Heuristic Procedure for a Multiproduct Dynamic Lot-Sizing Problem with Coordinated Replenishments

Chia-Shin Chung
College of Business Administration, Department of Operations Management and Business Statistics, Cleveland State University, Cleveland, Ohio 44115

H. Murat Mercan
Faculty of Business Administration, Department of Management, Bilkent University, Bilkent, Ankara, Turkey

In this article we develop a heuristic procedure for a multiproduct dynamic lot-sizing problem. In this problem a joint setup cost is incurred when at least one product is ordered in a period. In addition to the joint setup cost a separate setup cost for each product ordered is also incurred. The objective is to determine the product lot sizes, over a finite planning horizon, that will minimize the total relevant cost such that the demand in each period for each product is satisfied without backlogging. In this article we present an effective heuristic procedure for this problem. Computational results for the heuristic procedure are also reported. Our computational experience leads us to conclude that the heuristic procedure may be of considerable value as a decision-making aid to production planners in a real-world setting. © 1994 John Wiley & Sons, Inc.

1. INTRODUCTION

A multiitem dynamic lot-sizing problem with coordinated replenishments is the determination of lot sizes of several products over a planning horizon. These items are usually classified into families of products where the products in each family have similarities in processing. In this type of lot-sizing problem, there is a major (family) setup cost when at least one unit of any item is replenished (produced). A minor (individual) setup cost may also be incurred when a product is procured (produced). The objective is to determine a time-phased procurement (production) schedule that minimizes the total inventory carrying costs, setup costs, and procurement costs under demand constraints. This problem will be referred to as the multiproduct dynamic lot-sizing problem with coordinated replenishments (MDLCR).

Coordination of items in determining lot sizes for manufacturing or service organizations has gained considerable attention, not only because of the savings in operational costs but also because of its acceptance among practitioners. Joint replenishment can be seen both in manufacturing and commercial business organizations which have to deal with hundreds of items. Silver [10] points out some cases in which there is a family structure. The items which share a common supplier, a common mode of transportation, and a common production facility can be considered as a family. Therefore, MDLCR is well suited to distribution requirement planning (DRP) and materials requirement planning (MRP) systems.

Naval Research Logistics, Vol. 41, pp. 273–286 (1994)

CCC 0894-069X/94/020273-14

Retail stores usually supply consumer goods from either regional warehouses and/or central warehouses, and regional warehouses supply their goods from central warehouses. Each retail store develops its ordering plan based on customer demands and other policy restrictions. In many cases DRP system is decentralized at the lowest level of the echelon. The lowest-level demand points (i.e., retail stores) place orders to regional warehouses. Coordination of items which are supplied from the same regional or central warehouses results in cost savings. Also the same arguments could be made for other demand points of the echelon. Therefore, MDLCR can be used in determining the gross requirements of higher-level demand points (i.e. retail stores, regional warehouses).

In the manufacturing environment where MRP is used to determine the lot sizes of dependent items, master production schedules (MPS) of all end items are the main input to the MRP system. In practice, MPS are usually prepared without capacity consideration and without any coordination of the products sharing a common facility. Coordination of end items which share a common facility could result in cost savings and better utilization of the capacity. MDLCR can, therefore, be used in determining MPS of end items which share a common facility. We would like to point out, however, that the major drawback of current practice and proposed practice in MPS preparation is the capacity blindness of the methods used. An optimal solution procedure for the MDLCR problem with capacity constraints was developed by Erenguc and Mercan [4]. Also, Mercan and Erenguc [8] developed a heuristic procedure for a similar problem, but with capacity constraints on the setup time. Their objective was to minimize the holding cost.

Optimal solution procedures for MDLCR were developed by [3, 7, 11, 12, 14]. Among these authors only Erenguc [3] and Kao [7] presented computational experience. Kao tested his algorithm on a set of two-product, 12-period problems. Erenguc, on the other hand, tested his procedure on different sets of 2-, 4-, 6-, 10-, and 20-product, 12-period problems and reported satisfactory results. All of these algorithms, however, are non-polynomial algorithms and the computation time may become a limiting factor when the problem size increases. Nevertheless, these tests were conducted on mainframe computer facilities, which are much faster than microcomputers. However, most small to medium business organizations use microcomputers and therefore, optimal solution algorithms may not be utilized in practical settings due to computational considerations. As can be seen from our computational study even for 18-period problems, Erenguc's algorithm requires considerable CPU time on microcomputers for problems having high major setup cost and minor setup cost ratio. Therefore, fast and good heuristic procedures are still needed.

In this article we develop a polynomial-time heuristic procedure for MDLCR. The heuristic procedure is tested with the benchmark problems which were provided by Erenguc [3]. This article is organized as follows. In Section 2, we state the problem and some properties of MDLCR. A brief review of previous work is given in Section 3. The heuristic procedure is explained in Section 4. Section 5 is devoted to computational results. We give some concluding remarks in Section 6.

2. THE MDLCR PROBLEM

We use the following notation throughout the article:

T: number of periods in the planning horizon.
J: number of products.

For each item j; $j \in \{1, 2, \ldots, J\}$ and each period t, $t \in \{1, 2, \ldots, T\}$, we define the following:

S_t: joint setup cost incurred if at least one unit of any product is ordered (produced) in period t, $S_t > 0$.
s_{jt}: individual setup cost incurred if at least one unit of item j is ordered (procured) in period t, $s_{jt} \geq 0$.
h_{jt}: cost of holding inventory per unit of item j in period t.
d_{jt}: demand for item j in period t.
p_{jt}: cost of purchasing (producing) one unit of item j at the beginning of period t.
x_{jt}: number of units of product j ordered (produced) at the beginning of period t.
I_{jt}: ending inventory of product j in period t.

The kind of coordination we consider in this article is as follows. A joint setup cost S_t is incurred if some items are replenished in period t, $t \in \{1, 2, \ldots, T\}$. In addition to the joint setup cost S_t, an individual (minor) setup cost s_{jt} is also incurred for every item j, $j \in \{1, 2, \ldots, J\}$ which is replenished in period t, $t \in \{1, 2, \ldots, T\}$. In addition to the setup costs, there are procurement (production) and inventory costs. The objective is to determine an ordering policy that minimizes the total relevant cost over the planning horizon while satisfying time-varying demands without backlogging.

We make the following assumptions:

1. Demands are known and constant.
2. Backlogging is not allowed.
3. Replenishment lead time is negligible.
4. Inventories at the beginning and the end of the planning horizon are zero.

Note that assumptions 3 and 4 are not restrictive and the heuristic procedure can easily be modified to accommodate these cases. The case of backlogging can also be included in the heuristic procedure with some modifications. However, the optimal solution procedure for MDLCR with backlogging has not been studied in the literature. Since there are no benchmark problems against which we can test the performance of the modified heuristic procedure, we choose not to discuss the MDLCR problem with backlogging.

We would like to note that MDLCR can be formulated as a dynamic programming problem. The reader is referred to [7, 12, and 14] for dynamic programming formulation of MDLCR. Also, MDLCR can be formulated as a concave minimization problem (see [3] for concave minimization formulation of the problem.)

MDLCR is a lower-semicontinuous concave minimization problem over a nonempty compact polyhedral set [3]. Therefore, there exists an extreme-point optimal solution to MDLCR. Zangwill [14] and Silver [11], using different approaches, showed that a solution $\{x_{11}, x_{12}, \ldots, x_{JT}; I_{11}, I_{12}, \ldots, I_{JT}\}$ is an extreme-point solution if and only if the following two conditions hold:

$$x_{jt}I_{jt-1} = 0, \qquad \text{for all } j = 1, 2, \ldots, J \text{ and } t = 2, 3, \ldots, T, \tag{1}$$

$$x_{j1}I_{j0} = 0, \qquad \text{for all } j = 1, 2, \ldots, J. \tag{2}$$

The following property is given by Erenguc as a direct consequence of the extreme-point property.

PROPERTY 1: There always exists a minimum cost policy with the property that, for each $j \in \{1, 2, \ldots, J\}$ and each $t \in \{1, 2, \ldots, T\}$, x_{jt} take one of the following values:

$$0, d_{jt}, d_{jt} + d_{jt+1}, \ldots, d_{jt} + d_{jt+1} + \cdots + d_{jT}. \tag{3}$$

3. REVIEW OF PREVIOUS WORK

Multiitem lot-sizing problems with coordinated replenishments have been studied by many researchers. Two lines of research can be seen in this area. The case of the constant demand pattern is studied in [1, 2, 5, 10]. The case of time-varying demand has been studied in [3, 7, 11, 12, and 14]. Except Silver [11], all of these authors dealt with concave procurement cost and concave holding cost functions.

Zangwill [14] developed a dynamic programming formulation for MDLCR. He uses the extreme-point property of the optimal solution to define state and stage variables. He defines time periods as the stage variables and inventory levels of all products in each period as the state variables. In his formulation, the state space increases exponentially as the number of products increases. Therefore, his formulation may not be applicable to a realistic-sized problem.

A different dynamic programming formulation for MDLCR was developed by Kao [7]. In his formulation states are the regeneration points which are defined by Property 1. This formulation is then represented as an acyclic network and the problem is solved by finding the shortest path from the source node to the sink node in this network. Although the state space is smaller than those proposed by Zangwill, the number of nodes and arcs in the acyclic network increases exponentially with the number of products and time periods. Therefore, the problem can easily become intractable even for small-size data.

Silver [11] assumes constant inventory, production, and setup costs. His approach is similar to Kao's approach, and therefore the number of items can easily become a limiting factor in the solution procedure.

Veinott [12] developed a different solution procedure. He considers two different policies for each period $t \in \{1, 2, \ldots, T\}$: ordering at least one product and not ordering at all. Therefore, in his formulation there are 2^{T-1} possible production patterns for each product which can be solved by a Wagner and Whitin algorithm [13]. Veinott's approach results in solving $J2^{T-1}$ shortest-path problems.

More recently, Erenguc [3] developed a finite branch-and-bound algorithm for solving MDLCR. He used a linear function to underestimate the concave objective function. In this formulation, a shortest-path problem is solved at the nodes of the branch-and-bound tree. He also provided computational results of the branch-and-bound algorithm for MDLCR in which the holding cost is a linear function and the replenishment cost is a lower semicontinuous concave function. As demonstrated by Erenguc, the branch-and-bound algorithm usually solves much fewer than $J2^{T-1}$ shortest-path problems. Among these algorithms, Erenguc's approach appears to be potentially the most computationally viable approach.

4. A HEURISTIC PROCEDURE FOR MDLCR

4.1. Informal Description of the Heuristic Procedure

The heuristic procedure proposed in this article is a forward dynamic programming algorithm similar to that of Wagner and Whitin for the single-item problem. A heuristic

solution to the problem is obtained by solving the subproblem of periods 1 through t recursively for period $t = 1, 2, \ldots, T$, which is referred to as subproblem t. Each subproblem t will be solved by comparing two different policies.

Assume that subproblems 1 through $t - 1$ have been solved recursively. To find the best possible solution using the heuristic procedure for subproblem t, we let $k + 1 \leq t$ be the last period prior to the end of period t where a major setup cost may be incurred. And for any product j, let $m_{jl} \leq l$ be the last period prior to the end of period l where $x_{jm_{jl}} > 0$ in the heuristic solution for subproblem l. Hence m_{jl} is the last period in which product j is replenished prior to the end of period l. Two policies are compared in terms of their total costs and a better policy will be selected for each k. For any j, $j \in \{1, 2, \ldots, J\}$, we describe the two policies as follows:

POLICY A: To replenish some of the items in period $k + 1$. The lot size for each of these items covers the demand of that item for periods $k + 1$ through t; i.e., $m_{jt} = k + 1$ for some item j. The demand of the remaining items for periods $k + 1$ through t is satisfied in period m_{jk}, or $m_{jt} = m_{jk}$.

POLICY B: To replenish none of the items in period $k + 1$. For all j, $j \in \{1, 2, \ldots, J\}$, the demand of item j for periods $k + 1$ through t is satisfied in period m_{jk}, or $m_{jt} = m_{jk}$, for all j.

To implement Policy A for subproblem t, we need to determine the items which are replenished in period $k + 1$ and the items which are replenished in period m_{jk}. We let $P_1(k, t)$, $k = 1, 2, \ldots, t - 1$, be the index set of the items whose replenishments are scheduled for period $k + 1$. And let $P_2(k, t)$, $k = 1, 2, \ldots, t - 1$, be the index set of the items which will be replenished in period m_{jk}. Determination of $P_1(k, t)$ and $P_2(k, t)$ will be discussed in Section 4.2. Once $P_1(k, t)$ and $P_2(k, t)$ are determined for each $k < t$, the cost functions for policies A and B can be developed, which are denoted $C_A(k, t)$ and $C_B(k, t)$ respectively. The cost $C_A(k, t)$ includes the following components:

1. The cost of subproblem k.
2. The cost of replenishing product j, $j \in P_1(k, t)$. This cost includes the major setup cost, and for all products j, $j \in P_1(k, t)$, the cost of replenishing product j in period $k + 1$ which covers the demand of item j from periods $k + 1$ through t, and the cost of carrying inventory of item j for periods $k + 1$ through t.
3. For all products j, $j \in P_2(k, t)$, the cost of replenishing product j in period m_{jk} which covers the demand of item j from periods $k + 1$ through t, and the cost of carrying inventory of item j for periods m_{jk} through t.

And the cost $C_B(k, t)$ includes the following components:

1. The cost of subproblem k.
2. For all product j, $j \in \{1, 2, \ldots, J\}$, the cost of replenishing product j in period m_{jk}. This cost includes: the cost of replenishing product j in period m_{jk} whose order quantity is determined under Policy B and the cost of carrying inventory of item j for periods m_{jk} through t.

For each k, the minimum of the two costs, $C_A(k, t)$ and $C_B(k, t)$, will be determined. The heuristic solution for subproblem t can then be obtained by finding the smallest $\min\{C_A(k, t), C_B(k, t)\}$ for all k. The remaining procedure is to compute the smallest cost for subproblem t, $t \in \{1, 2, \ldots, T\}$, recursively and then to retrieve the best ordering policy.

4.2. Formal Description of the Heuristic Procedure

For any j, $j \in \{1, 2, \ldots, J\}$, and any $k < t$, $t \in \{1, 2, \ldots, T\}$, we give the following additional notation:

$C(t)$: Smallest cost of the heuristic solution for subproblem t which includes periods 1 through t.

$D_j(k, t)$: Cumulative demand for periods $k, k + 1, \ldots, t$, which is given by

$$\sum_{i=k}^{t} d_{ji}.$$

$H(j, k, t)$: Cumulative cost of carrying inventory of item j from periods k through t whose order quantity in period k is given by $D_j(k, t)$, which can be expressed as

$$\sum_{i=k+1}^{t} h_{ji} D_j(i, t).$$

$B(k, j)$: Cumulative cost of holding one unit of item j for periods m_{jk} through k, which is given by

$$\sum_{i=m_{jk}}^{k} h_{ji}.$$

Given the above notation, for any $k < t$, the sets $P_1(k, t)$ and $P_2(k, t)$ are defined as follows:

$$P_1(k, t) = \{j: s_{jk+1} \leq B(k, j) * D_j(k + 1, t) + (p_{jm_{jk}} - p_{jk+1}) * D_j(k + 1, t)\},$$

$$P_2(k, t) = \{j: s_{jk+1} > B(k, j) * D_j(k + 1, t) + (p_{jm_{jk}} - p_{jk+1}) * D_j(k + 1, t)\},$$

$$P_1(k, t) \cup P_2(k, t) = \{1, 2, \ldots, K\}.$$

Note that to determine whether item j is in $P_1(k, t)$ or in $P_2(k, t)$, we compare the cost of initiating replenishment of item j in period $k + 1$ versus replenishing this order quantity in the last replenishment period prior to k. If the former cost is smaller than the latter cost, item j is included in $P_1(k, t)$; otherwise it is included in $P_2(k, t)$. Now we can give the mathematical expressions for $C_A(k, t)$ and $C_B(k, t)$.

For any $k < t$, $\text{Cost}_A(k, t)$ is equal to

$$C(k) + \sum_{j \in P_1(k)} (s_{jk+1} + H(j, k + 1, t) + p_{jk+1} * D_j(k + 1, t)) + S_{k+1})$$

$$+ \sum_{j \in P_2(k)} (H(j, k + 1, t) + P_{jm_{jk}} * D_j(k + 1, t) + B(k, j) * D_j(k + 1, t))$$

if $P_1(k, t) \neq \emptyset$; and ∞, if $P_1(k, t) = \emptyset$.

For any $k < t$, $C_B(k, t)$ is given by

$$C_B(k, t) = C(k) + \sum_{j=1}^{k} (H(j, k + 1, t) + p_{jm_{jk}} * D_j(k + 1, t) + B(k, j) * D_j(k + 1, t)).$$

The recursive formula for finding $C(t)$, for $t = 1, 2, \ldots, T$, is as follows:

$$C(t) = \min_{k<t} \min\{C_A(k, t), C_B(k, t)\}. \tag{5}$$

To initiate the heuristic procedure for all $j \in \{1, 2, \ldots, J\}$, we set $h_{j0} = M$, $m_{j0} = 0$, and $C(0) = 0$, where M is a very large number. Note that for each t, $t \in \{1, 2, \ldots, T\}$, m_{jt} must be updated after $C(t)$ is computed. The following procedure updates m_{jt}: If $C(t) = C_A(k, t)$, then

$$m_{jt} = k + 1, \qquad \text{if } j \in P_1(k, t),$$

$$m_{jt} = m_{jk}, \qquad \text{if } j \in P_2(k, t),$$

If $C(t) = C_B(k, t)$, then for all j, $j \in \{1, 2, \ldots, J\}$,

$$m_{jt} = m_{jk}.$$

Note that for any $t \in \{1, 2, \ldots, T\}$, $C(t)$ is obtained either from $C_A(k, t)$ or $C_B(k, t)$ for some $k < t$, whichever cost is minimum. If the minimum cost is obtained from $C_A(k, t)$, for some k, $k < t$, then the last replenishment period of all items which are in $P_1(k, t)$ prior to period t is set to $k + 1$ and the last replenishment period of all items in $P_2(k, t)$ is set to m_{jk}. On the other hand, if the minimum cost is obtained from $C_B(k, t)$ for some k, $k < t$, then the last replenishment period of all items prior to period t is set equal to m_{jk}.

We can now give the formal procedure for the heuristic algorithm.

Step 1. Initialization.
Step 2. Recursion.
  For $t \leftarrow 1$ until $t \leftarrow T$ begin;
    for $k \leftarrow 0$ until $k \leftarrow t - 1$ begin;
      compute $P_1(k, t)$, $P_2(k, t)$;
      compute $C_A(k, t)$, $C_B(k, t)$;
    end;
    compute $C(t) = \min_{k<t} \min\{C_A(k, t), C_B(k, t)\}$;
    update m_{jt};
  end;
Step 3. Obtaining the solution.
  For $j \leftarrow 1$ until $j \leftarrow K$ begin;
    Set $t = T$;
    (3a) Find m_{jt};
      set $x_{jm_{jt}} = D_j(m_{jt}, t)$;
      for $m_{jt} + 1 \le i \le t$,
        $x_{ji} = 0$;
    (3b) Set $t = m_{jt} - 1$;
      repeat (3a) until $t = 0$.
  end;

4.3. Some Properties of the Heuristic Procedure

To find the best possible solution using the heuristic procedure for each subproblem t, two policies are compared in terms of their total costs. Policy A assumes that there are some items in period $k + 1$ which are replenished; hence the major setup cost is incurred. For those items which are replenished in period $k + 1$, their order quantity in period $k + 1$ covers the demand for periods $k + 1$ through t. Minor setup costs, cost of holding inventory, and procurement costs of these items are also added to the total cost.

The demand of the remaining items for periods $k + 1$ through t will be covered in the last replenishment period prior to the end of period k. The reason for not considering covering the demand of these items in other periods is twofold. First, the heuristic solution for subproblem k will be affected, since additional major or minor setup costs might be incurred. Second, it will lose the simple structure of the proposed algorithm.

In calculating the total cost for meeting these demands we do not include the minor setup costs of these items, as they were already included in the total cost for subproblem k. However, there will be additional holding cost of carrying $D_j(k + 1, t)$ units from period m_{jk} to period k. Also, the difference between the cost of replenishing $D_j(k + 1, t)$ units in period m_{jk} and the cost of replenishing this quantity in period $k + 1$ need to be added into the total cost.

Policy B assumes that none of the items will be replenished in period $k + 1$. The demand of any item for periods $k + 1$ through t will be covered in the last production period prior to the end of period k, which is determined by the heuristic solution for subproblem k. Hence the major setup cost in period $k + 1$ will not be incurred. The tradeoff is that additional holding cost of carrying $D_j(k + 1, t)$ units from period m_{jk} to period k for all items will need to be included.

We would like to note that the computational effort required by the heuristic procedure is polynomial and the heuristic procedure finds an optimal solution for a single-item dynamic lot-sizing problems. These two properties are given below.

PROPERTY 2: The heuristic procedure finds a solution in $O(JT^2)$ operations.

PROOF: By calculating $H(j, k + 1, t)$ and $D_j(k + 1, t)$ for $k = t - 1, t - 2, \ldots, 1$ recursively, $C_A(k, t)$ and $C_B(k, t)$ can also be calculated recursively for $k = t - 1, t - 2, \ldots, 1$. Hence the calculation of $C_A(k, t)$ and $C_B(k, t)$, $k = t - 1, t - 2, \ldots, 1$ requires $O(JT)$ operations, which is the same for $C(t)$. This proves that calculating $C(T)$ requires $O(JT^2)$ operations.

PROPERTY 3: The heuristic procedure always finds an optimal solution if there is only one item in the family.

PROOF: To prove Property 3, we will show that our algorithm produces a solution no worse than that of Wagner and Whiten when there is one product in a family.

Since there is only one product, the major and the minor setup costs can be combined into a single setup cost. Hence the optimal solution can be found by the Wagner-Whitin algorithm, which is given by the following recursion:

$$C'(t) = \min_{k<t}\{C'(k) + C''(k + 1, t)\},$$

where $C'(t)$ is the minimum cost for the subproblem t and $C''(k + 1, t)$ is the cost of producing in period $k + 1$ to cover the demands of periods $k + 1, k + 2, \ldots, t$.

It can easily be shown that $C_A(k, t) \leq C'(k) + C''(k + 1, t)$, for any $1 \leq k \leq T$. Hence for any t, $t \in \{1, 2, \ldots, T\}$, the following relationship holds:

$$C(t) = \min_{k<t} \min\{C_A(k, t), C_B(k, t)\} \leq \min_{k<t}\{C_A(k, t)\}$$

$$\leq \min_{k<t} \min\{C'(k) + C''(k + 1, t)\} = C'(t),$$

which implies that $C(T) = C'(T)$.

5. COMPUTATIONAL EXPERIENCE WITH THE HEURISTIC PROCEDURE FOR MDLCR

The heuristic procedure for MDLCR was tested on 150 benchmark problems. In all of the problems, the planning horizon was 12 periods (i.e., $T = 12$) and five sets of problems were generated based on the number of items in a family. The number of items in a family was set to 2, 4, 6, 10, and 20 respectively (i.e., $J \in \{2, 4, 6, 10, 20\}$.) In each set there were 10 problems which were generated randomly. In all the problems for all $j \in \{1, 2, \ldots, J\}$, and $t \in \{1, 2, \ldots, T\}$, h_{jt} was set equal to 1. All the other problem parameters were assumed to follow normal distribution with the following characteristics:

$S_j \ \alpha \ N(120, 36)$
$s_{jt} \ \alpha \ N(60, 18)$
$c_{jt} \ \alpha \ N(4, 1)$
$d_{jt} \ \alpha \ N(50, 20)$, with probability 0.5 and $N(100, 20)$ with probability 0.5.

The reader is referred to [3] for details of the test problems. The heuristic procedure is coded in Turbo Pascal and run on an IBM/PS2/MODEL 70 (with math coprocessor), with 1 MB RAM and 25 MHz. The test problems and the optimal solution procedure were provided by Erenguc. For fair comparison of the CPU times between the heuristic procedure and the branch-and-bound algorithm, the code developed by Erenguc is downloaded to the same machine. We used a Fortran compiler for the branch-and-bound code.

In lieu of the benchmark problems the test was conducted in three phases. The problems were prefixed as A, B, and C for phase 1, 2 and 3, respectively. In the first phase of the computational experiment five sets of problems with $T = 12$ and $J \in \{2, 4, 6, 10, 20\}$ were solved. These problems were generated originally by Erenguc [3]. The problems of this phase are referred to as 1A, 2A, 3A, 4A, and 5A to correspond to five different values of J. A summary of our computational results for Phase one of the computational experiment is reported in Table 1. The heuristic procedure was able to find an optimal solution for 10 of the 50 problems solved in this phase. The average percentage difference between the heuristic solution and the optimal solution is 0.79%. The difference between the heuristic solution and the optimal solution varies between 0.00% and 4.66%. However, in 38 of 50 problems solved, the percentage difference between the heuristic solution and the optimal solution is less than 1%.

Table 1. Test results of the heuristic procedure on five sets of MDLCR problems.

Problem set	Size ($K \times T$)	Percentage difference between the optimal solution and the heuristic solution			No. of problems optimal solution found by heuristic	Computational requirement for the branch-and-bound algorithm (CPU seconds)			CPU time for the heuristic procedure (seconds)
		Average	Min.	Max.		Average	Min.	Max.	
1A	2 × 12	0.81	0.00	4.66	5	1.75	0.77	5.44	1.65
2A	4 × 12	0.59	0.00	2.09	4	4.19	1.53	8.68	1.65
3A	6 × 12	1.27	0.08	2.94	0	3.77	2.14	5.16	1.65
4A	10 × 12	0.74	0.00	1.97	1	4.25	2.19	6.87	1.65
5A	20 × 12	0.56	0.06	0.86	0	2.96	2.41	4.40	2.31

The computational effort required by the heuristic procedure is 1.65 CPU seconds for 2-, 4-, 6-, and 10-item problems and 2.31 CPU seconds for 20-item problems. The computational effort required by the branch-and-bound procedure ranges between 0.77 and 8.68 CPU seconds. The average CPU time required by the optimal solution procedure is 3.38 seconds.

In the second phase of the computational experiments, in all the problems in sets 1A, 2A, 3A, 4A, and 5A, to each S_t, $t \in \{1, 2, \ldots, 12\}$, the numbers 0, 70, 110, 250, and 600 were added, respectively, to assess the effect of the incrase in the major setup cost on the computational effort. The problem sets obtained in this manner are referred to as 1B, 2B, 3B, 4B, and 5B. Computational results for these problems are summarized in Table 2. The heuristic procedure was able to find optimal solutions of seven problems. The average percentage difference between the optimal solution and the heuristic solution is 0.95%. The percentage difference between the optimal solution and the heuristic solution varies between 0.00% and 4.66%. In 28 of 50 problems solved in this phase the heuristic procedure could generate a solution which is within 1% of the optimal solution.

The computational effort required by the heuristic procedure is not sensitive to increase in the major setup cost, whereas the computational effort of the branch and bound algorithm increases with the problem size. The average CPU time required by the branch-and-bound algorithm increases to 12.40 CPU seconds, while the average CPU time of the heuristic procedure remained 1.78 seconds for all sets of problems.

Table 2. Test results of the heuristic procedure on five sets of MDLCR problems where S_t, $t \in \{1, 2, \ldots, 12\}$ increased by 0, 70, 110, 250, and 600 for problem sets 1A, 2A, 3A, 4A, and 5A, respectively.

Problem set	Size ($K \times T$)	Percentage difference between the optimal solution and the heuristic solution			No. of problems optimal solution found by heuristic	Computational requirement for the branch-and-bound algorithm (CPU seconds)			CPU time for the heuristic procedure (seconds)
		Average	Min.	Max.		Average	Min.	Max.	
1B	2 × 12	0.81	0.00	4.66	5	1.75	0.77	5.44	1.65
2B	4 × 12	0.62	0.00	1.90	2	7.41	2.91	14.94	1.65
3B	6 × 12	0.89	0.01	2.07	0	13.12	10.99	23.79	1.65
4B	10 × 12	1.12	0.08	2.74	0	8.71	6.98	11.80	1.65
5B	20 × 12	1.34	0.32	3.05	0	30.99	25.48	38.34	2.31

In the third phase of the computational experiments for all $t \in \{1, 2, \ldots, 12\}$ and $j \in \{1, 2, \ldots, J\}$, s_{jt} were set equal to zero and for each S_t, $t \in \{1, 2, \ldots, 12\}$, the numbers 70, 190, 240, 500, and 1000 were added to their original values for problems 1A, 2A, 3A, 4A, and 5A, respectively. The problem sets thus obtained are denoted 1C, 2C, 3C, 4C, and 5C. Computational results for these problems are reported in Table 3. The heuristic procedure was able to find optimal solution in 26 of 50 problems solved. The average percentage difference between the optimal solution and the heuristic solution is 0.36%. Although the percentage difference between the solutions varies from 0.00% to 3.99%, in 46 of the 50 problems the heuristic procedure could generate a solution within 1% of the optimal solution.

The computational effort required by the heuristic procedure is 1.65 CPU seconds for 2-, 4-, 6-, and 10-item problems and 2.31 CPU seconds for 20-item problems. The average CPU time required by the branch-and-bound algorithm is 21.21 CPU seconds. The computational effort increases as the number of items increases. However, the increase in the computational effort for the heuristic procedure is insignificant as compared to the computational requirement of the branch and bound.

In order to better assess the performance of the heuristic procedure for longer planning horizon problems, we conducted another experiment. In this experiment we solved ten 10-item, 18-period problems. The problem set thus obtained was prefixed as D. The following summarizes our findings:

1. The heuristic procedure was able to find an optimal solution for 2 of the 10 problems solved in the problem set D.
2. The average percentage difference between the heuristic solution and the optimal solution is 1.08%. The maximum difference between the heuristic solution and the optimal solution is only 2%. However, in 6 of the 10 problems solved in problem set D, the percentage difference between the two solutions is less than 1%.
3. The computational effort required by the heuristic procedure ranges between 2.47 and 3.65 CPU seconds. The average CPU time required by the heuristic procedure is 2.58 seconds, whereas the computational effort required by the branch-and-bound algorithm increases sharply. The average CPU time required by the branch-and-bound algorithm is 983.37 seconds. The CPU time for the branch-and-bound algorithm ranges between 708.10 and 1296.84 seconds.

It is clear from Table 4 that the quality of the solutions obtained by the heuristic procedure is not sensitive to the problem parameters. This is evident from comparing

Table 3. Test results of the heuristic procedure for five sets of MDLCR problems where S', $t \in \{1, 2, \ldots, 12\}$ is increased by 0, 70, 190, 240, 500, and 1000, and for all $t \in \{1, 2, \ldots, 12\}$ and $j \in \{1, 2, \ldots, K\}$; $s_{jt} = 0$ for problem sets 1A, 2A, 3A, 4A, and 5A, respectively.

Problem set	Size ($K \times T$)	Percentage difference between the optimal solution and the heuristic solution			No. of problems optimal solution found by heuristic	Computational requirement for the branch-and-bound algorithm (CPU seconds)			CPU time for the heuristic procedure (seconds)
		Average	Min.	Max.		Average	Min.	Max.	
1C	2 × 12	0.41	0.00	3.99	7	3.43	1.60	9.66	1.65
2C	4 × 12	0.43	0.00	3.10	7	10.64	4.62	21.76	1.65
3C	6 × 12	0.39	0.00	0.92	3	13.84	10.11	20.22	1.65
4C	10 × 12	0.38	0.00	1.20	4	29.34	21.37	34.60	1.65
5C	20 × 12	0.17	0.00	0.64	5	48.81	38.56	56.03	2.31

Table 4. Summary results for MDLCR problem

Problem type	Percentage difference between the optimal solution and the heuristic solution			No. of problems heuristic solution is within 1% of optimal solution	Average CPU time for the branch-and-bound algorithm (seconds)	Average CPU time for the heuristic procedure (seconds)
	Average	Minimum	Maximum			
A	0.79	0.00	4.66	38	3.38	1.78
B	0.95	0.00	4.66	28	12.40	1.78
C	0.36	0.00	3.99	46	21.21	1.78
D	1.06	0.00	2.00	6 (out of 10)	983.37	2.58

the average percentage difference between the heuristic and the optimal solutions in Table 4. Furthermore, the heuristic procedure does not seem to perform worse as the problem size gets larger (see Tables 1 and 3). Also, our experiments show no evidence to assert that the difficulty of the problem in terms of obtaining an optimal solution affects the quality of the solutions obtained by the heuristic procedure. The branch-and-bound algorithm required more computational effort to solve problems in set C than the problems in set A. However, it is clear from Tables 1 and 3 that the heuristic procedure was able to obtain better solutions for the problems in set C than the problems in set A.

The computational effort required by the branch-and-bound algorithm is sensitive to the number of items in a family, the number of periods in the planning horizon, and the ratio of the major setup cost to minor setup cost (see Table 4). However, the computational effort required by the heuristic procedure only slightly increases as the number of items in a family and/or the number of periods in the planning horizon increases. We also find that the ratio of the major setup cost to the minor setup cost has no effect on the computational effort of the heuristic procedure.

MDLCR is an NP-hard problem. Therefore, as the number of items in a family and the number of periods in a planning horizon increase, the computational effort required by the branch-and-bound algorithm is expected to increase drastically, as is evidenced in our experiments. On the other hand, the heuristic procedure developed in this article is an $O(JT^2)$ algorithm. Hence, our procedure has a definite computational advantage over the branch-and-bound algorithm.

Since many business organizations have to deal with thousands of items, using the branch-and-bound algorithm may be computationally prohibitive. Also, the heuristic procedure may be appealing tom any practitioners due to its simplicity. Considering these facts and the performance characteristics of the heuristic procedure, one is led to conclude that the heuristic procedure presented in this article may be of considerable value as a decision-making aid to operations managers in practical settings.

6. CONCLUDING REMARKS

In this article we presented a polynomial-time heuristic procedure for a multiitem dynamic lot-sizing problem with joint replenishment costs. The performance of the heuristic procedure is tested on 150 benchmark problems whose optimal solutions are known. In 68% of those problems the heuristic procedure yielded a solution which is within 1% of the optimal solution. Furthermore, the efficiency of the heuristic procedure is not sensitive to the problem parameters. It is apparent from the test results that the

heuristic procedure is capable of solving problems of substantial size with very reasonable computational effort.

The heuristic procedure developed in this article can be extended to find good solutions for MDLCR problems in which backlogging is allowed. However, a new criterion is needed to determine which items are going to be replenished in period $k + 1$ and in the last period prior to period k, and which items will be backlogged for some periods. Consequently, a new cost function must be developed.

It is also possible to extend the heuristic procedure to find a good solution for MDLCR problems with capacity limitation. MDLCR problems with capacity limitation can be seen in group technology environments. In group technology, certain class(es) of items (i.e., families) are manufactured in the same manufacturing cell [6]. However, to the best of the authors' knowledge there are no benchmark problems for the aforementioned variations of MDLCR. One possible research direction is to develop efficient optimal solution algorithms and/or the heuristic procedures for other variations of MDLCR.

Finally, we would like to note that in the experiments we have conducted, the difference in computing time between the optimal and heuristic algorithms may not seem a great deal. But as Table 4 indicates, when the problem size increases, their difference becomes so great that using the optimal algorithm is simply impractical.

ACKNOWLEDGMENT

We would like to thank the anonymous referees for making many helpful comments.

REFERENCES

[1] Brown, R.G., *Decision Rules for Inventory Management*, Rinehart and Winston, New York, 1967, Chap. 5.

[2] Doll, C.L., and Whybark, D.C., "An Iterative Procedure for the Single Machine, Multi Product, Lot Scheduling Problems," *Management Science,* **20**, 50–55 (1973).

[3] Erenguc, S.S., "Multiproduct Dynamic Lot-Sizing Model with Coordinated Replenishments," *Naval Research Logistics,* **35**, 1–22 (1988).

[4] Erenguc, S.S., and Mercan, H.M., "A Multi-Family Dynamic Lot-Sizing Model with Coordinated Replenishments," *Naval Research Logistics*, **37**, 539–558 (1990).

[5] Goyal, S.K., "Determination of Optimum Packing Frequency of Items Jointly Replenished," *Management Science*, **21**, 436–443 (1974).

[6] Groover, M.P., *Automation Production Systems and Computer Aided Manufacturing*, Prentice-Hall, Englewood Cliffs, NJ, 1980.

[7] Kao, E.P.C., "A Multi-Product Dynamic Lot-Size Model with Individual and Joint Set-up Costs," *Operations Research*, **27**, 279–284 (1979).

[8] Mercan, H.M., and Erenguc, S.S., "A Multi-Family Dynamic Lot-Sizing Model with Coordinated Replenishments: A Heuristic Procedure," *International Journal of Production Research*, **21**, 173–189 (1993).

[9] Salomon, M., *Deterministic Lotsizing Models for Production Planning*, Lecture Notes in Economics and Mathematical Systems, Vol. 355, Springer Verlag, Berlin, 1991.

[10] Silver, E.A., "A Simple Method for Determining Order Quantities in Joint Replenishment under Deterministic Demand," *Management Science*, **22**, 1351–1361 (1976).

[11] Silver, E.A., "Coordinated Replenishments of Items Under Time-Varying Demand: Dynamic Programming Formulation," *Naval Research Logistics Quarterly*, **26**, 141–151 (1979).

[12] Veinott, A.F., Jr., "Minimum Concave Cost Solution of Leontief Substitution Models of Multi-Facility Inventory Systems," *Operations Research*, **17**, 262–291 (1969).

[13] Wagner, H.M., and Whitin, T.M., "Dynamic Version of the Economic Lot Size Model," *Management Science*, **5**, 89–96 (1958).
[14] Zangwill, W.I., "A Deterministic Multiproduct, Multifacility Production and Inventory Model," *Operations Research*, **14**, 486–507 (1966).

Manuscript received March 9, 1992
Revised manuscript received July 1993
Accepted July 12, 1993