Lot Sizing in General Assembly Systems with Setup Costs, Setup Times, and Multiple Constrained Resources

Elena Katok • Holly S. Lewis • Terry P. Harrison
Division of Economics and Business, Colorado School of Mines, Golden, Colorado 80401
Department of Management Science and Information Systems, The Pennsylvania State University, University Park, Pennsylvania 16802
Department of Management Science and Information Systems, The Pennsylvania State University, University Park, Pennsylvania 16802

We introduce a heuristic method for finding good, feasible solutions for multiproduct lot sizing problems with general assembly structures, multiple constrained resources, and nonzero setup costs and setup times. We evaluate the performance of this heuristic by comparing its solutions to optimal solutions of small randomly generated problems and to time-truncated Optimization Subroutine Library (OSL) solutions of medium-sized randomly generated problems. In the first case, the heuristic locates solutions averaging 4 percent worse than optimal in less than 1 percent of time required by OSL. The heuristic solutions to medium-sized problems are approximately 26 percent better than solutions OSL finds after 10,000 CPU seconds, and the heuristic finds these solutions in approximately 10 percent of OSL time.
(*Lot Sizing*; *General Assembly System*; *Heuristics*; *Integer Programming*)

1. Introduction

Production planning typically encompasses three timeframes for decision-making: long-term, short-term, and medium-term. Long-term planning focuses on anticipated aggregate needs and involves such strategic decisions as product and process choices, resource planning, and facility location and design. Short-term planning involves day-to-day scheduling and sequencing decisions. Operational, or medium-term, production planning, the focus of this paper, establishes production quantities over a time horizon of several months so as to optimize some performance measure while satisfying existing capacity constraints and meeting known and forecasted demand requirements.

The medium range production planning problem, often referred to as the lot sizing problem, has received much attention in the management science literature (see, for example, Kuik et al. 1994, Bahl et al. 1987, Maes et al. 1991, and Harrison and Lewis 1996). These lot sizing models can be classified by the number of products, the number of capacitated resources, the number of assembly stages, and the nature of the assembly system (linear, pure assembly, or general assembly structure). The single-product, single-stage uncapacitated problem with constant demand rate is easily solved to optimality by the Economic Order Quantity (EOQ) (Harris 1913). When the demand rate is not constant, Wagner and Whitin (1958) provide an efficient dynamic programming algorithm, and many efficient heuristic procedures also exist (e.g., Silver and Meal 1973, Silver and Peterson 1985). The problem becomes difficult when capacity is constrained. Florian et al. (1980) show that several general families of the single-product lot sizing problem with one capacitated resource are NP-hard. Bitran and Yanasse (1982) identify classes of the single-item capacitated problem with linear production and holding costs that are solvable in polynomial time, and those groups that are NP-hard. They also show that the polynomial time

0025-1909/98/4406/0859$05.00

single-product cases become NP-hard when only two products with independent setups are considered. Maes et al. (1991) prove that simply finding a feasible solution to the multi-item problem with multiple constrained resources is NP-complete when setups consume capacity.

1.1. Problem Definition

This research expands the work of Harrison and Lewis (1996) to address the most unrestricted form of the lot sizing problem—one involving multiple products with nonlinear assembly structures, multiple constrained resources, and nonzero setup costs and setup times. The formulation under study is very close to that originally introduced by Billington et al. (1983).

We use the following assumptions to define the multilevel capacitated lot sizing problem.

- Any type of bill of material (BOM) structure can be specified. We define a set $\mathcal{S}_p$ for each product p that includes immediate successors of p in the BOM. $\mathcal{S}_p = \emptyset$ if p is an end item.
- The lead times are one period long, regardless of quantity, meaning that any item p processed during time period t cannot be used either to manufacture an immediate successor of p or to fulfill the external demand for p until period $t + 1$.
- We do not allow backorders in this model, although they can be easily included.
- There are R resources whose binding capacity may potentially restrict system throughput, and any product may potentially encounter any number of those resources during a single time period (simultaneous resource possession is allowed).
- The planning horizon is T time periods, with period 0 representing the time period immediately prior to the beginning of the planning horizon.

Mathematically, the multilevel capacitated lot sizing problem (MLCLSP) can be stated as follows.

1.1.1. Index Sets.

$\mathcal{P}$—Products, $\mathcal{P} = \{p \mid p = 1, \ldots, P\}$.

$\mathcal{R}$—Resources, $\mathcal{R} = \{r \mid r = 1, \ldots, R\}$.

$\mathcal{S}_p$—Successors of product p in the BOM ($\mathcal{S}_p = \emptyset$ if p is an end item).

$\mathcal{T}$—Time periods, $\mathcal{T} = \{t \mid t = 0, 1, \ldots, T\}$.

1.1.2. Data.

a_{pr}—consumption of resource r to produce one unit of product p.

c_{rt}—amount of resource r available in time t.

d_{pt}—external demand for product p in period t.

f_{pr}—fixed consumption of resource r to process product p.

$\hat{h}_p$—initial inventory stock of product p.

$\hat{H}_p$—ending inventory stock of product p.

k_{pj}—number of units of product p to produce one unit of product j, where $j \in \mathcal{S}_p$.

s_{pr}—setup cost for product p on resource r.

v_{pr}—variable cost of processing one unit of product p on resource r.

w_{pt}—holding cost for product p in period t.

$\bar{x}_{pt}$—upper bound on production of product p in period t.

1.1.3. Decision Variables.

h_{pt}—inventory of product p at the end of period t $\forall(p \in \mathcal{P}, t \in \mathcal{T})$.

x_{pt}—production of product p during period t $\forall(p \in \mathcal{P}, t \in \mathcal{T})$.

y_{pt}—1 if $x_{pt} > 0$; 0 otherwise $\forall(p \in \mathcal{P}, t \in \mathcal{T})$.

1.1.4. Formulation.

(Problem MLCLSP)

$$\min \sum_{t\in\mathcal{T}} \sum_{p\in\mathcal{P}} \sum_{r\in\mathcal{R}} (v_{pr}x_{pt} + w_{pt}h_{pt} + s_{pr}y_{pt}) \tag{1}$$

subject to

$$h_{p(t-1)} + x_{pt} - h_{pt} = d_{p(t+1)} + \sum_{j\in\mathcal{S}_p} k_{pj}x_{j(t+1)}$$

$$\forall\ (p \in \mathcal{P}, t = 1, \ldots, T-1), \tag{2}$$

$$h_{p(T-1)} + x_{pT} = \hat{H}_p \quad \forall(p \in \mathcal{P}), \tag{3}$$

$$\sum_{p\in\mathcal{P}} (a_{pr}x_{pt} + f_{pr}y_{pt}) \le c_{rt} \quad \forall(r \in \mathcal{R}, t \in \mathcal{T}), \tag{4}$$

$$x_{pt} \le \bar{x}_{pt}y_{pt} \quad \forall(p \in \mathcal{P}, t \in \mathcal{T}), \tag{5}$$

$$x_{pt}, h_{pt} \ge 0 \quad \forall(p \in \mathcal{P}, t \in \mathcal{T}), \tag{6}$$

$$\hat{h}_p \le h_{p0} \le \hat{H}_p \quad \forall(p \in \mathcal{P}), \tag{7}$$

$$y_{pt} \in \{0, 1\} \quad \forall(p \in \mathcal{P}, t \in \mathcal{T}). \tag{8}$$

The objective function (1) minimizes the total production, setup, and inventory carrying cost over the planning horizon. Constraint set (2) guarantees flow for end items (when $\mathcal{S}_p = \emptyset$) and work in process. Constraint set (3) sets the minimum inventory level at the end of the planning horizon to support production in periods beyond the planning horizon. Constraint set (4) preserves capacity feasibility in each period. Constraint set (5) links the production and setup variables.

1.2. Background Literature

The literature on lot sizing is vast, and therefore an exhaustive overview here is neither feasible nor desirable. A recent overview of the lot sizing literature is given in Kuik et al. (1994). Bahl et al. (1987) and Maes et al. summarize efforts for all classes of lot sizing problems, while Harrison and Lewis (1996) provide an extensive review of lot sizing procedures for the multistage, multi-item case.

In this section we discuss a limited number of articles to allow the reader to position our work in relation to the literature. There are two fundamentally different approaches to solving MLCLSP. The first is the optimization approach, which seeks optimal solutions, and the second is the heuristic approach, which seeks good solutions and tests their quality empirically. Helber (1995) separates the heuristic procedures into three classes: decomposition approaches, stochastic local search procedures, and Lagrangian-based procedures. We add a fourth class to this list: LP-based procedures. These four heuristic classes are not mutually exclusive, and techniques can be used in concert effectively. For example, combinations of decomposition approaches with stochastic local search procedures are described in Helber (1995), and combinations of LP-based procedures with stochastic local search techniques are covered in Salomon (1991) and Kuik et al. (1993).

The optimization approaches are valuable because typically they involve methods for finding strong lower bounds. The research stream initiated by Eppen and Martin (1987) develops the shortest-route formulation of the single-stage capacitated lot sizing problem (CLSP), where the linear programming relaxation provides stronger lower bounds than the standard formulation of Billington et al. (1983). Tempelmeier and Helber (1994) extend this shortest-route formulation to MLCLSP. Pochet and Wolsey (1991) look at both uncapacitated and capacitated (single resource) multi-item problems having either a single-stage or a multistage general structure. They find near-optimal solutions by using the shortest-route formulation and adding valid inequalities that, in principle, can be automatically generated by a mathematical programming software package.

Some optimization approaches use Lagrangian relaxation to generate strong lower bounds. Diaby et al. (1992a) solve a single level problem with a single capacitated resource (labor), setup costs, and setup times. They use Lagrangian relaxation with subgradient optimization to generate lower bounds, followed by a branch-and-bound algorithm to locate the optimal solution. Their empirical tests on randomly generated problems show that capacity constraint relaxations produce better lower bounds than do demand constraints relaxations. Diaby et al. (1992b) use the same technique to locate near-optimal solutions to ''very large scale'' single-stage capacitated problems with setup times.

The first group of heuristic procedures, decomposition approaches, ignore the multilevel structure of MLCLSP and solve a sequence of CLSPs. Blackburn and Millen (1982) develop a decomposition procedure that accounts for the multilevel nature of the problem implicitly by modifying setup and holding costs. They consider uncapacitated pure assembly systems with constant demand, infinite time-horizon, and setup costs, and they solve these problems with an existing single-level lot sizing algorithm. Heinrich and Schneeweiss (1986) extend the method to general assembly systems, nonconstant demand, and finite time-horizon.

Alternatively, Tempelmeier and Helber (1994) focus on lot sizing with general assembly structures, multiple constrained resources, and nonzero setup costs (no setup times). They use the cost adjustment procedures of Blackburn and Millen (1982) and Heinrich and Schneeweiss (1986) to account for multiple production levels implicitly, and then use the heuristic technique first proposed by Dixon and Silver (1981) to solve a sequence of single-level capacitated problems. The final stage of the algorithm uses a smoothing heuristic to restore feasibility. The heuristic's performance was tested first on four sets of small problems, where optimal solutions were available, and the heuristic performed well relative to optimal solutions (reported deviations from optimality for the four problem sets were 4.87 percent, 3.79 percent, 2.21 percent, and 2.44 percent). The heuristic was also tested on a set of large problems, where optimal solutions were not available, and heuristic solutions were reported to be between 2 percent and 26 percent above strong lower bounds that were computed using an extension of the Eppen and Martin (1987) method.

Several heuristic approaches to lot sizing use Lagrangian relaxation. Billington et al. (1986) examine the multi-item, general assembly structure problem with setup costs, setup times, and a single constrained resource. They use a branch-and-bound strategy with Lagrangian relaxation to solve subproblems, followed by a smoothing procedure to eliminate infeasibilities.

Trigeiro et al. (1989) consider the single-level lot sizing problem with setup costs, setup times, and a single constrained resource. They use Lagrangian relaxation of the capacity constraints to decompose the problem into uncapacitated single-item problems, which are then solved by dynamic programming. A heuristic smoothing procedure is used as the last stage to construct feasible production plans.

Tempelmeier and Derstroff (1996) present another approach to the MLCLSP. They use Lagrangian relaxation of the multilevel inventory balancing constraints and capacity constraints, and update Lagrangian multipliers using subgradient optimization to compute lower bounds. Feasible solutions are constructed by a heuristic finite scheduling procedure that shifts production away from overloaded periods. The procedure is tested extensively using small and large problems, with and without setup times. The presence of setup times does not appear to affect the algorithm's performance. Solutions to small problems are, on average, very close to optimal (reported deviation from optimality is 1.47 percent for problems without setup times and 1.35 percent for problems with setup times), while solutions to large problems are compared to tight lower bounds computed by Lagrangian relaxation (reported deviation from lower bound in percentage of upper bound is about 17 percent for problems without setup times and 16.5 percent for problems with setup times).

Stochastic local search procedures, the third class of heuristic solution methods, include simulated annealing, tabu search, genetic algorithms, and evolution strategies. Salomon (1991) and Helber (1995) overview and compare some of these methods, and Kuik and Salomon (1990) provide a detailed discussion of simulated annealing-based methods. Helber (1995) compares methods involving decomposition, stochastic local search, decomposition and stochastic local search, and the Lagrangian based procedure of Tempelmeier and Derstroff (1996). He finds that procedures using simulated annealing locate solutions with the best average quality, but require so much computational effort that they could be tested only on problem sets containing small-scale problems. A procedure combining evolution strategy with decomposition performed the best on the medium-sized problems. Because of the amount of computational effort required, the only two methods that could be used on large-scale problems were pure decomposition and the Tempelmeier and Derstroff heuristic. The Tempelmeier and Derstroff heuristic, on average, used the least CPU time in all three problem sets (small, medium, and large-scaled) while locating solutions that were, on average, almost as good as or better than solutions located by the pure decomposition algorithm.

The fourth group of heuristic approaches involves various LP relaxations of a mixed integer programming formulation of the MLCLSP. Maes et al. (1991) present three LP-based heuristics for solving multi-item, multistage lot sizing problems with multiple constrained resources, setup costs, and no setup times. These three heuristics start with a solution to the LP relaxation of the problem and solve sequences of LP restrictions until an integer solution is found. Heuristic performance is tested extensively via small random test problems with serial assembly requirements and the procedures are found to perform quite well. There is, however, a tradeoff between the quality of solutions and the computational effort, for the best performing heuristic requires branch-and-bound as the last step of the algorithm. Salomon (1991) and Kuik et al. (1993) compare the performance of these heuristics to methods using simulated annealing and tabu search, using randomly generated problems. They find that simulated annealing and tabu search perform better than pure LP-based heuristics, but the performance of these heuristics can be improved by incorporating some elements of tabusearch and simulated annealing into them.

Harrison and Lewis (1996) describe the Coefficient Modification Heuristic (CMH), which finds excellent solutions to multi-item, multistage lot sizing problems with multiple constrained resources and serial assembly systems. The CMH is specifically designed to handle setups that consume capacity (setup times) and assumes that the setup costs are small enough to be ignored. This algorithm exploits the model's special structure by repeatedly solving small LP restrictions of the original problem. It implicitly accounts for the capacity consumed in setups by modifying the technological coefficients of certain variables in these restrictions. Solution quality is evaluated using Optimization Subroutine Library (OSL) (IBM 1991) by first comparing heuristic and optimal solutions for a set of small problems. The heuristic finds solutions that are extremely close to optimal in small fractions of the time required by OSL. The heuristic is also tested on sets of medium-sized problems, and it finds better quality solutions than time-truncated OSL runs with far less computational effort.

In this paper we extend the Coefficient Modification Heuristic of Harrison and Lewis to handle setup costs and general assembly structures. The heuristic consists of two stages. In a manner similar to Harrison and Lewis, the first stage solves a sequence of related linear programming problems, iteratively modifying technological and objective function coefficients to generate a good initial solution. The second stage solves a sequence of restrictions of the linear programming relaxation of the MLCLSP to improve first-stage solution quality while preserving feasibility. We outline the development of this solution framework in §2. In §3 we evaluate the heuristic's performance with two sets of randomly generated test problems. We first compare our method's solutions to optimal plans for small problems to gain an understanding of heuristic solution quality relative to optimal. We also test our algorithm on medium-scale problems to evaluate algorithm performance in more realistic situations. In this latter case we do not compare heuristic and optimal solutions, since the medium-scale problems are too difficult to solve to optimality. Instead, we compare heuristic plans to time-truncated Optimization Subroutine Library (OSL) solutions.

2. The Heuristic Algorithm

2.1. Motivation

In this section we formally describe the Coefficient Modification Heuristic with Cost Balancing and Setup Reduction (CMHBR). This algorithm, just as the CMH in Harrison and Lewis, utilizes the fact that each binary variable in MLCLSP is linked to a continuous variable in such a way that either both must be positive or zero. This special structure, discussed in detail in §2.2, guarantees that if we fix the binary variables at particular values, the resulting restriction of MLCLSP is an LP that has fewer variables and constraints than MLCLSP.

The first stage of the algorithm is a version of CMH, modified to account for the objective function values of the setup variables and to determine solutions in which the total setup cost is close to the total inventory carrying cost. We know that the uncapacitated single stage, single product, constant demand rate problem has the property that the total inventory cost equals the total setup cost at optimality (see Silver and Peterson 1985). Although this property does not hold in more complex cases, such as this one, our computational experience has shown that in many test problems the "balanced" solutions have lower objective function values than "unbalanced" solutions. We incorporate this observation into our algorithm.

The second stage of the algorithm improves the first stage solution by solving a series of LP restrictions of the original problem. If feasibility is lost at any point during the second stage, the algorithm attempts to restore it, and if unable to do so, reverts to the best feasible solution found. Therefore, once a feasible solution is found at the end of the first stage, final solution feasibility is guaranteed.

2.2. Reformulation of the Problem

The MLCLSP has two types of continuous variables—a set with corresponding binary variables linked to them and a set without linked binary variables—and three types of constraints. The first constraint block contains equations with both continuous and binary variables, the second block contains constraints with only continuous variables, and the third block contains linking constraints. We note that our heuristic procedure is not restricted to MLCLSP, but can be applied to a wider class of problems with this special structure. The reader should note, however, that in the framework of the MLCLSP, the set of continuous variables with corresponding binary variables includes the production variables x_{pt}, and the set of continuous without corresponding binary variables includes the inventory variables h_{pt}.

Similarly, in the context of the MLCLSP, the block of constraints with both continuous and binary variables includes capacity constraints (4), the block of equations with only continuous variables includes inventory related constraints (2) and (3), and the block of linking constraints includes Equation (5).

We adopt the notation of Harrison and Lewis to represent the problem MLCLSP by the following general formulation.

2.2.1. Index Sets.

$\mathcal{I}^1$—Index set of general constraints that contain both continuous variables and binary variables $\mathcal{I}^1 = \{i \mid i = 1, \dots, I^1\}$ (Equation (4) in MLCLSP).

$\mathcal{I}^2$—Index set of general constraints that contain only continuous variables $\mathcal{I}^2 = \{i \mid i = I^1 + 1, \dots, I^1 + I^2\}$ (Equations (2) and (3) in MLCLSP).

$\mathcal{I}^3$—Index set of linking constraints between continuous variables and binary variables $\mathcal{I}^3 = \{i \mid i = I^1 + I^2 + 1, \dots, I^1 + I^2 + I^3\}$ (Equation (5) in MLCLSP).

$\mathcal{I}$—Index set of all constraints $\mathcal{I} = I^1 \cup I^2 \cup I^3$.

$\mathcal{J}$—Index set of all continuous variables $\mathcal{J} = \{j \mid j = 1, \dots, J\}$ (production and inventory variables x_{pt} and h_{pt} in MLCLSP).

$\mathcal{J}_y$—Index set of continuous variables that have a corresponding binary (setup) variable $\mathcal{J}_y \subseteq \mathcal{J}$ (production variables x_{pt} in MLCLSP). Note that since all binary variables in the model are linked to continuous variables, $\mathcal{J}_y$ also serves as the index set for binary variables.

2.2.2. Data.

a_{ij}^{kl}—coefficient of the ith constraint for the jth variable in block kl.

c_j^l—objective function coefficient for variable j, $j \in \mathcal{J}$ in block l.

b_i^k—right-hand side of the ith constraint for block k.

$\bar{x}_j$—upper bound for variables j, $j \in \mathcal{J}$.

2.2.3. Decision Variables.

x_j—continuous variables, $j \in \mathcal{J}$.

y_j—binary (setup) variables, $j \in \mathcal{J}_y$.

2.2.4. Formulation.

$$\min \sum_{j \in \mathcal{J}} c_j^1 x_j + \sum_{j \in \mathcal{J}_y} c_j^2 y_j \tag{9}$$

subject to

$$\sum_{j \in \mathcal{J}} a_{ij}^{11} x_j + \sum_{j \in \mathcal{J}_y} a_{ij}^{12} y_j = b_i^1, \quad i \in \mathcal{I}^1, \tag{10}$$

$$\sum_{j \in \mathcal{J}} a_{ij}^{21} x_j = b_i^2, \quad i \in \mathcal{I}^2, \tag{11}$$

$$x_j - \bar{x}_j y_j \le 0, \quad j \in \mathcal{J}_y, \tag{12}$$

$$0 \le x_j \le \bar{x}_j, \quad j \in \mathcal{J}, \tag{13}$$

$$y_j \in \{0, 1\}, \quad j \in \mathcal{J}_y. \tag{14}$$

This structure, represented graphically in Figure 1, is not limited to the lot sizing problem of §1.2.

2.3. Details of the Algorithm

Harrison and Lewis (1996) note that if we fix the setup variables at $y_j = \hat{y}_j$ in Problem MLCLSP, we can form the following LP restriction:

$$\min \sum_{j \in \mathcal{J}} c_j^1 x_j + \sum_{j \in \mathcal{J}_y} c_j^2 \hat{y}_j \tag{15}$$

subject to

$$\sum_{j \in \mathcal{J}} a_{ij}^{11} x_j + \sum_{j \in \mathcal{J}_y} a_{ij}^{12} \hat{y}_j = b_i^1, \quad i \in \mathcal{I}^1, \tag{16}$$

$$\sum_{j \in \mathcal{J}} a_{ij}^{21} x_j = b_i^2, \quad i \in \mathcal{I}^2, \tag{17}$$

$$0 \le x_j \le \bar{x}_j, \quad j \in \mathcal{J}. \tag{18}$$

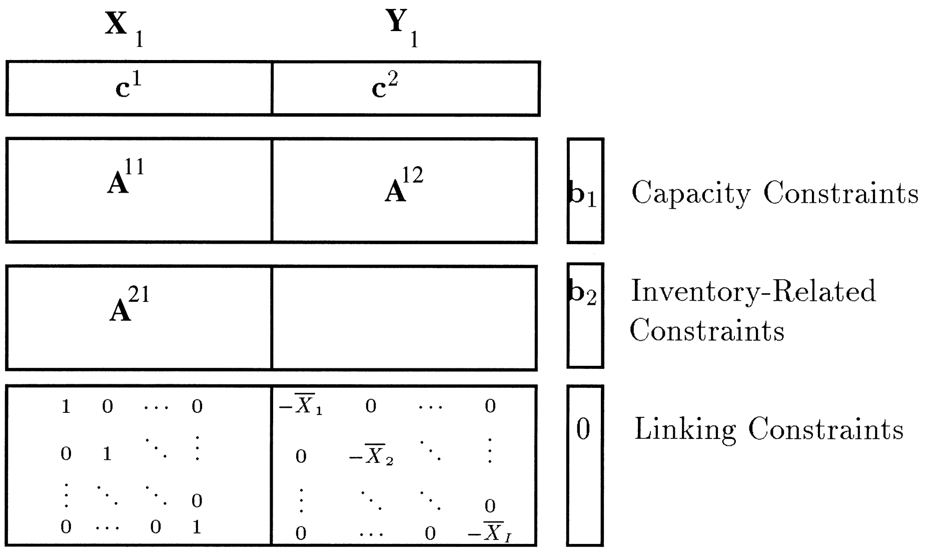
This restriction is much smaller than MLCLSP because it contains no binary variables or linking constraints, and since the second term in the objective function is constant, it can be ignored during model solution.

We implicitly approximate the impact of setups in a method parallel to Harrison and Lewis. For any value of x_j we modify the technological coefficients a_{ij}^{11} to account for the resource consumption during setups (setup times) while simultaneously modifying the objective function coefficients c_j^1 to approximate the impact of the setups on the objective function (setup costs). The next section presents the details of exactly how these coefficients are modified. If we call the modified technological coefficients $\tilde{a}_{ij}^{11}$ and the modified objective function coefficients $\tilde{c}_j^1$, the following formulation (LPR) approximates the impact of setups in MLCLSP.
(Problem LPR)

$$\min \sum_{j \in \mathcal{J}} \tilde{c}_j^1 x_j \tag{19}$$

subject to

Figure 1 **Problem Structure**



$$\sum_{j\in\mathcal{J}} \tilde{a}_{ij}^{11}\, x_j = b_i^1, \quad i \in \mathcal{I}^1, \tag{20}$$

$$\sum_{j\in\mathcal{J}} a_{ij}^{21} x_j = b_i^2, \quad i \in \mathcal{I}^2, \tag{21}$$

$$0 \le x_j \le \bar{x}_j, \quad j \in \mathcal{J}. \tag{22}$$

2.3.1. The Coefficient Modification Subroutine with Cost Balancing (CMSB). Since MLCLSP, unlike the problem posed in Harrison and Lewis, contains coefficients c_j^2, simply solving a sequence of LPR instances with modified coefficients underestimates the impact of setups and consequently yields suboptimal solutions. Therefore, the basic modification procedure outlined in the preceding paragraph must be adjusted to "push" the LPR solutions toward having the "right" number of setups.

The Coefficient Modification Subroutine with Cost Balancing (CMSB) attempts to anticipate which continuous variables, and hence corresponding binary variables, should be basic in the final solution by repeatedly solving instances of LPR, starting with an initial value of $\hat{y}_j = 0$. The algorithm assumes that the temporarily determined level of a continuous variable is optimal, and proportionally allocates the fixed setup times and costs over the continuous variables. It then further modifies the objective function coefficients to generate a balanced solution, as discussed in §2.1.

Each time LPR is solved, the coefficients of x_j in $\mathcal{I}^1$ are modified to reflect the values of x_j and $\hat{y}_j$. Each constraint coefficient is eligible to be modified at most once while each objective function coefficient is eligible to be modified at most twice. These parameters proved to be most successful in our empirical tests. When we allowed each objective function coefficient to be modified only once, the algorithm was unable to locate a feasible solution in many of the test cases. Conversely, allowing the coefficients to be modified more times increased the computational burden without consistently yielding better quality solutions.[1]

We introduce additional notation to describe the CMSB algorithm in greater detail. Let

$$\mathbf{x}_{(k)}^* = \begin{bmatrix} x_{1(k)}^* \\ \vdots \\ x_{J(k)}^* \end{bmatrix}$$

be the optimal solution to the kth subproblem. We can

[1] In 11 instances of our test problems (7 instances of problem 7 and 4 instances of problem 11) the CMSB was unable to locate a feasible solution given those default settings. After we allowed each constraint coefficient to be modified twice and each objective function coefficient to be modified four times, feasible solutions were located.

determine the total setup cost and the total inventory carrying costs[2] associated with $\mathbf{x}^*_{(k)}$. Let

$$\text{Cost } H = \sum_{j\in\mathcal{J}} c_j^1 x^*_{j(k)}$$

be the total inventory carrying cost associated with $\mathbf{x}^*_{(k)}$. Also let

$$\text{Cost } S = \sum_{j\in\mathcal{J}_y, x^*_{j(k)}\neq 0} c_j^2$$

be the total setup cost associated with $\mathbf{x}^*_{(k)}$. We can now define a function $\epsilon(\mathbf{x}^*_{(k)})$ that captures the relationship between the setup cost and the inventory carrying cost in such a way that the more unbalanced the solution, the higher the value of $\epsilon(\mathbf{x}^*_{(k)})$. For example, let

$$\epsilon_{(k)} = \epsilon(\mathbf{x}^*_{(k)}) = \frac{|\text{Cost } H - \text{Cost } S|}{\text{Cost } H + \text{Cost } S}$$

be the solution "imbalance" (the numerator) as a percentage of the total cost.[3]

Algorithm 1 provides a formal description of the Coefficient Modification Subroutine with Cost Balancing, using the following notation.

k: cumulative number of LPR subproblems solved.

$k_{\max}$: maximum number of LPR subproblems to be solved (fixed externally to a large number—$k_{\max} = 10^{31}$ in our case).

mod $A(j)$: the number of times coefficient a_{ij}^{11} was modified.

mod $C(j)$: the number of times coefficient c_j^1 was modified.

m: number of variables that have had their coefficients modified in the current iteration.

Algorithm 1 The Coefficient Modification Subroutine with Cost Balancing (CMSB)

```
1. $m \leftarrow 1$, mod$A(j) \leftarrow 0, \forall j \in \mathcal{J}_y$, mod$C(j) \leftarrow 0, \forall j \in \mathcal{J}_y$, $\epsilon_{(0)} \leftarrow 1$, $k \leftarrow 0$.
2. while ($m \neq 0$) and ($k \leq k_{\max}$) do
       solve LPR$_k$
       $m \leftarrow 0$
       forall $i \in \mathscr{I}^1$ and $j \in \mathcal{J}_y \ni x_j > 0$ and mod$A(j) < 1$ and mod$C(j) < 2$
           $\tilde{a}_{ij}^{11} \leftarrow \frac{a_{ij}^{11}\times\infty^*_{j(k)} + a_{ij}^{12}}{\infty^*_{j(k)}}$ ; $\tilde{c}_j^1 \leftarrow \frac{c_j^1\times\infty^*_{j(k)} + c_j^2}{\infty^*_{j(k)}}$
           if ($CostH > CostS$) then $\tilde{c}_j^1 = \tilde{c}_j^1(1-\epsilon_{(k)})^{\text{mod}C(j)}$;
           else $\tilde{c}_j^1 = \tilde{c}_j^1(1+\epsilon_{(k)})^{\text{mod}C(j)}$
           mod$A(j) \leftarrow$ mod$A(j) + 1$; mod$C(j) \leftarrow$ mod$C(j) + 1$; $m \leftarrow m + 1$
       endfor
       forall $j \notin \mathcal{J}_y$ and mod$C(j) < 2$
           if ($CostH > CostS$) then $\tilde{c}_j^1 = \tilde{c}_j^1(1+\epsilon_{(k)})^{\text{mod}C(j)}$;
           else $\tilde{c}_j^1 = \tilde{c}_j^1(1-\epsilon_{(k)})^{\text{mod}C(j)}$
           mod$C(j) \leftarrow$ mod$C(j) + 1$; $m \leftarrow m + 1$
       endfor
       $k \leftarrow k + 1$
   endwhile
3. forall $i \in \mathscr{I}^1$ and $j \in \mathcal{J}$
       $\tilde{a}_{ij}^{11} \leftarrow a_{ij}^{11}$; $\tilde{b}_i^1 \leftarrow b_i^1 - \sum_{j\in\mathcal{J}_y} a_{ij}^{12}\hat{y}_j$
       if $x^*_{j(k)} = 0$ then $\bar{x}_j = 0$
   endfor
   Solve LPR
   if LPR is infeasible STOP
```

Step 1 initializes all the variables and ensures that Step 2 is executed at least once.

Step 2 repeatedly solves the restricted version of MLCLSP, modifying the coefficients of nonzero decision variables to account for setups and to balance setup and holding costs. CMSB starts with a complete relaxation of the linking constraints. The first time LPR is solved, the solution completely ignores the setups. Subsequently, the modified coefficients $\tilde{a}_{ij}^{11}$ and $\tilde{c}_j^1$ implicitly take setups into account by allocating the fixed setup costs and times across the variable production costs and the variable processing times. Since the implied values of the setup variables change for each iteration of Step 2 (for each k), at the end of Step 2 a feasible solution is likely to emerge if the adjusted setup times adequately reflect the true loss of capacity due to setups. The mech-

[2] To provide intuition, we refer to the portion of the objective function associated with binary variables as the true "setup cost" and to the portion associated with continuous variables as the true "holding cost." These labels are accurate in the framework of the MLCLSP, but the reader should keep in mind that these costs have a different interpretation in other applications.

[3] Our experience is that the exact functional form of ϵ is not very important, so long as $\epsilon = 0$ when Cost H = Cost S, and $\epsilon \geq 1$ when Cost $H = 0$ or Cost $S = 0$. We solved our test problems with several different functional forms of ϵ, and there was no consistent difference in solution quality. Algorithm 1 and the results we report in §3 use the functional form of ϵ noted above.

anism for locating feasible solution is the same one used in Harrison and Lewis (1996). A feasible solution at this point is not guaranteed. Step 2 continues until either no coefficients are left to be modified ($\mathrm{mod}A(j) = 1\ \forall j$ and $\mathrm{mod}C(j) = 2\ \forall j$), or the maximum allowable number of LPR subproblems has been solved ($k = k_{\max}$).

Step 3 checks feasibility and attempts to regain it, if necessary. It restores the original coefficients a_{ij}^{11}, adjusting the RHSs of Equation (20) to reflect the entire capacity loss due to setups, fixes all the nonbasic variables at their lower bounds, and by solving LPR attempts to switch production among basic variables.

Step 3 does not guarantee a feasible solution (recall that the feasibility problem of CLSP is NP-complete). If a feasible solution cannot be located at this point, the algorithm terminates without a feasible solution. Nevertheless, our algorithm has located a feasible solution to each of our test problems.

2.3.2. Simple Setup Reduction Subroutine (SSR). The purpose of CMSB is twofold. First, it attempts to find a feasible solution to MLCLSP. Second, as explained in §2.1, it tries to find a solution where the total setup cost and the total inventory carrying cost are close. The second part does not guarantee a good solution, however, since the optimal solution to a capacitated problem may be extremely unbalanced (as was true for several of our test problems). Since the CMSB alone may be unable to generate high quality solutions, our algorithm includes a second stage to refine CMSB solutions.

The first portion of the refinement process is the Simple Setup Reduction Subroutine (SSR). This subroutine sequentially fixes each basic variable (x_k) at its lower bound (0), thereby eliminating a setup, and solves the resulting LP restriction. Of course, when the resulting LP is solved, some of the nonbasic variables may become basic and therefore add setups in different periods. Let us call this LP restriction Problem LPRR$_k$. It is identical to Problem LPR with the additional constraint that a single variable (x_k) is fixed at 0. We keep the modified coefficients $\tilde{c}_j^1$ and $\tilde{a}_{ij}^{11}$, generated by CMSB, in place of the original coefficients c_j^1 and a_{ij}^{11} because SSR determines whether the CMSB solution can be improved by eliminating a single setup and placing no further restrictions on the problem. Since the CMSB calculated modified coefficients $\tilde{c}_j^1$ and $\tilde{a}_{ij}^{11}$ to account implicitly for setups, we preserve these coefficients in the SSR to ensure that solutions SSR generates also account for setups.

(Problem LPRR$_k$)

$$\min \sum_{j\in\mathcal{J}} \tilde{c}_j^1 x_j \tag{23}$$

subject to

$$\sum_{j\in\mathcal{J}} \tilde{a}_{ij}^{11} x_j = b_i^1, \quad i \in \mathcal{I}^1, \tag{24}$$

$$\sum_{j\in\mathcal{J}} a_{ij}^{21} x_j = b_i^2, \quad i \in \mathcal{I}^2, \tag{25}$$

$$0 \le x_j \le \bar{x}_j, \quad j \in \mathcal{J}, \tag{26}$$

$$x_{(k)} = 0. \tag{27}$$

Algorithm 2 Simple Setup Reduction Subroutine (SSR)

```
forall $k \in \mathcal{J}_y \ni x_k > 0$
    solve LPRR$_k$
    if better solution found save the values of the
    decision variables in the new solution
endfor
```

Algorithm 2 summarizes the Simple Setup Reduction subroutine. The subroutine starts with the solution generated by CMSB. It then sequentially fixes to a value of 0, each continuous basic variable with a corresponding binary variable, and solves the corresponding LPRR$_k$. If the SSR subroutine finds a solution superior to the best one it has located so far, it saves the values of the decision variables in that solution and uses them in subsequent subproblems.

2.3.3. Restricted Setup and Inventory Reduction Subroutines (RSR) and (RIR). The SSR is a very simple method to eliminate setups—simply remove one setup at a time, and see if a better solution emerges. In problem LPRR$_k$, only a single variable (x_k) is forced to its lower bound. The purpose of the Restricted Setup Reduction (RSR) and the Restricted Inventory Reduction (RIR) subroutines is to fine-tune an existing solution further by switching production among several adjacent variables. Both routines start by fixing all nonbasic variables at their lower bounds. Then, the RSR subroutine sequentially forces each basic variable (x_k) to its lower bound (0), eliminating a setup, while releasing the variables on both sides of x_k[4] (even nonbasic variables). Solving the resulting LP restriction determines whether switching the production represented by the variable x_k to one of the earlier or later periods yields a better solution. The RIR subroutine sequentially releases each nonbasic variable (x_k) and the variables on both sides of it and solves the resulting LP restriction. This procedure permits x_k to become basic, in case doing so will yield a better solution.

The RSR and RIR are similar to SSR, except that many more variables are fixed at their lower bounds in each subproblem. The only nonbasic variables in the current solution not fixed at their lower bounds are the variables adjacent to x_k (and x_k itself in RIR). This set of fixed variables almost determines the basis and makes it possible to restore the original coefficients c_j^1 and a_{ij}^{11}. Since these original coefficients represent the true costs and the true resource consumption, we expect restoring them to yield superior solutions. In order to account for the capacity consumed by the setups we modify the right-hand sides to account for them explicitly by letting

$$\tilde{b}_i^1 = b_i^1 - \sum_{j \in \mathcal{J}_y} a_{ij}^{12} \hat{y}_j.$$

For every k, we define a new set,[5]

$$\mathcal{M}_k = \{m \mid m \in \mathcal{J}_y \ni x_m = 0\} - \{k - \alpha, k - \alpha + 1, \ldots, k - 1, k, k + 1, \ldots, k + \alpha\},$$

containing all nonbasic variables, except those within α range to x_k. In the RSR subroutine, where x_k is basic, we also add x_k to $\mathcal{M}_K$. In solving the test problems we used $\alpha = 3$, but solution quality does not seem to be very sensitive to the value of α, as long as α is small. In fact, using $\alpha = 2$ or $\alpha = 4$ did not affect the solution quality in our test problems. It is important, however, for α to be small, even in large problems.[6] Large α releases many variables, making problem LPRR($\mathcal{M}_K$) very similar to problem LPRR$_k$ and making RSR very similar to SSR.

The model solved in the RSR and RIR subroutine is: (Problem LPRR($\mathcal{M}_K$)

$$\min \sum_{j \in \mathcal{J}} \tilde{c}_j^1 x_j$$

subject to

$$\sum_{j \in \mathcal{J}} a_{ij}^{11} x_j = \tilde{b}_i^1, \quad i \in \mathcal{I}^1,$$

$$\sum_{j \in \mathcal{J}} a_{ij}^{21} x_j = b_i^2, \quad i \in \mathcal{I}^2,$$

$$0 \le x_j \le \bar{x}_j, \quad j \in \mathcal{J},$$

$$x_k = 0, \quad m \in \mathcal{M}_k.$$

LPRR($\mathcal{M}_K$) differs from LPRR$_k$ in three ways. First, the modified coefficients are restored to their original values. Second, the right-hand sides are modified to account for setups explicitly. Finally, more variables are fixed at their lower bounds. Algorithms 3 and 4 formally present the RSR and the RIR subroutines.

Algorithm 3 Restricted Setup Reduction (RSR)

```
forall $k \in \mathcal{J}_y \ni x_k > 0$
    $\mathcal{M}_k = \mathcal{M}_k \cup k$
    solve LPRR($\mathcal{M}_k$)
    if better solution found save the values of the
    decision variables in the new solution
    $\mathcal{M}_k = \mathcal{M}_k - k$
endfor
```

Algorithm 4 Restricted Inventory Reduction (RIR)

```
forall $k \in \mathcal{J}_y \ni x_k = 0$
    solve LPRR($\mathcal{M}_k$)
    if better solution found save the values of the
    decision variables in the new solution
endfor
```

2.3.4. Restoring Feasibility.

Algorithm 5 Restoring Feasibility Subroutine (RF)

```
forall $j \in \mathcal{J}_y \ni x_k = 0$
    $\bar{x}_j = 0$
endfor
solve LPR
```

[4] The phrase "on both sides of x_k" assumes an order in the set $\mathcal{J}$. What constitutes a reasonable order may differ in different applications. Our test problems are in the context of MLCLSP, and here adjacent variables are variables related to the same product in adjacent periods. Intuitively, this order should be important to the effectiveness of RSR and RIR subroutines.

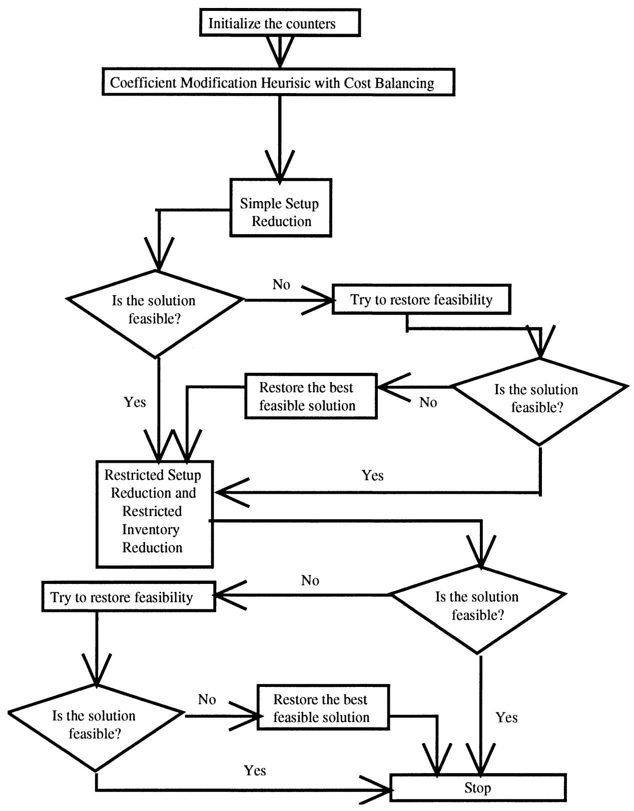
[5] For $k < \alpha$, $\mathcal{M}_k = \{m \mid m \in \mathcal{J}_y \ni x_m = 0\} - \{1, 2, \ldots, k - 1, k, k + 1, \ldots, k + \alpha\}$. For $k > J_y - \alpha$, $\mathcal{M}_k = \{m \mid m \in \mathcal{J}_y \ni x_m = 0\} - \{k - \alpha, k - \alpha + 1, \ldots, k - 1, k, k + 1 \cdots, J_y\}$, where J_y is the cardinality of $\mathcal{J}_y$.

[6] We used the same value of α in our small test problems and our medium-sized test problems.

The final part of the algorithm's second stage is very similar to Step 3 of Algorithm 1. It works to restore feasibility if lost during the SSR, RIR, or RSR subroutines, and Algorithm 5 describes these steps. This procedure restores the technological coefficients to their original values, adjusts the RHSs to account to setups, fixes all the nonbasic variables at 0, and solves LPR. If this strategy fails to restore feasibility, the routine terminates with the best feasible solution found so far.

2.3.5. Summary: The Entire Algorithm. Figure 2 summarizes the entire algorithm—the Coefficient Modification Heuristic with Cost Balancing and Setup Reductions (CMHBR). The CMSB subroutine provides an initial feasible solution. After the CMSB subroutine, the algorithm performs the SSR subroutine, checks feasibility, and restores feasibility if necessary. The last part of the algorithm fine-tunes the solution with a sequence of RSR and RIR subroutines.

Figure 2 Flow Chart


Initialize the counters
Coefficient Modification Heurisic with Cost Balancing
Simple Setup Reduction
Is the solution feasible?
No
Try to restore feasibility
Is the solution feasible?
No
Restore the best feasible solution
Yes
Yes
Restricted Setup Reduction and Restricted Inventory Reduction
Is the solution feasible?
No
Try to restore feasibility
Is the solution feasible?
No
Restore the best feasible solution
Yes
Yes
Stop

3. Computational Results

In this section we test the performance of CMHBR relative to OSL Version 1.2 (IBM 1991). OSL is a commercially available code, and examining our algorithm's effectiveness relative to OSL provides good benchmarks. First, we compare CMHBR and OSL performance via a series of small-scale random problems, which are solved to optimality using OSL with default settings, to see how CMHBR performs in a variety of environments. We then focus on medium-scale random instances to assess CMHBR performance in more realistically sized cases. All runs were executed on an IBM RS/6000 under IBM AIX FORTRAN Version 2.3.0 Compiler/6000 with compiler optimization level 3. We used OSL for solving all LP restrictions in the CMHBR algorithm.

3.1. Small-Scale Random Test Problems

We designed the first group of test problems to provide known benchmarks for evaluating CMHBR performance in a variety of environments. Twelve basic production planning problems were generated. In each case two end-products must be assembled from various groupings of intermediate items with part commonality over a four- to six-period planning horizon. Figure 3 summarizes the randomly generated bills of material (BOMs) for these problems. One unit of any product is required to manufacture this product's successor. Demand for the end-items varies uniformly from 1 to 10 units per period while external demand for subassemblies is prohibited. Initial finished goods and work-in-process inventories range from 1 to 10 units and from 5 to 15 units, respectively, and ending stock positions are required to match the initial ones to support production requirements in periods beyond the planning horizon. No backorders are allowed. Inventory holding costs range from 1 to 100 per unit for all items. Two resources limit system throughput. We show the assignments of resources to products in the last two columns of Table 1. Note that both resources are needed to process at least 1 product in 8 of the 12 problems.

Generally, three parameters affect the difficulty of a production planning problem: cumulative capacity usage, the ratio of setup times to processing times, and the ratio of setup costs to inventory carrying costs. In this experiment we systematically vary these three problem characteristics for the twelve groups of test cases to

Figure 3 BOM Structures: Small Problems

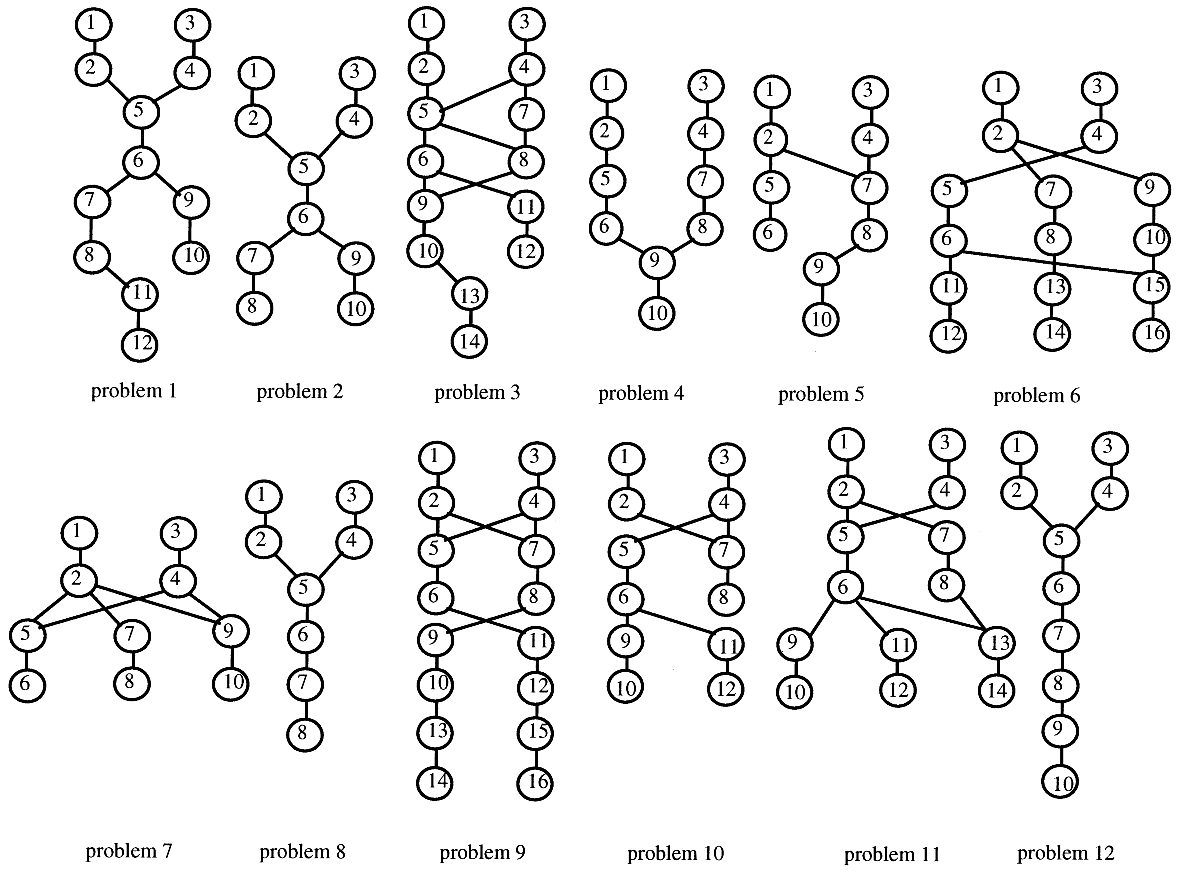


Table 1 Small Random Test Problems Characteristics: Products and Resources

Problem Number	Number of Periods	Per Unit Average End Item Demand	Per Unit Average Intermediate Item Demand	Assignment of Resources to Products: Resource 1	Assignment of Resources to Products: Resource 2
S1	4	5.2	13.1	6, 7, 9, 12	1, 4, 8, 12
S2	4	5.4	14.4	1, 3, 6, 10	2, 8, 9
S3	4	5.2	12.6	4, 5, 7, 14	2, 4, 6, 9, 12
S4	5	5.2	10.3	1, 3, 5, 8	3, 9
S5	6	6.0	14.0	2, 3, 6	3, 7, 10
S6	4	4.6	6.7	6, 16	2, 3, 8, 10, 11, 14
S7	6	5.6	13.2	2, 3, 10	3, 5, 7
S8	5	5.2	15.5	1, 6	1, 3, 6, 8
S9	4	4.6	8.7	9, 11, 15	2, 4, 5, 7, 10, 14, 15
S10	5	5.8	9.5	3, 5, 7, 11	2, 9
S11	4	5.2	12.6	1, 11, 14	3, 5, 8, 9
S12	5	5.7	15.1	8	2, 3, 6, 8, 10

Table 2 Highly Capacitated Small Problems Characteristics

	Capacity Needed for Setups (Percentage)		Capacity Needed for Production (Percentage)		Lot for Lot Solution Feasible?	
Problem Number	10	50	10	50	10	50
S1	48.36	79.58	46.64	15.42	N	Y
S2	66.87	87.53	28.13	7.47	N	N
S3	50.88	80.69	44.12	14.31	N	Y
S4	59.96	84.99	35.04	10.01	N	Y
S5	54.73	81.65	40.27	13.35	N	N
S6	63.25	86.23	31.75	8.77	N	Y
S7	50.53	80.76	44.47	14.24	N	N
S8	57.38	83.94	37.62	11.06	N	Y
S9	60.90	85.43	34.10	9.57	N	Y
S10	60.34	85.10	34.66	9.90	N	N
S11	47.39	78.42	47.61	16.58	N	N
S12	40.19	73.16	54.81	21.84	N	N
Average	55.06	82.29	39.94	12.71		

create specific instances for analysis. First, the underlying production system is either capacitated (with the cumulative capacity usage of 95 percent) or uncapacitated (with the cumulative capacity usage of 50 percent). We define *cumulative capacity usage* as in Trigeiro et al. (1989), by dividing the sum of the cumulative production requirements and the cumulative lot-for-lot setup requirements by the cumulative capacity.[7] Although the cumulative capacity usage of 95 percent may not fit everyone's definition of ''highly capacitated,'' our results do provide a useful benchmark.

Second, the range for the randomly determined setup times is 10 times the processing time range in the short setup treatments (10), while the setup times in the long setup treatments (50) are 5 times longer than the short setup treatments' setup times. Finally, the ratio of setup costs to holding costs is either low or high, where the range of setup costs in the high setting is 10 times that in the low setting. Setup costs range from 500 to 1000 under low setup cost specifications and from 5000 to 10000 in high setup costs treatments. Given these characteristics, 8 specific treatments are defined for each of the 12 basic production planning cases, for a total of 96 individual lot sizing problems.

To complement the information in Figure 3, we summarize the relevant characteristics for the twelve basic test cases under analysis in Table 1. We provide some basic information about the problem structures by noting the number of time periods, the per-unit average end-item demand per period, and per unit average intermediate item demand per period. We also show the assignment of resources to products. Table 2 provides additional information about highly capacitated problems. It separates the problems into treatments with short setup times (10) and long setup times (50), and shows the percentage of capacity needed for setups in every period with demand, the percentage of capacity needed to satisfy demand, and whether the lot-for-lot solution is feasible. Note that the lot-for-lot solution is only feasible in 25 percent of the problems. Table 3 summarizes characteristics of the resulting mixed integer models.

These 12 basic problem instances represent the first 12 test cases for which the MIP formulation in §1.1 could be solved to optimality in no more than 5000 CPU seconds, for all 8 treatments, using OSL with default settings. Each of the 96 problems was solved with the CMHBR and with OSL. We report capacity usage information for the capacitated problems (separated by the

Table 3 Small Random Test Problems Model Characteristics

Problem Number	Number of Rows	Number of Columns	Number of Binary Variables	Density (Percent)
S1	112	138	28	2.16
S2	99	121	28	2.43
S3	129	159	32	1.90
S4	108	140	25	2.18
S5	125	163	26	1.88
S6	142	176	32	1.65
S7	125	167	30	1.91
S8	88	114	20	2.76
S9	146	180	36	1.66
S10	128	166	30	1.80
S11	125	155	28	1.97
S12	108	140	25	2.18

[7] Trigeiro et al. (1989) show that ''even if the cumulative lot-for-lot capacity usage (including setup time) does not exceed the cumulative capacity available, a feasible solution may not exist,'' and therefore the lot-for-lot solution may not be feasible. For example, the lot for lot solutions are feasible in only 25 percent of our capacitated test problems (see Table 2).

Table 4 Capacitated Small Problems—Capacity Usage

Problem Number	Effective Utilization Mean (Percentage) 10	Effective Utilization Mean (Percentage) 50	Average Number of Setups (Percentage)	Percentage of Periods With Full Capacity Usage	$\frac{UB - LB}{UB}$ (Percentage)	Mean Time Between Orders (In Periods)
S1	47.27	42.39	42.2	0.00	63.8	2.6
S2	48.31	43.30	42.7	9.38	57.4	2.8
S3	51.11	46.49	37.2	3.13	64.4	2.7
S4	44.48	31.94	41.4	10.00	46.4	3.6
S5	49.76	43.60	37.5	14.58	51.2	3.6
S6	49.60	52.25	54.2	21.88	58.7	2.4
S7	49.93	36.27	35.4	27.08	30.1	3.9
S8	42.68	32.31	31.3	17.50	48.3	3.6
S9	40.63	36.49	46.3	0.00	59.8	2.9
S10	58.29	39.90	49.4	25.00	47.4	3.2
S11	36.26	31.26	34.4	9.38	48.3	3.6
S12	51.99	42.37	36.2	30.00	29.1	3.5
Average:	47.53	39.88	40.7	13.99	50.5	3.2

setup time length, as in Table 2) in Table 4. We note the effective mean utilization of the resources, the percentage of setups that occurred in the best solution (as a fraction of the total potential number of setups) and the fraction of the periods with at least one resource with fully utilized capacity. We say that the capacity of a resource is fully utilized if any slack capacity of that resource is insufficient for a single setup. In 29 of the 48 capacitated test problems, both resources are bottlenecks at some point in time. On average, only about 41 percent of the maximum potential number of setups are actually taken. Since about 23 percent of the maximum number of setups are inevitable (at least one set-up over the planning horizon is required for every product if any production is to take place), some batching is definitely occurring. We also report the percent difference between the lower bound (LP relaxation of MLCLSP) and the upper bound (the optimal solution) as a percentage of the upper bound ((UB − LB)/UB)—MLCLSP has weak lower bound. The last column in Table 4 shows the mean time between orders, measured in time periods.

We compare the CMHBR and optimal solutions in Table 5, where we report the average CPU time of the first integer solution located by OSL, the average CPU time of the optimal solution, and the average total time required by OSL to prove optimality. All figures in Table 5 are average values for the twelve basic cases.[8] We also note the average CPU time of the CMSB solution (the first integer solution found by the heuristic), the average CPU time of the best heuristic solution, and the average total CPU time required by the heuristic. All CPU times are expressed in seconds. Additionally, Table 5 presents the average percent differences between the objective values of the CMSB (the first heuristic feasible solution) and the optimal integer solutions, defined as $100 \times$ (CMSB Value—Optimal Integer)/Optimal Integer, and the average percent differences between the objective values of the best heuristic solutions and the optimal integer solutions, defined as $100 \times$ (Best Heuristic Value—Optimal Integer)/Optimal Integer.

These results suggest that CMHBR is a very good solution method, for it finds good feasible solutions very quickly. On average, heuristic solutions are about 4 percent above optimal plans, but they are located in less than one percent of the CPU time OSL needs to find the optimal solution. The CMHBR found optimal solutions for 6 percent of the cases. The relative difference between

[8] Detailed information for all 96 individual test problems, as well as the test problem data and the algorithm code, can be obtained directly from the authors.

Table 5 Small Random Test Problems: OSL and Heuristic Solutions Comparison

Problem Number	OSL First Time	Optimal Time	OSL Total Time	CMSB Time	CMSB Percent Difference with Optimal	CMHBR Best Time	CMHBR Total Time	Best Percent Difference with Optimal
S1	0.37	17.9	282.4	0.36	12.9	0.92	1.26	+2.2
S2	0.34	11.3	213.0	0.20	22.5	0.70	0.89	+7.1
S3	0.55	10.1	4205.0	0.41	13.0	1.33	1.58	+1.3
S4	0.31	5.8	62.6	0.31	23.2	0.66	1.04	+5.2
S5	0.48	744.2	2885.0	0.48	13.2	0.93	1.38	+2.5
S6	0.57	26.9	2412.7	0.33	21.7	1.31	1.90	+6.5
S7	0.47	61.9	521.4	0.55	14.8	1.03	1.35	+9.6
S8	0.24	3.7	18.2	0.30	20.4	0.70	0.91	+3.5
S9	0.56	366.7	6043.3	0.51	14.1	1.43	1.82	+3.6
S10	0.44	100.5	1867.8	0.49	22.3	0.99	1.53	+3.6
S11	0.36	17.2	82.2	0.37	20.9	0.86	1.17	+1.0
S12	0.29	5.9	25.0	0.38	11.8	0.72	1.16	+2.6
Average	0.41	114.3	1551.5	0.39	17.6	0.96	1.33	+4.1

CMHBR and optimal solutions was below 2.5 percent in 34 percent of the cases, between 2.5 percent and 5 percent in 38 percent of the cases, between 5 percent and 10 percent in 24 percent of the cases, and above 10 percent in 9 percent of the cases. We will briefly discuss cases where the heuristic performed poorly in §4. The CMHBR found an integer feasible solution in 100 percent of the cases. So although occasionally the heuristic solutions are not very good, they usually are quite close to optimal.

The CMHBR subroutines improve heuristic solution quality with modest computational effort. CMSB plans averaged 17.6 percent above optimal, but required only 0.34 percent of OSL's time to locate the optimal integer solution. During the second phase of the heuristic procedure, the SSR subroutine, decreased the relative difference with optimal by 40.1 percent, on average, after expending an additional 0.14 CPU seconds (34.9 percent more time). These heuristic solutions averaged 10.4 percent above optimal, but were located in only 0.46 percent of OSL's time. The final heuristic subroutines, the RSR and RIR, yielded the best heuristic solutions (4.1 percent above optimal on average), decreasing the relative difference with optimal by 60.6 percent and using 0.44 additional CPU seconds (82.7 percent more time).

The experiment's design allows us to examine the impact of controlled problem characteristics on heuristic performance. We use a matched pair t-test to control for dependence of data within each problem set (see Bickel and Doksum 1977). The number of matched pairs for each hypothesis test is 48, which allows us to invoke the Central Limit Theorem. We test the following three hypotheses.

To assess the impact of cumulative capacity usage on heuristic effectiveness, we test Hypotheses H_o against H_a.

HYPOTHESIS H_o. *Capacity considerations for the range of utilizations tested have no bearing on the performance of CMHBR.*

HYPOTHESIS H_a. *CMHBR performs relatively better on less capacitated problems.*

We find no evidence to reject H_o, for the test statistic is 0.54 with a p-value of 0.2900; therefore, capacity utilization does not seem to have an influence on the heuristic's performance for the range of utilizations tested in the paper.

We evaluate the importance of the magnitude of setup times relative to processing times by testing null hypothesis H_o against the two-sided alternative H_a.

HYPOTHESIS H_o. *The ratio of setup times to processing times has no bearing on the performance of CMHBR.*

HYPOTHESIS H_a. *The ratio of setup to processing times impacts CMHBR performance.*

Once again, we find no evidence to reject H_o with a test statistic of 0.91 and a p-value of 0.3640. Therefore, the magnitude of setup times relative to processing times does not impact heuristic effectiveness.

Finally, to explore the impact of setup costs, we test Hypothesis H_o against the two-sided alternative H_a.

HYPOTHESIS H_o. *The ratio of setup costs to inventory carrying costs has no bearing on the performance of CMHBR.*

HYPOTHESIS H_a. *The ratio of setup costs to inventory carrying costs impacts CMHBR performance.*

Once again, we find no evidence to reject H_o with a test statistic of 1.40 and a p-value of 0.1638. Therefore, the magnitude of setup costs relative to inventory carrying costs has no bearing on heuristic performance.

We are unable to find any evidence that the performance that CMHBR is significantly affected either by the tightness of capacity constraints or by the magnitude of setup times and setup costs. Therefore, we conclude that CMHBR performs consistently well in a variety of applications.

3.2. Medium-Scale Random Test Problems

Our second set of problems consists of 15 randomly generated problems that possess characteristics of more realistic production planning situations. Table 6 summarizes the problem characteristics from the product perspective, and Table 7 summarizes the problem characteristics from the resource perspective. These lot sizing problems schedule production for a random number of products across a 13- to 26-period planning horizon. The BOMs are randomly created to represent a variety of different product structures. BOM height, as suggested by the number of levels, is varied to create short, medium, and tall assembly structures. Four types of basic BOM forms are considered: ones with relatively even numbers of components on each BOM level (E), ones with increasing numbers of components on higher BOM levels (A), ones with decreasing numbers of components on higher BOM levels (V), and ones with hourglass shapes (H). In addition, each item has a small chance (one or two percent) of being a predecessor for items on more than one level. As in the small test problem set, end-item demand ranges from 1 to 10 units per period, and intermediate components possess no external demand. Initial finished goods inventory levels vary from 1 to 10 units, and initial WIP ranges from 5 to 15 units. Ending stock positions are required to match initial ones, and backorders are prohibited. Between three and six resources constrain production, and a resource's capacity is selected based on a randomly determined cumulative capacity usage ranging from 60 percent to

Table 6 Medium Random Test Problems Characteristics—Products

Problem Number	Number of Periods	BOM Type	Number of End Items	Number of Interm. Items	Number of Levels	Per Unit End Item Average Demand	Per Unit Interm. Item Average Demand	Average Per Unit Setup to Proc. Time	Average Setup to per Unit Inventory Carrying
M1	14	E	2	60	7	5.7	12.9	52.2	133.1
M2	19	E	2	56	6	5.6	12.2	51.9	108.1
M3	19	A	2	66	5	6.7	11.1	43.0	125.7
M4	14	A	2	74	5	5.4	10.5	51.2	102.3
M5	14	V	2	106	6	6.1	12.4	62.6	93.6
M6	14	H	2	80	6	4.8	10.6	53.5	88.4
M7	18	H	2	86	6	5.2	12.8	48.8	110.9
M8	15	V	2	44	4	5.9	7.8	46.3	99.2
M9	21	V	4	88	11	5.4	39.5	48.4	115.6
M10	22	E	5	78	11	5.6	21.8	53.1	107.6
M11	21	A	3	34	3	5.4	7.6	52.3	91.5
M12	13	V	4	38	3	4.7	10.2	44.7	107.1
M13	21	E	9	32	5	4.9	22.3	55.2	109.9
M14	16	E	3	52	3	13.9	19.1	51.5	84.0
M15	25	H	12	60	5	5.2	57.5	52.5	111.7

Table 7 Medium Random Test Problems Model Characteristics

Problem Number	Number of Rows	Number of Columns	Number of Binary Variables	Density (Percent)	Number of Resources	Average Number of Products per Resource	Average Total Requirements to Capacity (Percentage)
M1	1622	2355	546	0.16	3	15.0	80
M2	2149	3126	817	0.13	4	15.3	73
M3	2379	3530	855	0.11	3	18.0	73
M4	2067	2964	742	0.13	4	19.0	77
M5	3019	4296	1148	0.10	5	26.2	78
M6	2303	3275	868	0.13	5	20.8	76
M7	3012	4400	1116	0.09	4	21.0	83
M8	1435	2033	555	0.20	5	11.4	73
M9	3528	5297	1239	0.08	3	25.3	79
M10	2674	5389	1474	0.07	4	22.8	80
M11	1671	2392	651	0.17	6	8.5	77
M12	1169	1647	429	0.24	4	11.8	80
M13	2060	2964	819	0.13	5	10.6	84
M14	1875	2652	736	0.16	6	14.2	85
M15	3653	5516	1300	0.07	3	23.3	84

99 percent as described in §3.1. Each subassembly requires between 1 and 5 time units for processing and between 50 and 250 time units for resource setup. Since the setup times are very long relative to the processing times (more than 50 times longer, on average) and since the average requirements to capacity are not very high (78.8 percent, on average, compared to 95 percent in the small test problems), the effective mean utilization for the medium-sized problems is fairly low (23 percent on average). Per-unit holding costs range from 1 to 100,

Table 8 Medium Random Test Problems: OSL Solutions

Problem Number	Lower Bound	First Integer	First Time	Best Integer	(UB − LB)/UB (Percentage)	Best Time
M1	247,729	1367570.2	80.2	1299198.0	80.9	2560.7
M2	295,681	3646860.0	452.9	1548585.2	80.9	1485.6
M3	299,460	3511077.6	426.3	1481260.6	79.8	1555.7
M4	321,324	1482754.0	64.1	1474077.0	78.2	4028.8
M5	530,928	1085260.4	36.1	2502534.5	78.8	467.2
M6	400,561	2280383.8	127.1	1800878.7	77.8	1118.7
M7	433,101	1923374.6	108.4	2295521.6	81.1	3516.0
M8	242,055	4687544.5	631.5	1177825.2	79.4	2011.6
M9	587,016	1737494.7	179.1	2744408.4	78.6	4426.7
M10	523,076	1721549.1	135.6	3304218.6	84.1	4937.3
M11	212,098	1654595.3	129.9	1463063.6	85.5	217.4
M12	219,557	2583952.1	315.6	1021070.9	78.5	1680.0
M13	329,975	2119602.4	177.2	1808520.0	81.8	4232.6
M14	351,784	2485547.4	341.1	1923374.6	81.8	108.4
M15	598,682	1194781.0	49.8	4313165.6	86.2	4248.3
Average			217.0		80.9	2439.7

Table 9 Medium Random Test Problems: Heuristic Solutions

Problem Number	CMSB Solution	CMSB Time	Best Solution	Best Time	Total Time	Percent Difference with OSL Best
M1	1294392.8	16.6	1051239.1	95.2	128.2	−19.09
M2	2772093.0	199.5	1224759.7	183.2	241.0	−20.91
M3	2740588.5	108.6	1101008.7	294.8	346.2	−25.67
M4	735731.1	72.2	1179498.6	168.2	234.0	−19.98
M5	750134.0	7.8	2041465.7	411.4	562.0	−18.42
M6	1744444.1	35.8	1442894.5	202.5	278.0	−19.88
M7	1852907.9	22.4	1740976.3	363.2	487.4	−24.16
M8	2967814.4	277.5	819767.1	43.5	66.5	−30.40
M9	1504991.8	28.8	2482023.6	464.8	464.8	−9.56
M10	1161825.9	113.3	2340854.6	422.3	656.4	−29.16
M11	1471913.8	33.0	709518.9	82.8	139.3	−51.51
M12	2630238.8	65.3	638357.1	46.3	60.6	−37.48
M13	1872835.8	97.3	1510155.6	193.3	260.3	−16.50
M14	2120261.7	68.2	1428792.2	163.9	216.1	−25.71
M15	1024098.3	11.8	2772527.4	699.4	699.4	−35.72
Average		77.2		255.6	322.7	−25.61

while setup costs vary from 5000 to 10000. Table 7 presents the resulting mixed integer model characteristics.

As in our previous analysis, we compare the performance of CMHBR to default OSL; however, optimal solutions for these problems could not be located and verified within reasonable amounts of time. As a result, we focus on the best solutions found by OSL within 10,000 CPU seconds. The cited instances are the first 15 problems for which OSL was able to locate at least one integer solution within the allotted time. Tables 8 and 9 summarize the results of these test runs. As with the small problems, the lower bound is weak, explaining the poor OSL performance. For each of these problems the CMHBR located a better integer feasible solution with far less computational effort.

The CMHBR solutions average 25.6 percent better than OSL plans and were located in about 10 percent of OSL's time to find its best solution. An integer feasible solution was found by the heuristic in only 35.6 percent of the time needed by OSL to determine a feasible plan, and the first heuristic plans (the CMSB solution) averaged 19.5 percent better than the first OSL solutions and 11.1 percent better than the best OSL ones. The relative difference between the best OSL solution and the heuristic solution is increased, on average, by 47.4 percent through the SSR subroutine after an additional 54.5-second search (70 percent more time) and by 18.8 percent through the RSR and RIR subroutines after an additional 124-second search (94.2 percent more time).

4. Conclusions

We have presented a new heuristic (CMHBR) for finding good solutions for lot sizing problems with general assembly structures, setup costs, setup times, and multiple constrained resources. Our algorithm simply exploits the special structure of the underlying mathematical programming model, the one-to-one links between a subset of the continuous variables and the set of binary variables. Therefore, our method could be used to solve a large class of problems possessing this structure (for example, stochastic mixed integer problems or facility location problems).

We have compared the CMHBR with OSL on two problem sets. The first was a set of randomly generated, optimally solved small-scale problems, and the second was a set of randomly generated, more realistically sized problems. For the first group, the CMHBR performs well relative to OSL, in most cases locating good solutions in very small fractions of OSL's computational times. There were nine cases[9] out of our 96 small test problems where the heuristic performed relatively poorly, locating solutions 10.2 percent to 13.3 percent above optimal. In all those cases at least one of the SSR, RSR, and RIR procedures found a solution that was superoptimal but slightly infeasible, while the other routines were able to improve the CMSB solution only slightly. In a real manufacturing setting where some overtime is permitted, however, those superoptimal and slightly infeasible solutions could be quite useful.

For the second problem set, the CMHBR finds solutions which average 25.6 percent better than time-truncated OSL solutions. Again, the CMHBR uses much less CPU time to find those solutions. These results hold empirically across a wide range of problem instances. Overall, the CMHBR is a practical, fast and efficient technique for solving the general assembly structure lot sizing problem with multiple constrained resources. It requires only the use of an LP routine and therefore can be easily implemented on a large variety of platforms.

[9] Two cases of problem 2, one case of problems 4, 5, and 6, and four cases of problem 7.

References

Billington, P. J., J. O. McClain, and L. J. Thomas, "Mathematical Programming Approaches to Capacity-Constrained MRP Systems; Review, Formulation and Problem Reduction," *Management Sci.*, 29, 10 (1983), 1126–1141.

——, ——, and ——, "Heuristics for Multilevel Lot-Sizing with a Bottleneck," *Management Sci.*, 32, 8 (1986), 989–1006.

Bitran, G. R. and H. H. Yanasse, "Computational Complexity of the Capacitated Lot Sizing Problem," *Management Sci.*, 28, 10 (1982), 1174–1186.

Bahl, H. C., L. P. Ritzman, and J. N. D. Gupta, "Determining Lot Sizes and Resource Requirements: A Review," *Oper. Res.*, 35, 3 (1987), 329–345.

Bickel, P. J. and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics*, Holden-Day, 1977.

Blackburn, J. D. and R. A. Millen, "Improved Heuristics for Multi-Stage Requirements Planning Systems," *Management Sci.*, 28, 1 (1982), 44–56.

Diaby, M., H. C. Bahl, M. H. Karwan, and S. Zionts, "Capacitated Lot-Sizing and Scheduling by Lagrangean Relaxation," *European J. Oper. Res.*, 59 (1992a), 444–458.

——, ——, ——, and ——, "A Lagrangean Relaxation Approach for Very-Large-Scale Capacitated Lot-Sizing," *Management Sci.*, 38, 9 (1992b), 1329–1340.

Dixon, P. S. and E. A. Silver, "A Heuristic Solution Procedure for the Multi-Item, Single-Level, Limited Capacity Lot-Sizing Problem," *J. Operational Management*, 2 (1981), 23–39.

Eppen, D. E. and R. K. Martin, "Solving Multi-Item Capacitated Lot-Sizing Problems Using Variable Redefinition," *Operations Res.*, 35, 6 (1987), 832–848.

Florin, M., J. K. Lenstra, and A. H. G. Rinnooy Kan, "Deterministic Production Planning: Algorithms and Complexity," *Management Sci.*, 26, 7 (1980), 669–679.

Harris, F. W., "How Many Parts to Make at Once," *Factory, The Magazine of Management*, 10, 2 (1913), 135–136.

Harrison, T. P. and H. S. Lewis, "Lot Sizing in Serial Assembly Systems with Multiple Constrained Resources," *Management Sci.*, 42, 1 (1996),

Heinrich, C. and C. Schneeweiss, "Multi-Stage Lot-Sizing for General Production Systems," in S. Axsater, C. Schneeweiss, and E. Silver (Eds.), *Lecture Notes in Economics and Mathematical Systems*, Springer Verlag, Heidelberg, Germany, 1986.

Helber, S., "Lot Sizing in Capacitated Production Planning and Control Systems," *OR Spektrum*, 17 (1995), 5–18.

International Business Machines, Inc., *Optimization System Library User's Guide, Version* 1.2, 1991.

Kuik, R. and M. Salomon, "Multi-Level Lot Sizing Problem: Evaluation of a Simulated-Annealing Heuristic," *European J. Oper. Res.*, 45, 1 (1990), 25–37.

——, ——, and L. Van Wassenhove, "Batching Decisions: Structure and Models," *European J. Oper. Res.*, 75 (1994), 243–263.

——, ——, ——, and J. Maes, "Linear Programming, Simulated Annealing and Tabu Search Heuristics for Lot Sizing in Bottleneck Assembly Systems," *IIE Trans.*, 25, 1 (1993), 62–72.

Maes, J., J. O. McClain, and L. N. Van Wassenhove, "Multilevel Capacitated Lot Sizing Complexity and LP-Based Heuristics," *European J. Oper. Res.*, 53 (1991), 131–148.

Pochet, Y. and L. A. Wolsey, "Solving Multi-Item Lot-Sizing Problems Using Strong Cutting Planes," *Management Sci.*, 37, 1 (1991), 53–67.

Salomon, M., "Determining Lotsizing Models for Production Planning," in *Lecture Notes in Economics and Mathematical Systems*, 355, Springer Verlag, Heidelberg, Germany, 1991.

Silver, E. A. and H. C. Meal, "A Heuristic for Selecting Lot Size Quantities for the Case of Deterministic Time-Varying Demand Rate and Discrete Opportunities for Replenishment," *Production and Inventory Management 2nd Quarter*, 1973.

—— and R. Peterson, *Decision Systems for Inventory Management and Production Planning*, 2nd Ed., Wiley, New York, 1985.

Tempelmeier, H. and M. Derstroff, "A Lagrangian-Based Heuristic for Dynamic Multi-Level Multi-Item Constrained Lotsizing with Setup Times," *Management Sci.*, 42, 5 (1996), all pages.

—— and S. Helber, "A Heuristic for Dynamic Multi-Item Multi-Level Capacitated Lotsizing for General Product Structure," *European J. Oper. Res.*, 75 (1994), 296–311.

Trigeiro, W. W., L. J. Thomas, and J. O. McClain, "Capacitated Lot Sizing with Setup Times," *Management Sci.*, 35, 3 (1989), 353–366.

Wagner, H. M. and T. H. Whitin, "Dynamic Version of the Economic Lot Size Model," *Management Sci.*, 5, 1 (1958), 89–96.

Accepted by Luk Van Wassenhove; received July 1995. *This paper has been with the authors* 6 *months for* 2 *revisions.*