*,'CPU TIME (WW)= ', TIME
C
      RETURN
      END
C
C
C
      SUBROUTINE MEOQ ( I, A )
C
C MODEOQ : MODIFIED VERSION OF ECONOMIC ORDER QUANTITY
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z             NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C
C LOCAL VARIABLES
      INTEGER SUMDEM, QSTAR, ORDER, START
      REAL AVGDEM, PP
C
      T1 = SECOND ()
C
      PP = EPP(A,I)
      SUMDEM = 0
      DO 400 T = 1, PD
        SUMDEM = SUMDEM + TAB(I,1,T)
400   CONTINUE
      AVGDEM = SUMDEM / PD
C
      QSTAR = NINT( SQRT( 2* AVGDEM* SETC(A,I) /HOLDC(A,I)))
      DO 460  T = 1, PD
        IF ( TAB(I,1,T) .NE. 0 ) THEN
          START = T
          GO TO 461
        ENDIF
460   CONTINUE
```

```
461    K = 1
       ORDER = 0
       DO 470 T = START, PD
         IF( TAB(I,1,T) .EQ. 0 .AND. K .EQ. 1 ) GO TO 470
         IF( TAB(I,1,T) .EQ. 0 .AND. K .NE. 1 ) GO TO 469
         ORDER = ORDER+ TAB(I,1,T)
         IF ( ORDER .EQ. QSTAR ) THEN
           TAB(I,3,START) = ORDER
           LASTAT = START
           START = T+ 1
462        IF( TAB(I,1,START) .EQ. 0 )THEN
             START= START+ 1
             IF( START .GT. PD )GO TO 470
             GO TO 462
           ENDIF
           K = 0
           ORDER = 0
         ELSEIF ( ORDER .GT. QSTAR) THEN
           IF ( (K-1)* TAB(I,1,T) .LE. PP ) THEN
             TAB(I,3,START) = ORDER
             LASTAT = START
             IF( T .EQ.PD ) GO TO 470
             START = T+ 1
463          IF( TAB(I,1,START) .EQ. 0 )THEN
               START= START+ 1
               IF( START .GT. PD )GO TO 470
               GO TO 463
             ENDIF
             K = 0
             ORDER = 0
           ELSE
             ORDER = ORDER- TAB(I,1,T)
             TAB(I,3,START) = ORDER
             IF( T .EQ. PD )THEN
               TAB(I,3,T)= TAB(I,1,T)
             ELSE
               LASTAT= START
               START= T
               IS = START+ 1
               K= 1
               ORDER= TAB(I,1,T)
464            IF( TAB(I,1,IS) .EQ. 0 )THEN
                 IF( IS .EQ. PD )THEN
                   TAB(I,3,START)= ORDER
                 ELSE
                   IS= IS+ 1
                   GO TO 464
                 ENDIF
               ENDIF
             ENDIF
           ENDIF
```

```
          ELSEIF( (K-1)* TAB(I,1,T) .GT. PP )THEN
            ORDER = ORDER- TAB(I,1,T)
            TAB(I,3,START) = ORDER
            IF( T .EQ. PD )THEN
              TAB(I,3,T)= TAB(I,1,T)
            ELSE
              LASTAT= START
              START= T
              IS = START+ 1
              K= 1
              ORDER= TAB(I,1,T)
465           IF( TAB(I,1,IS) .EQ. 0 )THEN
                IF( IS .EQ. PD )THEN
                  TAB(I,3,START)= ORDER
                ELSE
                  IS= IS+ 1
                  GO TO 465
                ENDIF
              ENDIF
            ENDIF
          ELSEIF ( T .EQ. PD ) THEN
            IF ( (T-LASTAT)*TAB(I,1,T) .LE. PP )THEN
              TAB(I,3,LASTAT)= ORDER+ TAB(I,3,LASTAT)
            ELSEIF(( T- START )* TAB(I,1,T) .GT. PP ) THEN
              TAB(I,3,START)= ORDER- TAB(I,1,T)
              TAB(I,3,T) = TAB(I,1,T)
            ELSE
              TAB(I,3,START)= ORDER
            ENDIF
          GO TO 470
          ENDIF
469       K = K+ 1
C
          IF( T .EQ. PD ) TAB(I,3,START)= ORDER
470     CONTINUE
C
C
        T2 = SECOND ()
        TIME= T2- T1
C       PRINT*,'CPU TIME (MEOQ)= ', TIME
C
        RETURN
        END
        SUBROUTINE MLTC ( I, A )
C
C MODLTC : MODIFIED VERSION OF LEAST TOTAL COST
C
C PROGRAM PARAMETERS
        INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
        PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
       Z             NR= 5, NLEVEL= 2 )
C
```

```
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C LOCAL VARIABLES
      INTEGER ORDER, K, Z
      REAL SUM, CUM, PP, ADD
C
      T1 = SECOND ()
C
      PP = EPP(A,I)
      ORDER = 0
      K = 1
      Z = 1
      CUM = 0
      DO 600 T = 1, PD
      IF (K .EQ. 1 .AND. TAB(I,1,T) .EQ. 0 ) THEN
        Z = T+
      ELSE
        ADD = (K-1)* TAB(I,1,T)
        CUM = CUM + ADD
        IF ( ADD .GT. PP ) THEN
          TAB(I,3,Z) = ORDER
          IF ( T .EQ. PD ) TAB(I,3,T) = TAB(I,1,T)
          Z = T
          K = 1
          ORDER = TAB(I,1,T)
          CUM = 0
        ELSEIF ( CUM .GT. PP ) THEN
          IF ((CUM-PP) .LT. (PP-(CUM-ADD)))THEN
            ORDER= ORDER+ TAB(I,1,T)
            TAB(I,3,Z) = ORDER
            IF ( T .NE. PD ) THEN
              Z = T+ 1
              K = 0
              ORDER = 0
              CUM = 0
            ENDIF
          ELSE
            TAB(I,3,Z) = ORDER
            Z = T
            CUM = 0
            K = 1
            ORDER = TAB(I,1,T)
          ENDIF
        ELSE
          ORDER = ORDER+ TAB(I,1,T)
        ENDIF
```

```
      IF ( T .EQ. PD ) TAB(I,3,Z) = ORDER
      K = K+1
      ENDIF
C
600   CONTINUE
C
      T2 = SECOND ()
      TIME= T2- T1
C     PRINT*,'CPU TIME (MLTC)= ', TIME
C
      RETURN
      END
C
C
C
      SUBROUTINE SUMARY ( I )
C
C COMPUTES ON HAND INVENTORY VALUE
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
      DO 900 T = 1, PD
        IF( T .GT. 1 )THEN
          LASTI= TAB(I,2,T-1)
        ELSE
          LASTI= 0
        ENDIF
C
        TAB(I,2,T)= LASTI+ TAB(I,3,T)- TAB(I,1,T)
C
900   CONTINUE
C
C
      RETURN
      END
C
C
C
```

```
      SUBROUTINE DEMAND ( I, IREPLI, ICOEFF )
C
C THE 1ST PART : TIME VARYING OR CONSTANT DEMAND ( GIVEN )
C THE 2ND PART : UNIFORMLY DISTRIBUTED DEMAND
C THE 3RD PART : NORMALLY DISTRIBUTED DEMAND
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C LOCAL VARIABLES
      INTEGER NRD
      REAL DR( 52 )
      DOUBLE PRECISION DSEED
      DATA NCALL/0/
C
C GIVEN TIME VARYING OR  CONSTANT DEMAND
C     PRINT*,'TYPE IN DEMANDS FOR ITEM ', I
C     PRINT*,'                    FOR THE TIME PERIODS OF ',PD
C     READ 301, ( TAB(I,1,T), T=1, PD )
C01   FORMAT ( 12(I3,1X) )
C
C UNIFORMLY DISTRIBUTED DEMAND PATTERN
C
C
C
C NORMALLY DISTRIBUTED DEMAND PATTERN
C
C GENERATE RANDOM NUMBER BY RANDOM NUMBER GENERATOR
C AND TRANSFORM THIS INTO DEMAND QUANTITY
C
C
      NRD= PD
C
      IF( NCALL .EQ. 0 )DSEED= 173541.D0
      NCALL = 1
C
C
C
C
      CALL GGNML ( DSEED, NRD, DR )
```

```
C
      GO TO ( 31, 32 ) ICOEFF
31    BOUND= -3.90
      GO TO 334
32    BOUND= -.50
C
C
334   DO 341 INUM= 1, PD
        IF( DR(INUM) .LE. BOUND )THEN
          TAB(I,1,INUM)= 0
        ELSE
          IF( ICOEFF .EQ. 1 )RMEAN = 50.0
            RMEAN = 40.0
          TAB(I,1,INUM)= INT( RMEAN*( 1.0- DR(INUM)/ BOUND ))
        ENDIF
341   CONTINUE
      RETURN
      END
C
C
C
      SUBROUTINE  COSDATA ( I )
C
C THIS COMPUTES SETUP AND CARRYING COST AND EPP FOR EACH ITEM
C IN THE SYSTEM.
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C
C
      SETC(1,I) = COST(1,I)
      HOLDC(1,I) = COST(2,I)
      EPP(1,I) = SETC(1,I)/HOLDC(1,I)
C
C
C
      RETURN
      END
```

```
C
C
C
      SUBROUTINE MATRIX ( I )
C
C THIS COMPUTES EOQ, AVERAGE DEMAND, TIME BETWEEN ORDER
C FOR EACH ITEM
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
      T1= SECOND()
C
      IF ( I .GE. 1 .AND. I .LE. L ) THEN
      SUM = 0.0
        DO 260 T = 1, PD
          SUM = SUM + TAB(I,1,T)
260     CONTINUE
C
        AVG = SUM / PD
        MAT(2,I) = AVG
      ELSE
        DO 270 J = 1, S-L2
          IF ( STRUCT(J,I-L) .NE. 0 ) THEN
            IUSE = STRUCT(J,I-L)
            MAT(2,I) = MAT(2,I) + IUSE * MAT(2,J)
          ENDIF
270     CONTINUE
      ENDIF
C
C COMPUTE EOQ FOR EACH ITEM
C
        MAT(1,I) = NINT( SQRT( 2* MAT(2,I)* EPP(1,I)))
C COMPUTE TBO FOR EACH ITEM  I
C
      MAT(3,I) = MAT(1,I)/MAT(2,I)
C
      T2=SECOND()
      MTIME= T2-T1
C
      RETURN
      END
```

```
C
      SUBROUTINE MCLAREN ( I )
C
C SETUP COST ADJUSTMENT MECHANISM ( MC LAREN )
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, ON, NEED, DNWARD, MOVE
C
      T1= SECOND()
C
      IF( I .GE. 1 .AND. I .LE. S-L2 )THEN
      TEMSUM = 0.0
        DO 280 J = 1, S-L
          IF( STRUCT(I,J) .NE. 0 )THEN
            TEMSUM= TEMSUM+ SETC(1,L+J)* MAT(3,I)/MAT(3,L+J)
            IF( J .EQ. S-L ) THEN
              SETC(2,I)= SETC(1,I)+ TEMSUM
              HOLDC(2,I)= HOLDC(1,I)
            ENDIF
          ELSEIF( J .EQ. S-L )THEN
            SETC(2,I)= SETC(1,I)+ TEMSUM
            HOLDC(2,I)= HOLDC(1,I)
          ENDIF
 280    CONTINUE
      ELSE
        SETC(2,I)= SETC(1,I)
        HOLDC(2,I)= HOLDC(1,I)
      ENDIF
C
      EPP(2,I)= SETC(2,I)/HOLDC(2,I)
C
      T2= SECOND()
      MTIME= T2-T1
C
      RETURN
      END
C
```

```
C
      SUBROUTINE RECUR ( II )
C
C RECURSION PROCESS IS DONE STARTING FROM THE LOWEST LEVEL
C UP TO THE HIGHEST LEVEL
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C LOCAL VARIABLES
      INTEGER IP(L+L1)
      LOGICAL SKIP
C
      T1 = SECOND()
C
      SKIP= .FALSE.
      IUSE = 0
      DO 110 J= 1, S-L2
        IF( STRUCT(J,II-L) .NE. 0 )THEN
          IUSE= IUSE+ 1
          IP(IUSE)= J
        ENDIF
110   CONTINUE
C
C RECURSION PROCESS BEGINS
C
      IF( IUSE .EQ. 1 ) GO TO 190
      GO TO 290
C
190   IJ= IP(1)
---   NSUM= 0
      DO 120 T = 1, PD
        IF(TAB(I1,2,T).EQ.0) GO TO 120
        IF( T .EQ. PD ) GO TO 300
      IF( SKIP )THEN
        NSUM= NSUM+ TAB(II,1,T)
        IF( NSUM .EQ. ISUM )THEN
          SKIP= .FALSE.
          NSUM= 0
        ENDIF
        GO TO 120
      ENDIF
```

```
          IN= T+ 1
          IF( TAB(II,2,T) .EQ. TAB(II,1,IN))THEN
            DEL=SETC(1,IJ)+(HOLDC(1,II)-HOLDC(1,IJ))*TAB(IJ,3,IN)
            IF( DEL .LT. 0.0 )GO TO 189
              TAB(II,1,T)= TAB(II,1,T)+ TAB(II,1,IN)
              TAB(II,1,IN)= 0
              TAB(II,2,T)= 0
              TAB(IJ,3,T)= TAB(IJ,3,T)+ TAB(IJ,3,IN)
              TAB(IJ,2,T)= TAB(IJ,2,T)+ TAB(IJ,3,IN)
              CALL CHANGE ( IJ, II, T, IN )
              TAB(IJ,3,IN)= 0
              NEED(IJ)= .TRUE.
189         IF( IN .EQ. PD ) GO TO 300
            GO TO 120
          ELSEIF( TAB(II,2,T) .GT. TAB(II,1,IN))THEN
            ISUM= 0
            SKIP= .TRUE.
          ENDIF
120     CONTINUE
        GO TO 300
C
C WHEN ITEM IS COMMONLY USED FOR MULTIPLE PARENTS
C
290     DO 220 T= 1, PD
        IF( TAB(II,2,T) .EQ. 0 ) GO TO 220
        IF( T .EQ. PD ) GO TO 300
        IN= T+ 1
C
C
        IF( TAB(II,2,T) .EQ. TAB(II,1,IN) )THEN
          DO 230 J= 1, IUSE
            IJ= IF(J)
            IF( TAB(II,1,IN) .EQ. TAB(IJ,3,IN) )THEN
              IF( TAB(IJ,3,T) .EQ. 0 ) GO TO 220
              DEL=SETC(1,IJ)+(HOLDC(1,II)-HOLDC(1,IJ))*TAB(IJ,3,IN)
              IF( DEL .LT. 0.0 )THEN
                CALL FIND ( IJ, II, T, IN, DEL )
                IF(.NOT. MOVE )THEN
                  GO TO 279
                ENDIF
              ENDIF
                  TAB(II,1,T)= TAB(II,1,T)+ TAB(II,1,IN)
                  TAB(II,1,IN)= 0
                  TAB(II,2,T)= 0
                  TAB(IJ,3,T)= TAB(IJ,3,T)+ TAB(IJ,3,IN)
                  TAB(IJ,2,T)= TAB(IJ,2,T)+ TAB(IJ,3,IN)
                  CALL CHANGE ( IJ, II, T, IN )
                  TAB(IJ,3,IN)= 0
                  NEED(IJ)= .TRUE.
279           IF( IN .EQ. PD ) GO TO 300
              GO TO 220
```

```
          ELSEIF(TAB(II,1,IN).GT.TAB(IJ,3,IN).AND.
     Z    TAB(IJ,3,IN).GT.0) THEN
            IF( TAB(IJ,3,T) .EQ. 0 )GO TO 230
            DEL=SETC(1,IJ)+(HOLDC(1,II)-HOLDC(1,IJ))*TAB(IJ,3,IN)
            CALL FIND ( IJ, II, T, IN, DEL )
            IF( DEL .LT. 0.0 )THEN
              IF(.NOT. MOVE )THEN
                GO TO 230
              ENDIF
            ENDIF
289             TAB(II,1,T)= TAB(II,1,T)+ TAB(IJ,3,IN)
                TAB(II,1,IN)= TAB(II,1,IN)- TAB(IJ,3,IN)
                TAB(II,2,T)= TAB(II,2,T)- TAB(IJ,3,IN)
                TAB(IJ,3,T)= TAB(IJ,3,T)+ TAB(IJ,3,IN)
                TAB(IJ,2,T)= TAB(IJ,2,T)+ TAB(IJ,3,IN)
                CALL CHANGE ( IJ, II, T, IN )
                TAB(IJ,3,IN)= 0
                NEED(IJ)= .TRUE.
          ENDIF
230       CONTINUE
          IF( IN .EQ. FD ) GO TO 300
        ELSEIF( TAB(II,2,T) .GT. TAB(II,1,IN) )THEN
          K= 1
          DO 240 IT= 1, 3
            IF( TAB(II,1,IN) .NE. 0 )GO TO 259
            IN= IN+ 1
C           IF( IN .EQ. FD+1 ) GO TO 300
            K = K+ 1
240       CONTINUE
259       DO 250 J= 1, IUSE
            IJ= IP(J)
            IF( TAB(II,3,IN) .GT. 0 )GO TO 220
            IF( TAB(II,1,IN).GE.TAB(IJ,3,IN) .AND.
     Z      TAB(IJ,3,IN).GT.0 )THEN
              IF( TAB(IJ,3,T) .EQ. 0 )GO TO 250
              DEL=SETC(1,IJ)+K*(HOLDC(1,II)-HOLDC(1,IJ))*
     Z        TAB(IJ,3,IN)
              CALL FIND ( IJ, II, T, IN, DEL )
              IF( DEL .LT. 0.0 )THEN
                IF (.NOT. MOVE ) THEN
                  GO TO 250
                ENDIF
              ENDIF
                TAB(II,1,T)= TAB(II,1,T)+ TAB(IJ,3,IN)
                TAB(II,1,IN)= TAB(II,1,IN)- TAB(IJ,3,IN)
                TAB(II,2,T)= TAB(II,2,T)- TAB(IJ,3,IN)
                TAB(II,2,IN)= 0
                TAB(IJ,3,T)= TAB(IJ,3,T)+ TAB(IJ,3,IN)
                TAB(IJ,2,T)= TAB(IJ,2,T)+ TAB(IJ,3,IN)
                CALL CHANGE ( IJ, II, T, IN )
                TAB(IJ,3,IN)= 0
                NEED(IJ)= .TRUE.
                IF( TAB(II,2,T) .EQ. 0) GO TO 220
            ENDIF
250       CONTINUE
```

```
          IN= IN+ 1
          K = K+ 1
          GO TO 259
        ENDIF
220     CONTINUE
C
300     T2 = SECOND ()
        TIME= T2- T1
C       PRINT*,'CPU TIME (RECUR)= ', TIME
C
C
        RETURN
        END
C
C
        SUBROUTINE CHANGE ( IR, II, T, IN )
C
C DURING RECURSION PROCESS, SCHEDULE CHANGES ARE DONE FOR OTHER
C RELEVANT ITEMS AT THE SAME TIME
C
C PROGRAM PARAMETERS
        INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
        PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
       Z             NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
        COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
       Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
       Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
       Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
        INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
        REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
        LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C LOCAL VARIABLES
        LOGICAL DOWN
C
        DO 310 ID = 1, S-L
          IF( STRUCT(IR,ID) .NE. 0 )THEN
            IC= ID+ L
            IF( IC .LT. II .AND. DNWARD )GO TO 350
            IF( IC .LE. II ) GO TO 310
350         IUNIT= STRUCT(IR,ID)
            ISHIFT= IUNIT* TAB(IR,3,IN)
            TAB(IC,1,T)= TAB(IC,1,T)+ ISHIFT
            DOWN = .FALSE.
            IF(DNWARD.AND.TAB(IC,1,T).NE.TAB(IC,3,T))DOWN=.TRUE.
C           IF( PROB(IC) )THEN
C             GO TO 310
C           ELSE
              IF(TAB(IC,2,T).NE.0)TAB(IC,2,T)=TAB(IC,2,T)-ISHIFT
              IDEM= TAB(IC,1,T)+ TAB(IC,2,T)
              IF( IDEM .EQ. TAB(IC,3,T) ) GO TO 330
```

```
              TAB(IC,3,T)= TAB(IC,3,T)+ ISHIFT
330           TAB(IC,1,IN)= TAB(IC,1,IN)- ISHIFT
              IF ( DOWN .AND. TAB(IC,3,IN) .NE. 0 )
     Z          CALL DOWNCH ( IC, T, IN, ISHIFT )
              IF(TAB(IC,3,IN).NE.0)TAB(IC,3,IN)=TAB(IC,3,IN)-ISHIFT
              IF(TAB(IC,1,IN).EQ.0.AND.TAB(IC,3,IN).NE.0)THEN
                TAB(IC,2,IN)= 0
                INN= IN+ 1
                TAB(IC,3,INN)= TAB(IC,3,INN)+ TAB(II,3,IN)
                TAB(IC,3,IN)= 0
              ENDIF
C           ENDIF
          ENDIF
310     CONTINUE
C
        RETURN
        END
C
C
        SUBROUTINE FIND ( IR, II, T, IN, DEL )
C
C
C PROGRAM PARAMETERS
        INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
        PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z              NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
        COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z         MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z         SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z         CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
        INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
        REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
        LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C
        MOVE = .FALSE.
        DO 360 ID = 1, S-L
          IF( STRUCT(IR,ID) .NE. 0 )THEN
            IC = ID+ L
            IF( IC .LE. II ) GO TO 360
            IF( IC .GT. II )THEN
              IF(TAB(IC,3,IN).GT.0.AND.TAB(IC,3,T).EQ.0)THEN
                PROB(IC)= .TRUE.
              ENDIF
              IF( TAB(IR,3,IN) .EQ. TAB(IC,1,IN) .AND.
     Z            TAB(IC,1,IN) .EQ. TAB(IC,2,T) )THEN
                DELTA=(HOLDC(1,IC)-HOLDC(1,IR))*TAB(IR,3,IN)
                DEL= DEL+ DELTA
                IF ( DEL .GT. 0.0 )THEN
                  MOVE= .TRUE.
                ENDIF
```

```
            ENDIF
          ENDIF
        ENDIF
360   CONTINUE
C
C
C
      RETURN
      END
C
C
C
      SUBROUTINE PRINT ( I )
C
C PRINT MRP TABLEAU FOR EACH ITEM AND INVENTORY COST RESULTS
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C
C LOCAL VARIABLES
      INTEGER LASTI, ROW, COL
      REAL CARRYC(S), SETUPC(S)
C
C
      SUMINV(I) = 0
      NSETUP(I) = 0
      DO 900 T = 1,PD
        IF ( T .GT. 1 ) THEN
          LASTI = TAB (I,2,T-1)
        ELSE
          LASTI = 0
        ENDIF
C
C
        TAB(I,2,T) = LASTI+ TAB(I,3,T)- TAB(I,1,T)
        SUMINV(I) = SUMINV(I)+ TAB(I,2,T)
C
        IF ( TAB(I,3,T) .NE. 0 ) THEN
          NSETUP(I) = NSETUP(I) + 1
        ENDIF
C
900   CONTINUE
```

```
C
C     PRINT*, ' '
C     PRINT*, ' FOR ITEM ', I
C     PRINT*,'_____________________________________________'
      DO 910 ROW = 1, 3
C       PRINT 911, ( TAB(I,ROW,COL), COL = 1, PD )
911     FORMAT (1X, 12(1X,I4) )
910   CONTINUE
C     PRINT*,'_____________________________________________'
C     PRINT*,' '
C
C
      CARRYC(I)= SUMINV(I) * COST(2,I)
C     PRINT*,' CARRYING COST FOR ITEM I : ', CARRYC(I)
C
C
      SETUPC(I) = NSETUP(I) * COST (1,I)
C     PRINT*,' SETUP COST FOR ITEM I    : ', SETUPC(I)
C     PRINT*,' '
C
      IF( I .EQ. 1 )THEN
        TOTCAR= 0.0
        TOTSET= 0.0
        TOTC= 0.0
      ENDIF
C
      TOTCAR= TOTCAR+ CARRYC(I)
      TOTSET= TOTSET+ SETUPC(I)
      TOTC= TOTCAR+ TOTSET
      IF( I .EQ. S )THEN
C       PRINT*,' TOTAL CARRYING COST :  ',TOTCAR
C       PRINT*,' TOTAL SETUP COST :     ',TOTSET
C       PRINT*,' TOTAL INVENTORY COST : ',TOTC
        TOTAL( IFLAG, ILOOP )= TOTC
      ENDIF
C
C
C
      RETURN
      END
C
      SUBROUTINE DOWNCH ( IC, T, IN, JSHIFT)
C
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z             NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
```

```
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C
      DO 310 IK = 1, S-L
        IF( STRUCT(IC,IK) .NE. 0 )THEN
          IG= IK+ L
          IUNIT= STRUCT(IC,IK)
          KSHIFT= IUNIT* JSHIFT
          TAB(IG,1,T)= TAB(IG,1,T)+ KSHIFT
            IF(TAB(IG,2,T).NE.0)TAB(IG,2,T)=TAB(IG,2,T)-KSHIFT
            IDEM= TAB(IG,1,T)+ TAB(IG,2,T)
            IF( IDEM .EQ. TAB(IG,3,T) ) GO TO 330
            TAB(IG,3,T)= TAB(IG,3,T)+ KSHIFT
330         TAB(IG,1,IN)= TAB(IG,1,IN)- KSHIFT
            IF(TAB(IG,3,IN).NE.0)TAB(IG,3,IN)=TAB(IG,3,IN)-KSHIFT
            IF( TAB(IG,1,IN).EQ. 0 .AND. TAB(IG,3,IN).NE. 0 )THEN
              TAB(IG,2,IN)= 0
              INN= IN+ 1
              TAB(IG,3,INN)= TAB(IG,3,INN)+ TAB(II,3,IN)
              TAB(IG,3,IN)= 0
            ENDIF
        ENDIF
310   CONTINUE
C
C
      RETURN
      END
C
C
C
      SUBROUTINE PRINOUT
C
C
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C LOCAL VARIABLES
      REAL AVGCOS(3), AVGCPU(3), PERCOS(5,3), PERCPU(5,3),
     Z     PAVGCO(3), PAVGCP(3)
C
```

```
C
      DO 600 JCOL=1, 3
        AVGCOS(JCOL)= 0.0
        AVGCPU(JCOL)= 0.0
        FAVGCO(JCOL)= 0.0
        FAVGCP(JCOL)= 0.0
600   CONTINUE
C
      DO 900 IROW=1, 5
      - DO 900 JCOL=1, 3
          PERCOS(IROW,JCOL)= 0.0
          PERCPU(IROW,JCOL)= 0.0
900   CONTINUE
C PRINT OUT TOTAL INVENTORY COST TABLE AND TOTAL CPU TABLE
C COST TABLE
      PRINT*,' '
      PRINT*,' TOTAL INVENTORY COST TABLE '
      PRINT*,' '
      PRINT*,'-------------------------------------------------------------'
      PRINT*,'     WITH RECUR     WITHOUT RECUR     MCLAREN-WHYBARK'
      PRINT*,'-------------------------------------------------------------'
      DO 610 IROW= 1, 5
        PRINT 630, ( TOTAL( IROW, JCOL ), JCOL= 1, 3 )
630     FORMAT ( 1X, 3( E20.10 ) )
      PRINT*,'-------------------------------------------------------------'
C
C
610   CONTINUE
      DO 640 IROW= 1, 5
        DO 640 JCOL= 1, 3
          AVGCOS(JCOL)= TOTAL(IROW,JCOL)/ NR + AVGCOS(JCOL)
640   CONTINUE
      DO 650 IROW= 1, 1
        PRINT 653, ( AVGCOS( JCOL ), JCOL= 1, 3 )
653     FORMAT ( 1X, 3( E20.10 ) )
650   CONTINUE
C
C CPU TABLE
      PRINT*,' '
      PRINT*,' TOTAL CPU SECOND TABLE'
      PRINT*,' '
      PRINT*,'-------------------------------------------------------------'
      PRINT*,'     WITH RECUR     WITHOUT RECUR     MCLAREN-WHYBARK'
      PRINT*,'-------------------------------------------------------------'
      DO 660 IROW= 1, 5
        PRINT 670, ( CPU( IROW, JCOL ), JCOL= 1, 3 )
670     FORMAT ( 1X, 3( E20.10 ) )
      PRINT*,'-------------------------------------------------------------'
C
660   CONTINUE
      DO 680 IROW= 1, 5
        DO 680 JCOL= 1, 3
          AVGCPU(JCOL)= CPU (IROW,JCOL)/ NR + AVGCPU(JCOL)
680   CONTINUE
```

```
      DO 690 IROW= 1, 1
        PRINT 693, ( AVGCPU( JCOL ), JCOL= 1, 3 )
693     FORMAT ( 1X, 3( E20.10 ) )
690   CONTINUE
C
      PRINT*,' '
      PRINT*,' '
C
      DO 700 IROW= 1, 5
        DO 700 JCOL= 1, 3
          PERCOS(IROW,JCOL)=TOTAL(IROW,JCOL)/TOTAL(IROW,2)*100
700   CONTINUE
      DO 701 JCOL= 1, 3
      PAVGCO(JCOL)= AVGCOS(JCOL)/ AVGCOS(2) * 100
701   CONTINUE
C
C
C
      DO 800 IROW= 1, 5
        DO 800 JCOL= 1, 3
          IF( CPU(IROW,2) .EQ. 0.0 )THEN
            PERCPU(IROW,JCOL)=9999999.0
            GO TO 800
          ENDIF
          PERCPU(IROW,JCOL)= CPU(IROW,JCOL)/ CPU(IROW,2)*100
800   CONTINUE
      DO 801 JCOL= 1, 3
      PAVGCP(JCOL)= AVGCPU(JCOL)/ AVGCPU(2) * 100
801   CONTINUE
      PRINT*,' '
      PRINT*,' %AGE INVENTORY COST TABLE '
      PRINT*,' '
      PRINT*,'-------------------------------------------------------'
      PRINT*,'     WITH RECUR     WITHOUT RECUR     MCLAREN-WHYBARK'
      PRINT*,'-------------------------------------------------------'
      DO 710 IROW= 1, 5
        PRINT 730, ( PERCOS( IROW, JCOL ), JCOL= 1, 3 )
730     FORMAT ( 1X, 3( E20.10 ) )
      PRINT*,'-------------------------------------------------------'
C
C
710   CONTINUE
      DO 750 IROW= 1, 1
        PRINT 753, ( PAVGCO( JCOL ), JCOL= 1, 3 )
753     FORMAT ( 1X, 3( E20.10 ) )
750   CONTINUE
C
C CPU TABLE
      PRINT*,' '
      PRINT*,' %AGE CPU SECOND TABLE'
      PRINT*,' '
      PRINT*,'-------------------------------------------------------'
      PRINT*,'       WITH RECUR        WITHOUT RECUR       MCLAREN-WHYBARK'
      PRINT*,'-------------------------------------------------------'
```

```
      DO 860 IROW= 1, 5
        PRINT 870, ( PERCPU( IROW, JCOL ), JCOL= 1, 3 )
870     FORMAT ( 1X, 3( E20.10 ) )
      PRINT*,'-------------------------------------------------------'
C
860   CONTINUE
      DO 890 IROW= 1, 1
        PRINT 893, ( PAVGCP( JCOL ), JCOL= 1, 3 )
893     FORMAT ( 1X, 3( E20.10 ) )
890   CONTINUE
C
      PRINT*,' '
      PRINT*,' '
C
C
C
      RETURN
      END
C
C
C
      SUBROUTINE COSTIN
C
C
C
C PROGRAM PARAMETERS
      INTEGER L, L1, L2, L3, L4, S, PD, NR, NLEVEL
      PARAMETER ( L=2, L1= 4, L2= 8, L3= 0,L4=0,S=14,PD=52,
     Z            NR= 5, NLEVEL= 2 )
C
C
C COMMON BLOCKS USED
      COMMON STRUCT(S-L2, S-L),TAB(S,3,PD),COST(2,S),EPP(2,S),
     Z       MAT(3,S), SETC(2,S), HOLDC(2,S), PROB(S), MOVE,
     Z       SUMINV(S), NSETUP(S),NEED(S),LEVELS,DNWARD,IFLAG,
     Z       CPU(NR,3), TOTAL(NR,3), TIME, MTIME, ILOOP
C
      INTEGER STRUCT, TAB, A, T, SUMINV, NSETUP, LEVELS
      REAL COST, EPP, MAT, SETC, HOLDC, CPU, TOTAL, TIME
      LOGICAL FLAG, PROB, OK, NEED, DNWARD, MOVE
C
C LOCAL VARIABLE
      INTEGER NRC, NRS
      REAL CR(6), SR(14)
      DOUBLE PRECISION DSEED
C
C PROVIDES RANDOMIZED COST PARAMETER SET
C PER RUN
C
C
      NRC= 6
      DSEED= $$$$$$$.DO
      CALL GGUBS ( DSEED, NRC, CR )
C
```

```
      DO 310 I= S-L2+1, S
        COST(2,I)= .3
310   CONTINUE
C
      DO 320 I= L+1, S-L2
        SUMCA= 0.0
        DO 322 J= L1+1, S-L
          IF( STRUCT(I,J) .NE. 0 )THEN
            SUMCA= SUMCA+ COST(2,J+L)
          ENDIF
322     CONTINUE
        IF( CR(I) .GE. .67 )THEN
          RECH= .1
          GO TO 324
        ELSEIF ( CR(I) .GE. .33 )THEN
          RECH= .2
          GO TO 324
        ELSE
          RECH= .3
          GO TO 324
        ENDIF
324     COST(2,I)= RECH+ SUMCA
320   CONTINUE
C
      DO 340 I= 1,2
        SUMCA= 0.0
        DO 342 J= 1, L1
          IF( STRUCT(I,J) .NE. 0 )THEN
            SUMCA= SUMCA+ COST(2,J+L)
          ENDIF
342     CONTINUE
C
        IF( CR(I) .GE. .67 )THEN
          RECH= .2
          GO TO 344
        ELSEIF( CR(I) .GE. .33 )THEN
          RECH= .4
          GO TO 344
        ELSE
          RECH= .6
          GO TO 344
        ENDIF
C
344     COST(2,I)= RECH+ SUMCA
340   CONTINUE
C
C
C
```

```
C GENERATE SETUP COST PARAMETERS FOR THE SYSTEM
C
C
C
      NRS= 14
      DSEED= $$$$$$$.DO
      CALL GGURS( DSEED, NRS, SR )
C
C     BASE= 75.0* COST( 2,1)
C
C
      DO 360 I= 1, S
C
        IF( SR(I) .GE. .67 )THEN
          SR3=   50
          GO TO 362
        ELSEIF( SR(I) .GE. .33 )THEN
          SR3= 70
          GO TO 362
        ELSE
          SR3= 90
          GO TO 362
        ENDIF
C
362     COST(1,I)=  SR3
360   CONTINUE
C
C
C
      RETURN
      END
```

VITA

Mr. Seung Baum Kim was born in Taegu, Korea on January 12, 1949. He attended Kyungbook Junior and Kyunggi Senior High Schools from 1961 through 1967. He graduated from Yonsei University summa cum laude in 1975. Before coming to the United States for advanced studies, he had started his career as an assistant account manager at the Korean Exchange Bank, Seoul, Korea.

He was awarded a Rotary International Foundation Graduate Fellowship (1976-1977) and also received the Buchan Memorial Scholarship (1977-1978) while he was working towards his Master of Business Administration degree at Washington University in St.Louis, Missouri.

Upon graduation, he returned to Korea and resumed in 1979 his business career as an assistant manager at Samsung Group Planning and Coordination Office, Seoul, Korea.

In 1980, he became an assistant professor of accounting at Ajou University, Suwon, Korea. In 1981, he came back to the United States to pursue his Ph.D degree at the University of Illinois at Urbana-Champaign.

He has been specializing in Production and Operations Management at the University of Illinois and has worked as a teaching and research assistant since August, 1981.

During these periods, he submitted a paper titled, "Some Cost Effective Modifications of Single Stage Lot Sizing Rules," to the Canadian Journal of Operational Research and Information

Research (INFOR) and is currently preparing his paper on multi-level lot sizing algorithm proposed in his Ph.D dissertation.

He is a member of TIMS, AIDS, and APICS.