An International Journal
computers & mathematics
with applications

PERGAMON Computers and Mathematics with Applications 44 (2002) 263–276

www.elsevier.com/locate/camwa

Heuristic Genetic Algorithms for General Capacitated Lot-Sizing Problems

JINXING XIE* AND JIEFANG DONG
Department of Mathematical Sciences
Tsinghua University, Beijing 100084, P.R. China
jxie@math.tsinghua.edu.cn

(Received January 2001; accepted February 2001)

Abstract—The lot-sizing problems address the issue of determining the production lot-sizes of various items appearing in consecutive production stages over a given finite planning horizon. In general capacitated lot-sizing problems, the product structure can be a general acyclic network, the capacity constraints can be very sophisticated, and all the known parameters can be time-varying. This paper proposes heuristic genetic algorithms for these problems by designing a domain-specific encoding scheme for the lot-sizes and by providing a heuristic shifting procedure as the decoding schedule. The main contribution of these genetic algorithms is the presentation technique that encodes only the binary variables for the setup patterns but derives other decision variables by making use of the problem-specific knowledge. Some results from the computational experiments are also given.

Keywords—Production planning, Inventory control, Capacitated lot-sizing, Genetic algorithm (GA), Heuristics.

1. INTRODUCTION

The widespread and popular use of material requirements planning (MRP) systems in industry has resulted in increased interest in the topic of decision making in capacitated multistage manufacturing systems. However, the lot-sizing procedures used by currently available MRP systems are quite limited in their ability to coordinate production plans of various stages of the manufacturing processes and various capacity constraints. As firms have incorporated MRP concepts into their production planning and distribution systems, the capacitated multistage lot-sizing decision problem has become a problem of prime importance.

The lot-sizing problem addresses the issue of determining the production lot-sizes of various items appearing in consecutive production stages over a given finite planning horizon. The objective of the problem is to reduce the total manufacturing cost (including setup cost, inventory holding cost, overtiming cost, etc.) while trying to satisfy the customers' requirements with the limited capacity. This kind of problem can be classified into different categories according to

*Author to whom all correspondence should be addressed.
This research has been supported by the National Natural Science Foundation of China (Project No. 69904007).

0898-1221/02/$ - see front matter Typeset by AMS-TEX
PII: S0898-1221(02)00146-3

the product structures (e.g., single level system, serial, assembly, and general systems) and the capacity structures (e.g., uncapacitated, capacitated single resource, and capacitated multiple resources). Many researchers have studied the lot-sizing problems and designed a lot of optimal or heuristic lot-sizing procedures (e.g., references [1–9] are some excellent reviewing papers on lot-sizing problems).

Table 1 gives a brief review of some important or reviewing literatures for the different categories of the capacitated lot-sizing problems. Because the problems are NP-hard and even the feasibility problems with setup times are also NP-hard [3], most of the lot-sizing procedures and algorithms use heuristic techniques to solve the problems. However, the heuristic lot-sizing techniques for capacitated production systems usually concentrate on optimizing the production operations stage by stage, and/or only consider some simple product structures and/or simple capacity constraints. For example, the capacitated lot-sizing problems for general product structures and multiple resources are seldom considered. This is obviously an obstacle to the application of these lot-sizing techniques in real production planning and scheduling environments.

Table 1. Different capacitated lot-sizing problems and some important/reviewing literature.

Product Structure	Uncapacitated	Single Resource	Multiple Resources
Single Level	[2,10,11]	[12–16]	
Serial	[2,17]	[17]	
Assembly	[2,18–20]	[3,21–23]	
General	[2,24,25]	[6,26]	[27,28]

In the last decade, the genetic algorithm (GA), which is a search technique based on the mechanics of natural selection and natural genetics, is recognized as a powerful and widely applicable optimization method, especially for global optimization problems and NP-hard problems [29–31]. Recently, a lot of researchers studied the applications of GA for solving the lot-sizing problems with unlimited capacity [32–34] and with capacity constraints [35–42]. Numerical results obtained using these methods show that GA (probably combined with other meta-heuristics) is an effective approach to deal with the lot-sizing problems.

Before using GA to solve an optimization problem, there are two important points which must be addressed clearly: the first is the encoding (representation) scheme for the decision variables of the optimization problem, and the second is the evaluation scheme for the specific individual (chromosome) of the problem. These two schemes are interrelated and their improper combination can make GA unable to deal with the optimization problems efficiently, especially for the optimization problems with nontrivial constraints as in the general capacitated lot-sizing problems. For the capacitated lot-sizing problems, constraints that cannot be violated can be implemented by the penalty method and the decoder method. The penalty method imposes penalties on individuals that violate the constraints, while the decoder method creates decoders of the representation that avoid creating individuals violating the constraints [31]. Since each optimization problem has its own features, it is also recognized that better performance can be obtained when the problem-specific knowledge is incorporated into the simple GA [31]. In fact, there are some fundamental limitations to genetic algorithms according to the so-called No-Free-Lunch theorem [43–45]. One of the most significant implications of the No-Free-Lunch theorem is that algorithms should be matched to the search problem at hand. "If no domain-specific knowledge is used in selecting an appropriate representation, the algorithm will have no opportunity to exceed the performance of an enumerative search" [43]. Generally speaking, a better solution can be gained with deeper domain-knowledge being incorporated. However, the domain-knowledge of a specific optimization problem is usually too vast to be considered

completely with a single algorithm. Even worse, there may be some domain-knowledge which is difficult to be considered explicitly when designing an evolutionary algorithm. These difficulties reveal that when designing a specific evolutionary algorithm for a specific problem domain, a critical issue is how to exploit the domain-specific knowledge and how to incorporate it into the common logic and the general structure of the evolutionary algorithms.

According to this philosophy, a heuristic genetic algorithm for the general capacitated lot-sizing problems is presented in this paper. The general capacitated lot-sizing problems can be characterized by the following:

(1) each stage in the product network may have several predecessors and several successors;
(2) the capacity constraints may include the capacity limitations of multiple resources;
(3) the capacity of each resource can be adjusted by overtiming;
(4) besides resulting in setup cost, the setup of each process can also occupy capacity (i.e., setup times are included but the setup times are assumed to be sequence-independent in this study);
(5) independent demands can be given to all the items in the product network; and/or
(6) all the parameters in the problem can be time-varying.

This kind of capacitated lot-sizing problem is more general than those currently considered by most of the researchers and are more applicable to real production environments. The heuristic genetic algorithm proposed in this paper attempts to incorporate the problem-specific knowledge into the conventional genetic algorithms and heuristically optimizes the problem under all the constraints simultaneously. In order to do that, an encoding (representation) scheme for lot-sizes and a shifting procedure with penalty considerations are designed by making use of the critical properties of the optimal solutions to the problem. This kind of method for dealing with the sophisticated constraints can be considered as a fusion of the penalty method and the decoder method.

The remainder of this paper is organized as follows. A mathematical formulation of the general capacitated lot-sizing problems considered in this paper is provided in the next section. Then, in Section 3, we present the heuristic genetic algorithms for the problems with and without overtiming considerations, respectively. Section 4 gives some results of the computational experiments. Some general conclusions and further research directions are summarized in Section 5.

2. MATHEMATICAL FORMULATION

Consider the following general capacitated lot-sizing problem (GCLSP). Given the external demand for N items over a time horizon of T periods, find a solution which minimizes total setup, production, holding, and overtiming costs, satisfying the following conditions.

- The product structure can be organized as an acyclic directed network, where every node in the network is an item and the arc illustrates the assembly or distribution relation between items, and the weight of an arc is the quantity relation between the two terminal nodes of the arc. In general production systems, each node can have more than one immediate predecessor and more than one immediate successor. (By topological ordering for the directed graph, we can assume that all the nodes are labeled as satisfying the condition that the label of a predecessor is greater than that of each of its successors.)
- There are multiple resources with limited capacities that must be respected. When necessary, the capacity of a resource can be increased by overtime, which is also limited with a maximum value.
- The backlogging of each item is not allowed.
- The lead times are assumed to be constant and, without loss of generality, are assumed to be zero.

Mathematically, this problem can be stated as follows.

PROBLEM GCLSP.

$$\min \mathrm{COST}(Y,X,I,O) = \sum_{i=1}^{N}\sum_{t=1}^{T}\{s_{it}I_{it} + c_{it}X_{it} + h_{it}I_{it}\} + \sum_{k=1}^{K}\sum_{t=1}^{T} o_{kt}O_{kt}, \tag{1}$$

$$\text{s.t.} \quad I_{i,t-1} + X_{it} - I_{it} = d_{it} + \sum_{j\in S(i)} r_{ij}X_{jt}, \qquad i = 1,\ldots,N, \quad t = 1,\ldots,T, \tag{2}$$

$$\sum_{i=1}^{N}(a_{kit}X_{it} + A_{kit}Y_{it}) \le C_{kt} + O_{kt}, \qquad k = 1,\ldots,K, \quad t = 1,\ldots,T, \tag{3}$$

$$Y_{it} = \begin{cases} 0, & \text{if } X_{it} = 0, \\ 1, & \text{if } X_{it} > 0, \end{cases} \qquad i = 1,\ldots,N, \quad t = 1,\ldots,T, \tag{4}$$

$$I_{it}, X_{it} \ge 0, \qquad i = 1,\ldots,N, \quad t = 1,\ldots,T, \tag{5}$$

$$Y_{it} \in \{0,1\}, \qquad i = 1,\ldots,N, \quad t = 1,\ldots,T, \tag{6}$$

$$0 \le O_{kt} \le U_{kt}, \qquad k = 1,\ldots,K, \quad t = 1,\ldots,T, \tag{7}$$

where the given parameters are

N the number of items,

T the number of time periods in the planning horizon,

K the number of resources,

d_{it} the external demand for item i in period t,

$S(i)$ the set of immediate successors of item i ($S(i) = \emptyset$ if i is an end item),

r_{ij} the number of units of item i required to produce one unit of item j,

s_{it} the setup cost for item i in period t,

c_{it} the production cost for unit item i in period t,

h_{it} the holding cost for unit end-of-period inventory of item i in period t,

o_{kt} the overtime cost for unit resource k in period t,

C_{kt} the available normal capacity of resource k in period t,

U_{kt} the available maximum overtime of resource k in period t,

a_{kit} the capacity needed on resource k to produce one unit of item i in period t,

A_{kit} the fixed loss of resource k incurred for production preparation of item i in period t,

and the decision variables are

X_{it} the amount of item i produced in period t (lot-size),

Y_{it} a binary variable indicating where production is allowed for item i in period t,

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

O_{kt} the overtime of resource k in period t.

The objective function (1) means to minimize the total cost which includes setup, production, holding, and overtiming costs. The constraint (2) is the conservation equation for materials, and the constraint (3) enforces the capacity feasibility. The constraints (4) and (6) imply that the setup corresponding to an item must be paid before producing the item. The nonnegative restrictions for inventory and production quantities in constraint (5) assure no backlogging occurs, and the last inequality (7) states the lower and upper bounds available for overtiming for each resource at different time periods. Most of the concepts mentioned above can be found in many other papers on capacitated lot-sizing problems. For example, the mathematical formulation given in [3] is very similar to our model GCLSP except that the overtimes of resources are not included in it.

3. HEURISTIC GENETIC ALGORITHM

3.1. The Encoding Scheme

A genetic algorithm of an optimization problem is an iterative procedure attempting to heuristically search the optimal solution of the problem. Three basic genetic operators included in the procedure are known as reproduction, mutation, and crossover [29–31]. All these operators operate on the chromosomes (representations of the decision variables). While designing a genetic algorithm to solve a problem, we must first give an encoding (representation) scheme for the decision space (space of the decision variables) of the problem.

The decision variables in GCLSP are X_{it}, Y_{it}, I_{it}, and O_{kt}, among which Y_{it} is a 0-1 integer variable and the others are positive variables of real numbers. In order to design a computationally efficient genetic algorithm, in this study we only encode the setup patterns (variables Y_{it}) as chromosomes. The other real number variables X_{it}, I_{it}, and O_{kt} will be virtually considered as being dependent on Y_{it}, and thus, they will be computed from Y_{it} and the known parameters of the problem. The most important thing when designing such an encoder-decoder scheme to GCLSP is how to obtain the values of these variables from Y_{it} and the known parameters. In the following paragraphs, we will show how these variables can be heuristically worked out by making use of the problem-specific knowledge of the lot-sizing problems.

Denote the population size in a genetic algorithm as MAXPOP (assume to be an even number) and the maximum iterations (i.e., maximum generations) as MAXGEN. The j^{th} individual (i.e., decision variable) in the g^{th} generation/iteration is encoded as follows:

$$\begin{gathered} Y^{g,j} = \left(Y_{11}^{g,j}, Y_{12}^{g,j}, \ldots, Y_{1T}^{g,j}, Y_{21}^{g,j}, Y_{22}^{g,j}, \ldots, Y_{2T}^{g,j}, \ldots, Y_{N1}^{g,j}, Y_{N2}^{g,j}, \ldots, Y_{NT}^{g,j}\right), \\ j = 1, 2, \ldots, \text{MAXPOP}, \qquad g = 1, 2, \ldots, \text{MAXGEN}. \end{gathered} \tag{8}$$

According to Afentakis and Gavish [25], there exists the following property (usually named "zero-switch" property) for uncapacitated lot-sizing problem, a simplified version of GCLSP.

PROPOSITION. *There is an optimal solution to uncapacitated lot-sizing problem (the problem GCLSP without resources constraints) in which* $x_{it} I_{i,t-1} = 0$.

Making use of this important property, we can heuristically clarify the relationships between the decision variables of real numbers, and the binary decision variables Y_{it} and the known parameters of the problem. Given a production pattern (setup sequence), i.e., the binary decision variables

$$\{Y_{it},\ i = 1, \ldots, N,\ t = 1, \ldots, T\},$$

the production lot-sizes can be determined according to the "zero-switch" property as follows.

(i) For any item i and any period τ, if $Y_{i\tau} = 0$, then $X_{i\tau} = 0$.

(ii) For any item i and any periods $\tau_1 < \tau_2 \leq T$, if $Y_{i\tau_1} = Y_{i\tau_2} = 1$, and $Y_{i\tau} = 0$ for all $\tau_1 < \tau < \tau_2$, then

$$X_{i\tau_1} = \sum_{\tau=\tau_1}^{\tau_2 - 1} \left(d_{i\tau} + \sum_{j \in S(i)} r_{ij} X_{j\tau} \right). \tag{9}$$

That is to say, whenever a setup is introduced for an item, we produce enough quantity for this item to satisfy the demands of an integer number of periods, until the period immediately before the next setup for this item. However, when no setup is activated in period 1 for a product (item), this processing method may lead to backlogging in the first few periods for this product and its successive parent products. In order to assure the feasibility, the penalty method is used to deal with the constraints that no backlogging is allowed. That is to say, we can change the

objective function to include penalties for backlogging. Thus, the fitness value for each individual is calculated according to the following objective function:

$$\text{COSTU}(Y, X, I) = \sum_{i=1}^{N}\sum_{t=1}^{T}\{s_{it}Y_{it} + c_{it}X_{it} + h_{it}I_{it}\} + \beta\sum_{i=1}^{N}\sum_{t=1}^{T}[\max\{0, -I_{it}\}]^2, \tag{10}$$

where β is the penalty coefficient (a large enough positive number).

Based on these observations, the genetic algorithm has been used to solve the uncapacitated lot-sizing problems [32–34]. For a capacitated lot-sizing problem, the "zero-swith" property no longer holds. However, we can accept the similar methodology to determine the values of the real variables and concurrently include other procedures to assure the capacity feasibility. In the following, we generalize the genetic algorithm for the uncapacitated lot-sizing problems to be capable of solving GCLSP with multiple resources constraints and overtiming considerations.

3.2. Heuristic Genetic Algorithm for Capacitated Lot-Sizing Problems without Overtiming

For capacitated lot-sizing problems, the optimal solutions may not satisfy the "zero-switch" property. In fact, the decision variables X_{it}, I_{it} are not only dependent on the production pattern (setups sequence) Y_{it}, but also dependent on the available resources capacities C_{kt}. In this situation, we can first determine X_{it}, I_{it} from Y_{it} without considering the resources constraints and take this lot-sizes plan as an initial plan, which will be further modified by considering the resources constraints. In this paper, we will modify this initial lot-size plan by "shifting" techniques that have been widely used in the heuristics for the lot-sizing problems [22,27].

The shifting procedure checks the capacity feasibility of the initial lot-size schedule from the last period backwards to period 1. In each period t, a capacity tightness index, ρ_{kt}, which is defined to be the ratio of the total capacity for a resource needed to produce all the scheduled items in this period to the total available capacity for this resource at this period, is calculated for each resource k,

$$\text{CN}_{kt} = \sum_{i=1}^{N}(a_{kit} \times X_{it} + A_{kit} \times Y_{it}), \qquad k = 1, 2, \ldots, K, \tag{11}$$

$$\rho_{kt} = \frac{\text{CN}_{kt}}{C_{kt}}, \qquad k = 1, 2, \ldots, K, \tag{12}$$

where CN_{kt} is the total capacity for a resource k needed to produce all the scheduled items in period t. The remaining capacity for a resource k in period t can be calculated as

$$\text{CR}_{kt} = C_{kt} - \text{CN}_{kt}, \qquad k = 1, 2, \ldots, K. \tag{13}$$

It is obvious that the resources capacities are enough to produce all the scheduled items in a period t if the capacity tightness indexes are all less than or equal to one (i.e., $\rho_{kt} \leq 1$ for all $k = 1, 2, \ldots, K$) at this period. In this case, the shifting procedure moves to period $t-1$ and begins to check the capacity feasibility in this following period. If in some period t the total capacity for a resource needed to produce all the scheduled items in this period is larger than the total available capacity for this resource at the period (i.e., $\rho_{kt} \leq 1$ for some k), an infeasibility occurs for this resource k, and thus, one or more of the scheduled item(s) in this period must be shifted to the previous period(s). If there are two or more such kinds of resources, the procedure begins with the resource with the largest tightness index. In order to get a capacity-feasible plan more efficiently, we only consider moving the excessive scheduled production quantity to the period just before this period. In order to reduce as much as possible the possibility of any infeasibility resulting from the shifting procedure for the materials between different stages, the

items are shifted according to the decreasing order of the labels of the scheduled items in this period (i.e., from item N down to 1). This is based on the observation that the item with a larger label will not use an item with a smaller index as its component. That is to say, beginning with item N and down to item 1, we move the excessive production quantity of one or more scheduled items in this period to its previous period until the capacity-unfeasibility for this resource in this period is eliminated. If there is (are) other infeasible resource(s), we repeat this shifting procedure until all the infeasibility is eliminated. When the moving processes are finished, the modified lot-size plan will be a feasible plan with respect to the capacity constraints. Hence, the only infeasibility of this modified lot-size plan may be the backlogging for some items in some periods, and these kinds of infeasibilities will be tackled with the penalty method by using the similar penalty cost function (10) for the uncapacitated lot-sizing problem.

In summary, the shifting procedure goes successively from period T down to 1 and from the tightest resource to the most flexible one within a period. While the capacity infeasibility occurs in some period, move the excessive production of one or more items in this period just before current period to assure the capacity feasibility (if the current period is the first period, then the production quantity for this item in this period will be partly or completely lost). In this shifting or moving excessive production backwards procedure, the remaining capacities of all the resources in these two successive periods are adjusted through considering the shifted quantity and the possible eliminated or added setup times (i.e., the setup-consumed capacities). The setup patterns of the shifted items in this two periods will be modified whenever applicable.

The detailed heuristic genetic algorithm for capacitated lot-sizing problems without overtiming (GAC) can be described as follows.

Algorithm GAC

STEP 1. $g = 0$. Randomly initialize oldpop $= \{Y^{0,j},\ j = 1, 2, \ldots, \text{MAXPOP}\}$, the population set in g^{th} generation.

STEP 2. If $g =$ MAXGEN, print the solution and stop.

STEP 3. For each $Y^{0,j} \in$ oldpop, $j = 1, 2, \ldots, \text{MAXPOP}$, compute its objective function value as follows.

3.1. Construct the initial lot size schedule according to equation (9) without considering the resources constraints. Calculate X^j according to $Y^{0,j}$ as follows.

For items i from 1 to N, if $Y^{0,j}_{it_1} = 1$, $Y^{0,j}_{it_2} = 1$ $(1 \le t_1 < t_2 \le T+1)$, and $Y_{i\tau} = 0$ for all $t_1 < \tau < t_2$ (assume $Y^{0,j}_{i,T+1} = 1$), then let

$$X^j_{it_1} = \sum_{\tau=t_1}^{t_2-1} \left(d_{i\tau} + \sum_{m \in S(i)} r_{im} X^j_{m\tau} \right), \quad X^j_{i\tau} = 0, \qquad 1 \le t_1 < \tau < t_2 \le T+1.$$

3.2. Eliminate the capacity-infeasibility for all the resources by the shifting procedure. The processing in period t (from T down to 1) is as follows.

Calculate the resources tightness indexes ρ_{kt} for resource k (from 1 to K) according to equation (12). If all ρ_{kt} are less than one, go to Step 3.3. Otherwise select the tightest resource k and calculate CR_{kt} (the remaining capacity of resource k in period t) as

$$\text{CR}_{kt} = C_{kt} - \sum_{i=1}^{N} \left(A_{kit} Y^{0,j}_{it} + a_{kit} X^j_{it} \right).$$

While $\text{CR}_{kt} < 0$, for item i (from N down to 1), moving part or all of the production quantity of item i in period t backwards to period $t-1$:

(i) If $\text{CR}_{kt} + a_{kit} X^j_{it} > 0$, then move part of the production quantity of item i in period t backwards to period $t-1$ and the moved quantity is $-\text{CR}_{kt}/a_{kit}$. After this movement,

the remaining capacity of resource k in period t is zero, and the production quantity of item i in period $t-1$ is increased by $-\mathrm{CR}_{kt}/a_{kit}$. That is to say, we set

$$X_{it}^{j} = X_{it}^{j} + \frac{\mathrm{CR}_{kt}}{a_{kit}}, \quad X_{i,t-1}^{j} = X_{i,t-1}^{j} - \frac{\mathrm{CR}_{kt}}{a_{kit}}, \quad Y_{i,t-1}^{0,j} = 1, \quad \mathrm{CR}_{kt} = 0.$$

(ii) If $\mathrm{CR}_{kt} + a_{kit}X_{it}^{j} \le 0$, then move all of the production quantity of item i in period t backwards to period $t-1$, and the moved quantity is X_{it}^{j}. After this movement, the remaining capacity of resource k in period t is increased by $A_{kit} + a_{kit}X_{it}^{j}$, and the production quantity of item i in period $t-1$ is increased by X_{it}^{j}. That is to say, we set

$$X_{it}^{j} = 0, \quad X_{i,t-1}^{j} = X_{i,t-1}^{j} + X_{it}^{j}, \quad Y_{i,t-1}^{0,j} = 1, \quad Y_{it}^{0,j} = 0,$$
$$\mathrm{CR}_{kt} = \mathrm{CR}_{kt} + A_{kit} + a_{kit}X_{it}^{j}.$$

3.3. Determine I^j (inventory) ($I_{i,0}^{j}(\forall i, j)$ are known parameters) according to X^j (compute from period 1 to T for each item),

$$I_{i,t}^{j} = I_{i,t-1}^{j} + X_{i,t}^{j} - \sum_{m \in S(i)} X_{mt}^{j}, \qquad 1 \le t \le T.$$

3.4. Compute the corresponding objective function value according to (10) from $Y^{0,j}$, X^j, I^j.

STEP 4. Generate newpop $= \{Y^{1,j},\ j = 1, 2, \ldots, \mathrm{MAXPOP}\}$, the population set in $(g+1)^{\mathrm{th}}$ generation.

4.1. Calculate the fitness value $\mathrm{fit}(Y^{0,j})$ for each individual $Y^{0,j}$ according to the objective function values obtained in Step 3.4.

$$\mathrm{fit}\left(Y^{0,j}\right) = \max_{i=1}^{\mathrm{MAXPOP}} \mathrm{COSTU}\left(Y^{0,i}\right) + \varepsilon - \mathrm{COSTU}\left(Y^{0,j}\right), \qquad j = 1, 2, \ldots, \mathrm{MAXPOP},$$

where ε is a positive constant and $\mathrm{COSTU}(Y) = \mathrm{COSTU}(Y, X, I)$.

4.2. Reproduction/Selection. Select Y^{0,j_1}, Y^{0,j_2} from the set oldpop according to the fitness values. The probability to select $Y^{0,j}$ is

$$\mathrm{pr}\left(Y^{0,j}\right) = \frac{\mathrm{fit}\left(Y^{0,j}\right)}{\sum_{i=1}^{\mathrm{MAXPOP}} \mathrm{fit}\left(Y^{0,i}\right)}, \qquad j = 1, 2, \ldots, \mathrm{MAXPOP}.$$

4.3. Mutation. If the mutation probability is p_m, then after the mutation,

$$Y_{it}^{\prime 0,j} = \begin{cases} Y_{it}^{0,j}, & \text{in probability of } 1 - p_m, \\ \bar{Y}_{it}^{0,j}, & \text{in probability of } p_m, \end{cases} \qquad j = j_1, j_2.$$

Here,

$$\bar{Y}_{it}^{0,j} = \begin{cases} 0, & \text{if } Y_{it}^{0,j} = 1, \\ 1, & \text{if } Y_{it}^{0,j} = 0. \end{cases}$$

4.4. Crossover. If the crossover probability is p_c, then after the crossover,

$$Y^{1,j} = \begin{cases} Y^{\prime 0,j}, & \text{in probability of } 1 - p_c, \\ Y^{\prime\prime 0,j}, & \text{in probability of } p_c, \end{cases} \qquad j = j_1, j_2.$$

Here, randomly select a crossover position s $(1 \le s \le NT)$ and then let

$$Y^{\prime\prime 0,j_1} = \left(Y_1^{\prime 0,j_1}, Y_2^{\prime 0,j_1}, \ldots, Y_s^{\prime 0,j_1}, Y_{s+1}^{\prime 0,j_2}, Y_{s+2}^{\prime 0,j_2}, \ldots, Y_{NT}^{\prime 0,j_2}\right),$$
$$Y^{\prime\prime 0,j_2} = \left(Y_1^{\prime 0,j_2}, Y_2^{\prime 0,j_2}, \ldots, Y_s^{\prime 0,j_2}, Y_{s+1}^{\prime 0,j_1}, Y_{s+2}^{\prime 0,j_1}, \ldots, Y_{NT}^{\prime 0,j_1}\right).$$

4.5. Add the new individuals Y^{1,j_1}, Y^{1,j_2} to the set newpop.

4.6. If all MAXPOP individuals are produced, then go to Step 5; otherwise go to Step 4.2.

STEP 5. Set $g = g + 1$, oldpop = newpop, i.e., $Y_{it}^{0,j} = Y_{it}^{1,j}$, $\forall i, t, j$, and go to Step 2.

3.3. Heuristic Genetic Algorithm for Capacitated Lot-Sizing Problems with Overtiming

For capacitated lot-sizing problems with overtiming, we can also use the shifting procedure (see [22,27]) to move the lot-sizes backwards to respect the capacity constraints similarly to that of the capacitated lot-sizing problems without overtiming. The only difference is that in this situation, the capacity of a resource includes two parts, i.e., normal capacity and overtime capacity. Therefore, before moving excessive production backwards from a capacity-infeasible period, we first consider eliminating capacity-unfeasibility by overtiming in the current period. If the sum of the normal capacity and maximum overtime capacity is still not enough to assure the production scheduled, then apply the shifting procedure.

In order to assure the feasibility, we change the objective function to including penalties as follows:

$$\begin{aligned} \mathrm{COSTO}(Y,X,I,O) = &\sum_{i=1}^{N}\sum_{t=1}^{T}\{s_{it}Y_{it} + c_{it}X_{it} + h_{it}I_{it}\} + \sum_{k=1}^{K}\sum_{t=1}^{T} o_{kt}O_{kt} \\ &+ \beta\sum_{k=1}^{K}\sum_{t=1}^{T}[\max\{0,-I_{it}\}]^2, \end{aligned} \tag{14}$$

where β is the penalty coefficient (a large enough positive number).

The heuristic genetic algorithm for capacitated lot-sizing problems with overtiming (GAO) can be described as follows.

Algorithm GAO

Only Steps 3 and 4 of the Algorithm GAC need to be changed.

STEP 3. For each $Y^{0,j} \in$ oldpop, $j = 1,2,\ldots,$MAXPOP, calculate its objective function value as follows.

3.1. Construct the initial lot size schedule according to equation (9) without considering the resources constraints (same as in Step 3.1 of the Algorithm GAC).

3.2. Eliminate the infeasibilities of the resources by shifting procedure similar to Step 3.2 in the Algorithm GAC. This procedure goes from period T down to 1 and from the tightest resource to the loosest resource within a period. If the normal capacity of a resource is not enough to eliminate the infeasibilities in a period, then first consider overtiming with this resource in this period. If the total capacity (including normal capacity and the overtime) of this resource in this period is still not enough, then shift some or all of the production quantities to the period just before current period to assure the capacity-feasibility. (If the current period is the first period, then the production for this item in this period will be lost.) The shifting procedure goes for item N down to 1. During the shifting procedure, the setup times (i.e., the setup-consumed capacities) are also considered, and the remaining capacity is adjusted. The corresponding setup patterns $Y_{it}^{0,j}$, $Y_{i,t-1}^{0,j}$ are also correctly reset.

3.3. Determine inventory I^j (the same as Step 3.3 of the Algorithm GAC).

3.4. Calculate the corresponding objective function value COSTO according to (14) from $Y^{0,j}$, X^j, I^j.

STEP 4. Same as the Step 4 of the Algorithm GAC. However, we use COSTO to evaluate the fitness values of individuals in this algorithm instead of using COSTC in the Algorithm GAC.

4. COMPUTATIONAL EXPERIMENTS

The performance of the proposed algorithms GAC and GAO described in the previous section has been evaluated on a set of testing problems. The algorithms are programmed with C++

running on a Pentium-III. The experiments reveal that these algorithms obtain a very good approximation solution of GCLSP in reasonable computation time. In the following are some of the examples of the experiments.

We have conducted a primary test to investigate the impact of the control parameters of the genetic algorithms on the performance of GAC and GAO. The algorithms converge to a solution within 200 generations for most of the small-scale examples (e.g., $N \leq 10$, $K \leq 2$) and within 500 generations for most of the modest examples (e.g., $N \approx 20$, $K \approx 2$). The crossover probabilities between 0.6 to 1.0 and the mutation probabilities between 0.005 to 0.033 give no significant difference regarding the total cost performance. But, the mutation probabilities less than 0.005 show worse total cost performance. In addition to these control parameters, the elitist strategy can enhance the performance significantly, and thus, it is incorporated into the algorithms.

In all of the following numerical examples, we used the following control parameters for genetic algorithms:

- Maximum generations MAXGEN = 500.
- Population size MAXPOP = 30.
- Mutation probability $p_m = 0.033$.
- Crossover probability $p_c = 0.6$.

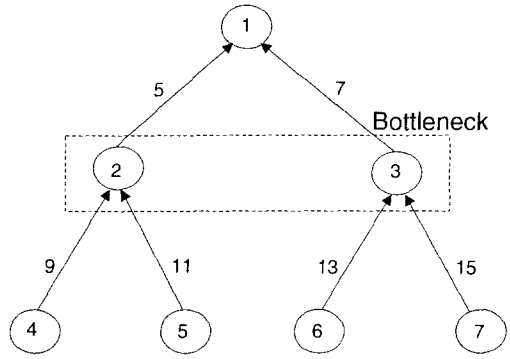

Figure 1. Product structure for Example 1.

Table 2a. External demands for end item and available capacity for resource (mean values*).

Period t	1	2	3	4	5	6
External demand d_{1t}	40	0	100	0	90	10
Available capacity C_t	10000	0	5000	5000	1000	1000

Table 2b. Setup costs and marginal holding costs for items (mean values*).

Item i	1	2	3	4	5	6	7
Setup cost $s_{it}(= s_i)$	400	500	1000	300	200	400	100
Holding cost $h_{it}(= h_i)$	12	0.6	1.0	0.04	0.03	0.04	0.04

*Average values for ten groups of randomly generated data.

EXAMPLE 1. CAPACITATED LOT-SIZING WITHOUT OVERTIMING FOR ASSEMBLY SYSTEM. In this example, the product structure is shown as Figure 1 with $N = 7$, $T = 6$, $K = 1$. Only the end item (item 1) has external demands, and the marginal holding costs and setup costs of all the items are assumed to be constant over time. Marginal production costs are constant over time and are assumed to be zero (i.e., $c_{it} = 0$, $\forall(i,t)$). Only the production of items 2 and 3

are capacitated, and the setup times are assumed to be ignorable (i.e., $A_{it} = 0$, $\forall(i,t)$, $a_{it} = 0$, $\forall t$, $\forall i \neq 2,3$, $a_{2t} = a_2 = 5$, $a_{3t} = a_3 = 8$). For this example, each tested algorithm is run with ten groups of randomly generated data. The mean demands for the end item (item 1) and the mean capacity for the resource (bottleneck) in different time periods are shown in Table 2a. The mean marginal holding costs and setup costs of different items in different time periods are shown in Table 2b. We compared GAC with SA (simulated annealing algorithm [23]), TS (taboo search algorithm [23]), and LR (Lagrangean relaxation algorithm [21]). These algorithms are programmed by ourselves according to the descriptions in the original papers of Kuik *et al.* [23] and Billington *et al.* [21], respectively. For each of the randomly generated problems, the three probability algorithms (i.e., algorithms GA, TS, and GAC) are run five times using different random number seeds, respectively. The final results are summarized on Table 3. Examination of the results shows that the solutions obtained by GAC are much better than that of SA and TS in approximately the same computing time, while GAC obtains approximately the same better solutions as LR while in much less time than that of LR.

Table 3. GAC compared with other algorithms.*

Algorithm	Mean Optimal Value	Compared with GAC	Computing Time (s)
SA	10740.00	1.162	9.90
TS	9620.00	1.041	8.80
LR	9239.00	0.999	34.10
GAC	9245.00	1.000	10.10

*In this example, each algorithm is tested with ten groups of randomly generated data, and the probability algorithms GA, TS, and GAC are run five times using different random number seeds.

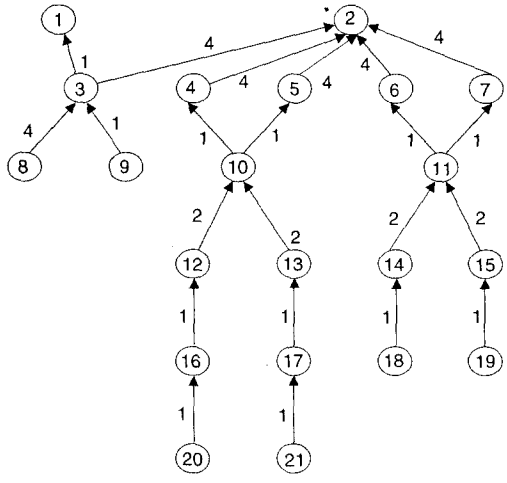

Figure 2. Product structure for Example 2.

EXAMPLE 2. CAPACITATED LOT-SIZING WITH OVERTIMING FOR GENERAL SYSTEM. In this example, the product structure is shown as Figure 2 with $N = 21$, $T = 6$, $K = 2$. This problem is a simplified version of a real-world production environment coming from a manufacturing company. Only the end items 1 and 2 have external demands, and the quantities of demands for both end items are nonzero only in the last period (these quantities are 40 units and 10 units for items 1 and 2, respectively, i.e., $d_{16} = 40$, $d_{26} = 10$, $d_{it} = 0$ for other i and t). All the items have initial stocks of five units (i.e., $I_{i0} = 5$, $\forall i$). The marginal production cost of one unit of each item is constant over time and is assumed to be 100 (i.e., $c_{it} = 100$, $\forall i,t$). The capacity of both

resources occupied by every setup of all the items are constant (i.e., $A_{kit} = 5$ for all k, i, t). All the holding costs, setup cost parameters, and capacity-consuming parameters are constant over time and are shown in Table 4. The available capacity, maximum overtime, and corresponding costs are shown in Table 5.

Table 4. Setup costs, holding costs, and capacity-consuming per unit.

Item i	Setup Cost $s_{it}(= s_i)$	Holding Cost $h_{it}(= h_i)$	Capacity-Consuming $a_{1it}(= a_{1i})$	Capacity-Consuming $a_{2it}(= a_{2i})$
1	1260	7	6	2
2	9765	217	6	8
3	2160	6	5	5
4	2520	14	7	6
5	2520	14	3	9
6	1800	10	8	3
7	1800	10	9	8
8	1440	1	2	2
9	360	1	7	6
10	4680	13	2	3
11	3240	9	3	7
12	2160	3	8	4
13	2160	3	4	6
14	1440	2	5	5
15	1440	2	4	6
16	1440	2	6	2
17	1440	2	4	2
18	720	1	9	4
19	720	1	8	8
20	720	1	9	7
21	720	1	5	4

Table 5. Available capacity, overtime, and corresponding costs.

Period t	1	2	3	4	5	6
Normal capacity C_{1t}	2000	1500	2000	3000	2000	3000
Normal capacity C_{2t}	2000	1500	1800	2400	2000	3000
Maximum overtime U_{1t}	100	300	200	300	180	350
Maximum overtime U_{2t}	200	220	180	240	270	320
Overtime cost o_{1t}	200	200	200	200	200	200
Overtime cost o_{2t}	100	100	100	100	100	100

GAC gives a feasible solution with total cost of 588826 for this problem within ten minutes. Checking the parameters of the problem, we can see that the capacity constraint in this problem is very tight and GAC gives an acceptable solution. We also tested the lot-sizing problems with the same product structure using other input data, and GAC always obtained acceptable solutions in reasonable computing times. For the algorithms SA, TS of [23] and LR of [21] cannot deal with the lot-sizing problems with setup times, we have not provided the similar comparison results as those provided in Example 1.

5. SUMMARY

This paper develops a heuristic genetic algorithm for general capacitated lot-sizing problems. The product structures and capacity constraints can be any type, resources capacity can be ad-

justed by overtiming, each item in the system can have external demands, and the cost parameters (setup costs, holding costs, production costs, overtiming costs) can be time-varying. The experiments show that this method is efficient and effective to solve the general capacitated lot-sizing problems. However, the computational experiments and theoretical analyses presented in this paper are relatively weak, and more computation experiments and theoretical analyses should be made to this kind of heuristic genetic algorithm. For example, optimal control parameters for genetic algorithms such as crossover probability and mutation probability can be identified by more detailed numerical experiments, and it is very attractive to study the behavior of the cost and computation time for the heuristic genetic algorithm as the complexity of the problem increases. The heuristic algorithm proposed in this paper can also be generalized to solve some more complicated lot-sizing problems such as the problems with sequence-dependent setup times and/or sequence-dependent cost parameters. Other genetic operators and selection strategies, such as the uniform crossover and the ranking or tournament selection strategy, can be incorporated into this heuristic genetic algorithm easily. Other more complicated shifting procedures can also be incorporated into this algorithm; for example, we can consider shutting the scheduled items to the next earlier period that already has setups for that item. Finally, incorporating the heuristic genetic algorithm with other metaheuristics to solve the general capacitated lot-sizing and scheduling problems is also very attractive. These will be the potential directions for further research.

REFERENCES

1. H.C. Bahl, L.P. Ritzman and J.N.D. Gupta, Determining lot sizes and resource requirements: A review, *Operations Research* **35**, 329–345, (1987).
2. Y.P. Gupta and Y. Keung, A review of multi-stage lot-sizing models, *International Journal of Operations and Production Management* **10**, 57–73, (1990).
3. J. Maes, J.O. McClain and L.N. Van Wassenhove, Multilevel capacitated lot-sizing complexity and LP-based heuristics, *European Journal of Operational Research* **53**, 131–148, (1991).
4. M. Salomon, *Deterministic Lot-Sizing Models for Production Planning*, Springer-Verlag, Berlin, (1991).
5. R. Kuik and M. Salomon, Batching decision: Structures and models, *European Journal of Operational Research* **75**, 243–263, (1994).
6. S. Helber, Lot-sizing in capacitated production planning and control system, *OR Spektrum* **17**, 5–18, (1995).
7. N.C. Simpson and S.S. Erenguc, Multi-stage production planning research: History and opportunities, *International Journal of Operations and Production Management* **16** (6), 25–40, (1996).
8. A. Drexl and A. Kimms, Lot sizing and scheduling—Survey and extensions, *European Journal of Operational Research* **99** (2), 221–235, (1997).
9. S.S. Erenguc, N.C. Simpson and A.J. Vakharia, Integrated production/distribution planning in supply chains: An invited review, *European Journal of Operational Research* **115** (2), 219–236, (1999).
10. B.J. Coleman, A further analysis of variable demand lot-sizing techniques, *Production and Inventory Management Journal* **33**, 19–24, (1992).
11. A. Aggarwal and J.K. Park, Improved algorithms for economic lot size problems, *Operations Research* **41**, 549–571, (1993).
12. I. Barang, T.J. VanRoy and L.A. Wolsey, Strong formulations for multi item capacitated lot-sizing, *Management Science* **30**, 1255–1261, (1984).
13. G.D. Eppen and R.K. Martin, Solving multi-item capacitated lot-sizing problems using variable redefinition, *Operations Research* **35**, 832–848, (1987).
14. H.O. Guenther, Planning of lot sizes and capacity requirements in a single stage production system, *European Journal of Operational Research* **31**, 223–231, (1987).
15. J. Maes and L.N. Van Wassenhove, Multi-item single-level capacitated dynamic lot-sizing heuristics: A general review, *Journal of the Operational Research Society* **39**, 991–1004, (1988).
16. W.W. Trigeiro, L.J. Thomas and J.O. McClain, Capacitated lot sizing with setup times, *Management Science* **35**, 353–366, (1989).
17. M.R. Lambrecht, J. VanderEechen and H. Vanderveken, Review of optimal and heuristic models for a class of facilities in series dynamic lot size problems, In *Multi-Level Production/Inventory Control Systems: Theory and Practice*, (Edited by L.B. Schwarz), pp. 69–94, North-Holland, Amsterdam, (1981).
18. P. Afentakis, B. Gavish and V. Karmarkar, Optimal solutions to the lot-sizing problem in multi-stage assembly systems, *Management Science* **30**, 222–239, (1984).
19. P. Afentakis, A parallel heuristic algorithm for lot sizing in multistage production systems, *IIE Transactions* **19**, 34–42, (1987).

20. K. Rosling, Optimal lot sizing for dynamic assembly systems, In *Multistage Production Planning and Inventory Control*, (Edited by S. Axsater *et al.*), pp. 119–131, Springer, Berlin, (1986).
21. P.J. Billington, J.O. McClain and L.J. Thomas, Heuristic for multi-level lot-sizing with a bottleneck, *Management Science* **32**, 989–1006, (1986).
22. Y. Roll and R. Karni, Multi-item, multi-level lot sizing with an aggregate capacity constraint, *European Journal of Operational Research* **51**, 73–87, (1991).
23. R. Kuik, M. Salomon, L.N. Van Wassenhove and J. Maes, Linear programming, simulated annealing, and tabu search heuristic for lot sizing in bottleneck assembly systems, *IIE Transactions* **25**, 62–72, (1993).
24. C.E. Heinrich and C. Schneeweiss, Multi-stage lot-sizing for general production systems, In *Multistage Production Planning and Inventory Control*, (Edited by S. Axsater *et al.*), pp. 150–181, Springer, Berlin, (1986).
25. P. Afentakis and B. Gavish, Optimal lot-sizing algorithms for complex product structures, *Operations Research* **34**, 237–249, (1986).
26. H. Tempelmeier and M. Derstroff, A Lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times, *Management Science* **42** (5), 738–757, (1996).
27. A.R. Clark and V.A. Armentano, A heuristic for a resource-capacitated multi-stage lot-sizing problem with lead-time, *Journal of the Operational Research Society* **46**, 1208–1222, (1995).
28. G. Belvaux and L.A. Wolsey, A specialized branch-and-cut system for lot-sizing problems, *Management Science* **46** (5), 724–738, (2000).
29. D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, (1989).
30. L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, (1991).
31. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, (1994).
32. J. Xie, An application of genetic algorithms for general dynamic lotsizing problems, In *Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, Sheffield, UK, pp. 82–87, (1995).
33. N. Dellaert and J. Jeunet, Solving large unconstrained multilevel lot-sizing problems using a hybrid genetic algorithm, *International Journal of Production Research* **38** (5), 1083–1099, (2000).
34. N. Dellaert, J. Jeunet and N. Jonard, A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs, *International Journal of Production Economics* **68** (3), 241–257, (2000).
35. R. Sikora, Genetic algorithm for integrating lot-sizing and sequencing in scheduling a capacitated flow line, *Computers & Industrial Engineering* **30** (4), 969–981, (1996).
36. I. Lee, R. Sikora and M.J. Shaw, A genetic algorithm-based approach to flexible flow-line scheduling with variable lot sizes, *IEEE Transactions on System, Man and Cybernetics (B)* **27** (1), 36–54, (1997).
37. L. Ozdamar and S.I. Birbil, Hybrid heuristics for the capacitated lot-sizing and loading problem with setup times and overtime decisions, *European Journal of Operational Research* **110** (3), 525–547, (1998).
38. L. Ozdamar and G. Barbarosoglu, Hybrid heuristics for the multi-stage capacitated lot sizing and loading problem, *Journal of the Operational Research Society* **50** (8), 810–825, (1999).
39. A. Kimms, A genetic algorithm for multi-level, multi-machine lot sizing and scheduling, *Computers & Operations Research* **26** (8), 829–848, (1999).
40. U. Kohlmorgen, H. Schmeck and K. Haase, Experiences with fine-grained parallel genetic algorithms, *Annals of Operations Research* **90**, 203–219, (1999).
41. Y.-F. Hung, C.-C. Shih and C.-P. Chen, Evolutionary algorithms for production planning problems with setup decisions, *Journal of the Operational Research Society* **50** (8), 857–866, (1999).
42. Y.-F. Hung and K.-L. Chien, Multi-class multi-level capacitated lot sizing model, *Journal of the Operational Research Society* **51** (11), 1309–1318, (2000).
43. N.J. Radcliffe and P.D. Surry, Fundamental limitations on search algorithms: Evolutionary computing in perspective, In *Computer Science Today: Recent Trends and Developments*, (Edited by J. van Leeuwen), pp. 275–291, Springer-Verlag, New York, (1995).
44. D.H. Wolpert and W.G. Macready, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* **1** (1), 67–82, (1997).
45. J. Xie and W. Xing, Incorporating domain-specific knowledge into evolutionary algorithms, *Beijing Mathematics* **4** (2), 131–139, (1998).