Transaction E: Industrial Engineering
Vol. 16, No. 2, pp. 100–113
© Sharif University of Technology, December 2009

SCIENTIA IRANICA

Lot Sizing and Lead Time Quotations in Assembly Systems

F. Kianfar[1] and G. Mokhtari[1,*]

Abstract. *In this paper, a simultaneous lead time quotation and lot sizing problem in an assembly system is investigated. We address a production system with a product that has deterministic demand over a T-period planning horizon and is produced in lots because of the economy of scale. If a lot is completed before the demand period, inventory carrying cost is incurred. On shortages, a lead time is quoted to customers and a lead time quotation cost is incurred. Finally, if the order is delivered later than its due date period, a tardiness cost is charged. The components supply lead time is stochastic, which follows a discrete distribution. The problem is to decide on the lot size of products and components, supply and production starting periods and the due date of lots (to be quoted to customers) so that relevant costs are minimized. The objective function is the sum of the production, inventory carrying, lead time quotation and tardiness costs. We develop a genetic algorithm to solve the proposed model. An experimental framework is set up to test the efficiency of the proposed method, which turns out to rate high, both in terms of cost effectiveness and execution speed.*

Keywords: *Lot-sizing; Lead time quotation; Genetic algorithms; Production planning.*

INTRODUCTION

This paper addresses a combination of lot sizing and lead time quotation problems, which are motivated by a construction equipment manufacturer that could be viewed as an assembly system. In fact, for assembly systems, several types of components are needed to produce one finished product. So, the inventories of the different types of component become dependent. It must be noted that an assembly system could also be called a two-level system.

Lot sizing decisions give rise to the problem of identifying when and how much of a product to produce such that total related costs are minimized. Making the right decisions in lot sizing will affect directly the system performance and its productivity, which are important for a manufacturing firm's ability to compete in the market. We consider a lot sizing problem in a backlogging case, meaning that it is possible to satisfy the demand of the current period in future periods. In the considered production system, on shortages, a lead time must be quoted to the customer, which will be one of the contractual terms. A lead time quotation incurs a cost that arises in the form of sales price reductions, customer goodwill, etc. Because of uncertain supply lead times, the quoted lead time may not be met; a case wherein tardiness cost is incurred. To the best of our knowledge, a combination of the above-mentioned lead time quotations and lot-sizing problems, simultaneously, has not been considered by other researchers.

The lead time quotation problems make practical sense when a firm offers a due date to its customers during sale negotiations and has to offer a price reduction when the due date is far away from the expected one. In order to maintain a good image among the customers, many companies tolerate reasonable holding costs in favor of keeping the established due dates.

In quoting lead times to the customers, a trade-off has to be made between the length and reliability of the lead time. Promising a short lead time might lead to an impossible task for the order realization function, with regard to delivering the order close to the order due date. On the other hand, long lead times make it easy for the order realization function to obtain a good due date performance, but these lead times are, in general, unacceptable to the customers. Long lead times not only dissatisfy customers but also increase WIP and total inventories. Many authors have studied

1. *Department of Industrial Engineering, Sharif University of Technology, Tehran, P.O. Box 11155-9414, Iran.*
*. *Corresponding author. E-mail: mokhtari@behsad.com*

Received 30 June 2007; received in revised form 31 August 2008; accepted 15 November 2008

this trade-off between the length of lead time quoted to the customer and delivery reliability.

In this paper, a multi-period and multi-component lot sizing problem, for assembly systems with uncertain lead times, is investigated. Because of the lead time uncertainty, the traditional backlogging cost has two divisions: the lead time quotation cost and tardiness cost. To our knowledge, this variant of the lot sizing problem has not been considered by other authors.

The rest of the paper is organized as follows. In the following section, some of the relevant work, both in the lot sizing and lead time quotation literature, is summarized. Then, analytical model is presented, the specific problem is described and modeled and, in the final section, we apply genetic algorithms to solve the sample problems and analyze the results.

LITERATURE REVIEW

In this paper, we consider a problem which is simultaneously related to lead time quotation and lot sizing literature. From the lot sizing viewpoint, this problem is a multi-period, two-level, incapacitated lot sizing problem with stochastic lead times and backlogging. Therefore, we address principal research streams in these distinct areas.

Lot Sizing Literature

The concept of batching has been examined in various families of problems. Two major families are lot sizing and lot scheduling. For a review of lot scheduling problems, the reader is referred to Sox et al. [1] and Potts and Kovalyov [2].

Lot sizing problems are production planning problems with setups between production lots. Because of these setups, it is often too costly to produce a given product at every period. On the other hand, generating fewer setups by producing large quantities to satisfy future demand results in high inventory holding costs. Thus, the objective is to determine the periods where production should take place, and the quantities to be produced, in order to satisfy demand, while minimizing production, setup and inventory holding costs. Other costs might also be considered. Examples are backorder cost, changeover cost, etc. Several models have been proposed for lot sizing problems. One of the ancestors of these models is the Economic Order Quantity (EOQ) model, which is a continuous time model with an infinite time horizon. It considers a single item and imposes no capacity restrictions. Later, the EOQ model was extended to consider multiple items and capacity limits. The discrete lot sizing models assume that the planning horizon is finite and divided into discrete periods for which demand is given and may vary between periods. Wagner and Whitin's [3] single item incapacitated problem is a seminal work in this area. Zangwill [4] considered the Wagner and Whitin problem in a backlogging case. For recent reviews of single item and single-level lot sizing problems, the reader is referred to Brahimi et al. [5] and Karimi et al. [6].

Production systems may be single-level or multi-level. In multi-level systems, there is a parent–component relationship between the items. After processing by several operations, raw materials change to end products. The output of an operation (level) is input for another operation. Therefore, the demand at one level depends on the demand for its parents' level. This kind of demand is named dependent demand. Multi-level problems are more difficult to solve than single-level problems. Bahl et al. [7] reviewed the single and multi-level lot sizing papers. Berretta and Rodrigues [8] and Dellaert and Jeunet [9] investigated the multi-level lot sizing problem. Most multi-level lot sizing papers consider zero or deterministic lead times, but Dellaert et al. [10] considered the lot sizing problem with positive deterministic lead times. Clark and Armentano [11] proposed a heuristic for the multi-stage lot-sizing problem with general product structures and lead times. Dellaert and Jeunet [12] studied the impact of positive lead times on the multi-level lot-sizing problem in a rolling schedule environment.

Stochastic lot sizing problems consider the demand or lead time to be stochastic. We review only the stochastic lead time case. With stochastic lead times, a difficulty is added to the problem because, in that case, orders may cross, that is, they may not be received in the same sequence in which they are placed. Hadley and Whitin [13] combined the assumptions that orders cannot cross and lead times are independent. Liberatore [14] considered the single-item lot sizing with deterministic demand and stochastic lead time and avoided the problem of orders crossing in time by assuming that demands are not interchangeable. The non-interchangeability assumption works in such a way that the demand designated for order 1 cannot be satisfied by the early arrival of order 2 since, by assumption, demand 1 can only be satisfied by order 1. Liberatore showed that it is optimal to order for groups of consecutive demands. Nevison and Burstein [15] showed that if lead time distributions are arbitrary (except when they are independent of order size and do not allow orders to cross in time) each order in an optimal solution will exactly satisfy a consecutive sequence of demands; a natural extension of the classic results by Wagner and Whitin. If, on the other hand, orders can cross in time, optimal solutions will still be lumpy, in the sense that each order will satisfy a set (not necessarily consecutive) of the demands.

In our formulation, we assume that lead times are stochastic and demands are not interchangeable.

A practical method for a multi-level lot sizing problem is MRP. In fact, MRP begins with the end-product need date and uses the lead time to calculate the components release date. Clearly, the calculation does not take into account the actual lead time because it is not known at this moment. The calculation uses a forecasted value of lead time, i.e. planned lead time. Gupta and Brennan [16] studied MRP systems using simulation. They showed that lead time uncertainty has a large influence on the cost. The statistics done on simulations by Bragg et al. [17] show that lead times substantially influence the inventories.

Dolgui and Ould-Louly [18] considered the search for the optimal values of the planned lead times for the MRP method under lead time uncertainty and the "Lot for Lot" policy. The problem is to find the planned lead times, which minimize the expected backlogging and holding costs in a two-level assembly system. They suppose infinite supply capacity (lead time does not depend on lot size) and constant demand level. The solution is based on the use of an auxiliary Markov chain. Ould-Louly and Dolgui [19] extended their previous work such that the decision variables were the planned lead times of components and the periodic ordering quantity (with a common periodicity for all of the components). The used criteria is the sum of the average holding cost for the components, the average backlogging cost for the finished product, and the setup cost. A mathematical formulation, based on Markov chains, is proposed to measure the average cost on an infinite horizon with a fixed demand per period. The solution method is given under a complementary assumption. This assumption is that the lead times of the different types of component follow the same distribution probability, and that the holding costs per period of the ordered quantities are the same.

In the considered problem in this paper, the unit backlogging cost is not constant at different periods. Product application is seasonal and shortage cost will be higher at some periods, implying that the planned lead time will not be constant in the planning horizon; for the critical periods, more safety lead time is needed. Therefore, the planned lead time approach could not be applied.

Lead Time Quotation Works

The lead time quotation is also called the due date assignment, especially when combined with order sequencing and scheduling problems (e.g. [20,21]). The majority of papers that consider due date assignment and sequencing together address a single machine shop. Several models of assigning due dates are considered in the scheduling literature, the simplest being the model in which all jobs have a common due date. Gordon et al. [22] provided a unified framework of common due date assignment problems in the deterministic case by surveying the literature concerning models involving a single machine and parallel machines.

When there is a significant setup time and/or setup cost, it is useful to consider problems where three types of decision are combined: the scheduling, batching, and due date assignment. Cheng and Kovalyov [23] considered such a problem in a common due date case in a single machine system, wherein a set-up time is required before the batch of a group is processed.

But, in the case of complex assemblies, a customer order due date assignment is a higher level decision in the production planning hierarchy, which could be combined with shop floor scheduling problems. A due date assignment must be set at the MPS level in coordination with the marketing and sales function.

Some studies use simulation and regression to analyze the performance of manufacturing systems under different due date setting regimes (e.g. [24,25]). An overview of the studies which examine (using simulation) the relative performance of simple heuristic rules for the due date assignment can be found in the survey by Cheng and Gupta [26]. Another approach is the use of flow time for due date assignment; each order has a flow time that usually follows a probability distribution function (e.g. [27-31]). A queuing theory has also been used in due date assignment literature. For example, Wein [32] considered the simultaneous due date setting and priority sequencing. Duenyas [33] considered the due date quotation in a production system modeled as a single server queue. Slotnick and Sobel [34] modelled the due date assignment as a semi-Markov decision process.

From the lead time quotation viewpoint, we address a situation where, upon shortages, a lead time is quoted to the customer and a cost that is proportional to the quoted lead time is incurred. If products are delivered with tardiness, a tardiness cost, which is proportional to delivery lateness, is charged.

MODEL FORMULATION

We consider the problem of having a dynamic external demand for one finished product. The demand for this product is known exactly for number T of periods; the demand window. The finished product demands are satisfied at the end of each period. In order to reduce set-up costs, the demand for several periods can be lumped at the cost of a carrying charge. We assume that the finished product is assembled from sub-assemblies or components. Like the product, these components will have a replenishment lead-time or production lead-time, ordering or set-up costs and holding costs, for carrying the inventory over some periods. The lot sizing decisions for the product completely determine the requirements for components with a lead-time correction. The assembly system capacity is supposed infinite. The lead times of component types are independent random variables (each component type has its own lead time, which is a random variable). The lead time (the number of periods from order placement until order arrival) is independent of whether an order is placed at any particular period or the amount of any order.

Our objective is to minimize the sum of the set-up, and inventory holding costs (for product and components) as well as lead time quotations and tardiness costs for the finished product over the T-periods horizon.

Notation and Formulation

We follow the Krarup and Bilde [35] approach, which is called the facility location-based formulation and the disaggregate formulation too. They presented a reformulation for a single-item, single-level, deterministic lot sizing problem by introducing new variables, x_{ij}, representing the quantity produced in time period i to satisfy the demand in time period j for all $j \in \{i, \cdots, T\}$. The difference of our formulation is due to backlogging; in $x_{ij} : j \in \{1, \cdots, T\}$.

To formulate the problem, we use the symbols summarized in Table 1.

We can mathematically formulate this problem as follows:

$$\min Z = \sum_{t=1}^{T} \Bigg[\alpha_t R_t + \sum_{j=1}^{T} (\beta_t + u_{jt} h + \pi_{2t} v_{jt}) X_{jt} + \pi_{1t} d_t K_t + \sum_{i=1}^{m} \Bigg(a_{it} S_{it} + \sum_{j=1}^{t} (b_{it} + (\delta + e_{ijt}) h_i) Y_{ijt} \Bigg) \Bigg], \tag{1}$$

st:

Table 1. Notations for the problem.

t, j	Period indices $j = 1, \cdots, T \quad t = 1, \cdots, T$
i	Component indicator $i = 1, \cdots, m$
Decision Variables:	
X_{jt}	Batch size of finished product, started in period j to fulfill the demand of period t
Y_{ijt}	Batch size of component i, started in period j to fulfill the requirements of period t
K_t	Lead time quoted to customers of period t
R_j	Boolean variable addressed to capture set-up cost in period j
S_{ij}	Boolean variable addressed to capture set-up cost of component i in period j
Parameters:	
h_i	Holding cost of the component i per period
h	Unit inventory carrying cost per period (for finished product)
$\alpha_t + \beta_t n$	Production cost of a finished product batch with size n released in period t
$a_{it} + b_{it} n$	Production cost of a batch of component i with size n released in period t
δ	Assembly lead time (deterministic), measured as an integer multiple of planning periods
L_i	Supply lead time of component i (a discrete random variable), measured as an integer multiple of planning periods
γ_i	Quantity of component i required to produce one unit of finished product
π_{2t}	One period tardiness cost per unit of finished product
π_{1t}	Cost of quoting one period lead time for each unit demanded in period t
d_t	Demand of period t
Auxiliary Variables:	
u_{jt}	Expected inventory carrying periods of batch X_{jt}
v_{jt}	Expected tardiness of batch X_{jt}
e_{ijt}	Expected inventory carrying periods of batch Y_{ijt}
W_j	Delay in starting assembling of X_{j*} batches (because of late receiving of required components)
w_j^i	Delay in starting assembling of X_{j*} batches (because of late receiving of required components other than component i)

$$\sum_{j=1}^{T} X_{jt} = D_t, \qquad t = 1, \cdots, T, \tag{2}$$

$$\sum_{j=1}^{T} X_{tj} \leq \text{M.R}_t, \qquad t = 1, \cdots, T, \tag{3}$$

$$\sum_{j=1}^{T} Y_{itj} \leq \text{M.S}_{it}, \qquad t=1, \cdots, T, \qquad i=1, \cdots, m, \tag{4}$$

$$\sum_{j=1}^{t} Y_{ijt} = \gamma_i \sum_{j=1}^{T} X_{tj}, \qquad t=1, \cdots, T, \quad i=1, \cdots, m, \tag{5}$$

$$R_t, S_{it} = 0 \text{ or } 1: \; t = 1, \cdots, T, \qquad i = 1, ..., m, \tag{6}$$

$$K_t \geq 0, \qquad \text{Integer}: t = 1, \cdots, T, \tag{7}$$

$$X_{jt}, Y_{ijt} \geq 0 : j = 1, \cdots 5, t,$$

$$t = 1, \cdots, T, \quad i = 1, \cdots, m. \tag{8}$$

The proposed model is a Mixed Integer Nonlinear Programming (MINLP) model, as it contains both continuous and integer variables. The objective function in Equation 1 is the sum of production, lead time quotation, tardiness, set-up and inventory holding costs over the planning horizon. Note that the possibility of time-varying unit production and set-up costs is allowed.

u_{jt} and v_{jt} are defined by:

$$u_{jt} = \sum_{r=0}^{t-j+K_t-\delta} (t - j + K_t - \delta - r).P(w_j = r), \tag{9}$$

$$v_{jt} = \sum_{r=t-j+K_t-\delta}^{\infty} [r - (t - j + K_t - \delta)].P(w_j = r). \tag{10}$$

In Equations 9 and 10, $t - j + K_t - \delta$ is the floating time of batch X_{jt}. Similarly, we have:

$$e_{ijt} = \sum_{r=0}^{\infty} \sum_{l=0}^{t-j+r} (t-j+r-l).P(w_t^i = r).P(L_i = l). \tag{11}$$

w_j and w_j^i are random expressions and are defined by:

$$P(w_j = r) = P(w_j \leq r) - P(w_j \leq r - 1), \tag{12}$$

$$P(w_j \leq r)$$

$$= \prod_{n=1,\cdots,m, \; t=1,\cdots,j, \; Y_{ntj}>0} P(L_n \leq j - t + r), \tag{13}$$

$$P(w_j^i \leq r) = \prod_{n=1,\cdots,m, \; t=1,\cdots,j, \; Y_{ntj}>0, \; n\neq i} P(L_n \leq j - t + r). \tag{14}$$

Equation 2 expresses the demand satisfaction constraint for finished product. Constraints 3 and 4, where M is a large number, guarantes that a set-up cost will be incurred when a batch is purchased or produced. The sum of supply of a component in a period must be equal to its requirement in the finished product assembly, as stated in Equation 5.

THE GENETIC ALGORITHM

In the past decade, meta-heuristics such as Genetic Algorithms (GA), tabu search and simulated annealing have become more and more popular for solving complex combinatorial problems. Readers not familiar with the basic concepts of genetic algorithms are referred to Goldberg [36]. General guidelines for the design of meta-heuristics are discussed in Hertz and Widmer [37]. One of the main reasons for the success of these meta-heuristics is their flexibility and capacity to handle large and complex problems. As a consequence, these methods are usually developed for extensions of the standard lot sizing problem, for which no good special purpose algorithm exists and which is too difficult to solve with commercial integer optimization software. For a recent review on the applications of meta-heuristics for lot sizing problems, see Jans and Degraeve [38].

For a GA, the major decisions concern:

1. Representation,
2. Evaluation,
3. Construction of genetic operators to generate offspring,
4. Choice of selection mechanism to determine the next population.

In this section, we present the way a genetic algorithm can be customized to address the problem. We first discuss the issue of encoding and the feasibility constraints. Then, we turn to the evolutionary stages of the algorithm and the specific genetic operators that have been designed to increase search efficiency.

Encoding

In the representation of a solution of lot sizing problems, there are two basic options for a direct representation, as we are dealing with mixed integer programming problems containing both integer and continuous variables [38]. In the first option, we explicitly include both integer variables and production quantities in the representation. The second option is to include only integer variables and discard the production quantities. Each setting of the integer variables corresponds to an optimal value for the production variables.

As discussed in previous sections, we know that the demand of each period will be satisfied by only one order. There is never more than one lot to cover a demand or a requirement. In other words, among all X_{jt}, $j = 1, \cdots, T$ for period t, there is only one j, such that $X_{jt} > 0$. We design the chromosome structure as an integer matrix (with T columns), based on this property:

$$\begin{bmatrix} P_{1,1} & \ldots & P_{1,T} \\ P_{2,1} & \ldots & P_{2,T} \\ P_{3,1} & \ldots & P_{3,T} \\ \ldots & \ldots & \ldots \\ P_{2+m,1} & \ldots & P_{2+m,T} \end{bmatrix}. \tag{15}$$

In the first row, $P_{1,t}$ equals the period within which the assembly of a finished product batch is released to satisfy the demand of period t. The first row data could be converted to X_{jt}, R_j variables:

$$X_{jt} = \begin{cases} d_t, & P_{1,t} = j \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$R_j = \begin{cases} 1, & \ni t : P_{1,t} = j \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

In the second row of the matrix, $P_{2,t}$ is the lead time quoted to the customers in period t. Each row of the next m rows corresponds to a component. For example, $P_{3,t}$ equals the period in which an order of component 1 is released to satisfy its requirements in period t. This data could be converted to S_{ij}, Y_{ijt} variables:

$$S_{ij} = \begin{cases} 1, & \ni t : P_{2+i,t} = j \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

$$Y_{ijt} = \begin{cases} \gamma_i \sum\limits_{\forall n : P_{1,n} = t} d_n, & P_{2+i,t} = j \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

For a solution to be feasible, the following restrictions must be considered in the corresponding chromosome:

a) $P_{1,t} > 0$ if, and only if, $d_t > 0$ for $t = 1, \cdots, T$, (20)

b) $P_{2+i,t} \leq t$ for $t = 1, \cdots, T$, $i = 1, \cdots, m$, (21)

c) $P_{2+i,t} > 0$ if, and only if, $\ni j : P_{1,j} = t$

for $t = 1, \cdots, T$. (22)

In our genetic algorithm, only feasible chromosomes will be considered. Therefore, these restrictions are considered in developed crossover and mutation operators.

A set of such chromosomes is generated randomly to form an initial population. The fitness of each chromosome in the population is evaluated based on the total cost of the plan it represents. A new population of the same size is generated from the original population by manipulating its chromosomes using genetic operators. The new population is evaluated and another population is generated from it. This process continues until a stopping criterion is met. Genetic operators generate new chromosomes (children) from existing ones (parents) by manipulating the values assigned to the genes (digits) of some chromosomes (that are chosen randomly) in two ways: crossover and mutation. Every crossover operator is applied to two chromosomes (parents) and results in two new ones (children). Every mutation operator is applied to one chromosome and results in a different chromosome.

Crossover

Crossover is the means by which two different chromosomes (i.e. members of the population) can combine to form some new "offspring". We tested two crossover operators.

Period Crossover [10]

The classical crossover is a one-point crossover that combines two chromosomes on the basis of one cross site randomly selected. Genetic material is swapped between two chromosomes to produce a pair of offspring. Our crossover is a period crossover that chooses a point in time, t^*, randomly selected in the planning horizon. To produce offspring, we combine the first periods of one parent's strings with the last periods of the second parent's strings, the appropriate corrections being made when lead times have to be incorporated (i.e. when $P_{1,t}$ is selected from a parent, $P_{2+i,j}$, $j = P_{1,t}$, $i = 1, \cdots, m$ will be selected from that parent too).

Uniform Crossover

In the uniform crossover, for each gene position, a decision (based on a random number) is made on whether to swap the genes of the two parents at that position (Y) or not (N). So, the child is produced in the following manner:

- For each period t, $P_{2,t}$ is selected from a parent by random.
- For each period t, $P_{1,t}$ is selected from a parent by random.
- If $P_{1,t}$ of the two parents are the same, then $P_{2+i,j}$, $j = P_{1,t}$ will be selected randomly from a parent for each item i, $i = 1, \cdots, m$. Otherwise, $P_{2+i,j}$ is copied from the parent from which $P_{1,t}$ was selected.

Mutation

The role of mutation in GA is to restore lost or unexpected genetic material into a population to prevent the premature convergence of GA to sub-optimal solutions. Any good search algorithm must explore a large search space in the beginning and the search should then narrow down as it converges to the solution.

Mutation can possibly trigger a series of changes in the chromosome in order to maintain its feasibility, i.e. the restrictions mentioned in Equations 20 and 22 must be satisfied. For example, if the mutated gene is from the first matrix row and $P_{2+i,j}$, $j = P_{1,t}$ are zero, then, $P_{2+i,j}$, $j = P_{1,t}$ are assigned randomly to maintain solution feasibility.

We applied four mutation operators as follows.

Single-Bit Mutation

In the single-bit mutation, a unique gene undergoes mutation. A single gene is randomly selected and its value is changed by a random amount.

Multi-Bit Mutation

In multi-bit mutation, each gene is mutated with a specific probability. Therefore, several genes of the mutating chromosome may be changed.

The encoding matrix cells are classified in three groups: $P_{1,t}$ (the first row), $P_{2,t}$ (the second row), and $P_{2+i,t}$ (rows from 2 to 2+m). The mutation probability of each gene depends on its group. So, this operator has three probability values as parameters.

Non-Uniform Mutation

The term "non-uniform" is derived from the Real Coded Genetic Algorithms (RCGA) literature. In RCGA, non-uniform means that the size of the gene generation interval shall be lower with the passing of generations [39]. The gene generation interval is proportional to $\Gamma(t)$, which is defined by:

$$\Gamma(t) = 1 - a^{\left(1-\frac{t}{t_{\max}}\right)^{b}}, \tag{23}$$

where a is a random number in [0, 1], b a constant parameter, t the time or generation number and $t_{\max}$ the maximum number of generations the RCGA is allowed to run. This function gives values in the range [0, 1] such that the probability of returning a number close to zero increases as the algorithm advances.

In this paper, the term "Non-uniform" means that the probability of mutation is not uniform during the algorithm execution. The probability of mutation is calculated by $\Gamma(t)$ and it decreases in the final stages of GA. This property causes this operator to make a uniform search in the initial space when t is small and very locally at a later stage, favoring local tuning.

Non-uniform mutation is done by selecting a random period. Starting from the chosen period and moving to the end or to the beginning of the chromosome (selected randomly), all the genes are changed to a random amount.

Single-Bit Inversion

This mutation operator mutates a cell (that is selected randomly) from the first two rows of the encoding matrix. The value of the selected gene is changed to an adjacent non-zero gene. If both of neighboring genes are non-zero, one of them is selected randomly.

Reproduction

This is one of the most important and, surprisingly, least controversial of the operations. This is where it is decided which members of the population will be allowed to survive, and which will perish. It is usually done via a weighted random selection. The weighting is done on the fitness of each individual.

That is, the more fit members of the population have a greater chance of progressing to the next generation than those less fit.

Elitism is a technique to preserve and use previously found best solutions in subsequent generations of GA. In an elitist GA, the statistic of the population's best solutions cannot degrade with generation. Maintaining archives of non-dominated solutions is an important issue. The final contents of the archive represent (usually) the result returned by the optimization process.

In this paper, we apply a tournament selection method. In tournament selection, a number, Tour, of individuals is chosen randomly from the population and the best individual from this group is selected as a parent. This process is repeated as often as there are individuals to choose. The parameter for tournament selection is the tournament size, Tour. Tour takes values ranging from 2 - popsize (number of individuals in the population).

Experimental Setup

Since the proposed model is a new and original model, no standard test problems could be found for it, neither in the literature nor in the OR websites. Thus, we developed a problem generator program to generate problems randomly. Four different structures were used in generating random problems. These structures are given in Table 2. Ten random problems were generated using each structure, which means that 40 random problems were generated.

We need to assume a distribution function for the supply lead time of each item. For this purpose, three discrete random variables were considered. Table 3

Table 2. Parameters range in four structures.

Parameter	Structure 1	Structure 2	Structure 3	Structure 4
T	Uniform [6,20]	Uniform [6,20]	Uniform [6,20]	Uniform [6,20]
m	Uniform [5,30]	Uniform [5,30]	Uniform [5,30]	Uniform [5,30]
α_t	Uniform [1,6]	Uniform [10,15]	Uniform [1,6]	Uniform [10,15]
β_t	Uniform [10,15]	Uniform [1,6]	Uniform [10,15]	Uniform [1,6]
a_{it}	Uniform [3,8]	Uniform [6,11]	Uniform [3,8]	Uniform [6,11]
b_{it}	Uniform [6,11]	Uniform [3,8]	Uniform [6,11]	Uniform [3,8]
π_{1t}	Uniform [7,12]	Uniform [7,12]	Uniform [7,12]	Uniform [7,12]
π_{2t}	Uniform [10,15]	Uniform [10,15]	Uniform [10,15]	Uniform [10,15]
d_t	Uniform [50,150]	Uniform [50,150]	Uniform [50,150]	Uniform [50,150]
γ_i	Uniform [1,4]	Uniform [1,4]	Uniform [1,4]	Uniform [1,4]
h_i	Uniform [1,6]	Uniform [1,6]	Uniform [7,12]	Uniform [7,12]
h	8	8	8	8
δ	0	0	0	0

shows the probability mass function of these variables. In the problem generator program, a probability mass function is randomly chosen and assigned to each item.

A computer program was developed using Turbo Pascal programming language to implement GA. This program includes 14 functions for calculating the fitness of individuals and 5 functions and 22 procedures for reading test problems (the genetic algorithm itself) and creating output data files. Using modular programming, we ensure program traceability and reliability.

The genetic algorithm requires a number of parameters to be specified. We kept the population size equal to 70 chromosomes in most experiments. The number of generations was 700. In tournament selection, we tested the algorithm performance with Tour $\in \{2, 3, 4\}$. Elitism size was set to 1.

When the best-known solutions are not available for comparison, they could also be obtained with the meta-heuristic itself, either by using different starting points or allowing for long runs [38]. We used this approach in this paper. Each test problem was solved almost 1100 times (in all of the algorithms) and the best solution found in these runs was used as an (approximately) optimum solution.

Table 3. Probability mass functions of supply lead time.

	Lead Time (as an integer multiple of planning periods)				
	0	1	2	3	4
Random Variable 1	0.1	0.3	0.4	0.1	0.1
Random Variable 2	0.2	0.4	0.2	0.1	0.1
Random Variable 3	0.6	0.1	0.1	0.1	0.1

Experimental Results

This section is devoted to the presentation and discussion of experimental data produced in the test phases. 111 different algorithms were designed and tested using 40 test problems to select the appropriate algorithm.

The configurations of tested algorithms are presented in Table 4. In this table, the ID column denotes the algorithm code; the Tour column denotes the tournament selection parameter and the Gens column shows the total number of solutions generated in each algorithm. If an algorithm uses more than one crossover or mutation operator, the probability of applying each operator is given in parentheses. At most, one mutation operator is applied to each offspring. For example, in algorithm G92, period and uniform crossover operators are used with probability of 0.3 and 0.5, respectively.

In Table 4, the terms NU-Random and Inversion denote non-uniform mutation and single-bit inversion, respectively. The non-uniform mutation was used uniformly in some of the algorithms, which are shown by U-Random.

Each algorithm was run 10 times for each test problem using 3 computers with the following configuration:

CPU: AMD(tm) 64 $\times$ 2 Dual Core Processor 3600+ 2.01 GHz,

RAM: 1 GB

In each run of each algorithm, the following information was reported and analyzed:

- Average fitness of the initial population,
- Standard deviation of the initial population,

Table 4. Algorithms design.

Row	ID	Crossover	Mutation	Tour	Gens
1	G111	Uniform-Cross (0.8)	Single-Bit (0.25)	3	49000
2	G46	Uniform-Cross (0.8)	Single-Bit (0.25)	2	49000
3	G89	Uniform-Cross (0.8)	Single-Bit (0.7)	3	49000
4	G103	Period-Cross (0.8)	Single-Bit (0.5)	3	49000
5	G105	Uniform-Cross (0.8)	Single-Bit (0.5)	3	49000
6	G92	Period-Cross (0.3), Uniform-Cross (0.5)	Single-Bit (0.7)	3	49000
7	G99	Uniform-Cross (0.8)	Single-Bit (0.8)	3	49000
8	G102	Uniform-Cross (0.8)	Multi-Bit (0.01,0.01,0.02)	3	49000
9	G104	Period-Cross (0.8)	Multi-Bit (0.01,0.01,0.02)	3	49000
10	G106	Uniform-Cross (0.8)	Multi-Bit (0.02,0.02,0.05)	3	49000
11	G90	Uniform-Cross (0.8)	Multi-Bit (0.03,0.03,0.01)	3	49000
12	G76	Uniform-Cross (0.8)	Multi-Bit (0.05,0.05,0.02)	3	49000
13	G70	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 2$), Single-Bit (0.5)	3	49000
14	G97	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.5)	3	49000
15	G83	Uniform-Cross (0.8)	NU-Random ($b = 1$), Single-Bit (0.5)	3	49000
16	G75	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.5)	3	63000
17	G80	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5)	3	42000
18	G81	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5)	3	35000
19	G82	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5)	3	28000
20	G88	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.7)	3	49000
21	G84	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5)	3	21000
22	G85	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5)	3	14000
23	G86	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5)	3	49000
24	G87	Period-Cross (0.8)	NU-Random ($b = 0.5$), Single-Bit (0.5)	3	49000
25	G63	Period-Cross (0.8)	NU-Random ($b = 1$), Single-Bit (0.3), Inversion (0.2)	3	49000
26	G65	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.5), Inversion (0.2)	2	49000
27	G67	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.5), Inversion (0.2)	2,3	49000
28	G69	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.3), Inversion (0.2)	3	49000

Table 4. Algorithm design (continued).

Row	**ID**	**Crossover**	**Mutation**	**Tour**	**Gens**
29	G36	Uniform-Cross (0.8)	Multi-Bit (0.05,0.05,0.02) 0.1, U-Random (0.1), Inversion (0.05), Single-Bit (0.1)	3	49000
30	G38	Period-Cross (0.1), Uniform-Cross (0.8)	Multi-Bit (0.05,0.05,0.02) 0.02, U-Random (0.1), Inversion (0.02), Single-Bit (0.2)	2	49000
31	G71	Period-Cross (0.3), Uniform-Cross (0.5)	U-Random (0.10), Single-Bit (0.25), Inversion (0.15)	3	49000
32	G73	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.3), Inversion (0.2) Pop-Size=100	3	49000
33	G74	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 1$), Single-Bit (0.3), Inversion (0.2) Pop-Size=50	3	49000
34	G95	Uniform-Cross (0.8)	NU-Random ($b = 1$), Single-Bit (0.5), Inversion (0.2)	3	49000
35	G96	Uniform-Cross (0.8)	NU-Random ($b = 1$), Single-Bit (0.3), Inversion (0.2)	3	49000
36	G98	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.3), Inversion (0.2)	3	49000
37	G91	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5), Inversion (0.15)	3	49000
38	G101	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5), Inversion (0.15)	3	42000
39	G107	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5), Inversion (0.15)	3	35000
40	G109	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5), Inversion (0.15)	3	28000
41	G110	Period-Cross (0.3), Uniform-Cross (0.5)	NU-Random ($b = 0.5$), Single-Bit (0.5), Inversion (0.15)	3	21000

- Average fitness of the last population,
- Standard deviation of the last population,
- Average of all of the solutions generated throughout the algorithm,
- The generation which found the best solution,
- The best solution found,
- Run time of each problem.

The results of the algorithms performance are given in Table 5. Table 6 shows the average and maximum run time for each run of each problem.

Comparing the average fitness of the initial and last population shows the improvement made by the GA. The average and standard deviation of the first population fitness for most of the algorithms is about 1.22 and 12000, respectively. In other words, the average discrepancy of the first populations from optimum solutions is approximately 22-27%. This is reduced to 1-3% for the last populations.

The standard deviation of the initial and last population was used to analyze the diversity of populations. Smaller standard deviation means more similar solutions exist in the population. Standard deviation information, along with the generation number which found the best solution, shows whether the algorithm has a premature convergence. For example, if the average number of generations (for which test problem and algorithm find the best solution) is about 200, it means that this algorithm has a rapid convergence to a solution (note that the total generations is 700).

The best solutions are generated by algorithm G91 with the following specifications:

- Crossover: Period-Cross (0.3), Uniform-Cross (0.5),
- Mutation: NU-Random ($b = 0.5$), Single-Bit (0.5), Inversion (0.15),
- Selection: Tournament Selection (with 3 participants),

Table 5. Algorithms results.

Row	ID	Solution Discrepancy from Optimum	Final Solution Generation	Population Discrepancy from Optimum			Population Standard Deviation	
				First	Last	Total	First	Last
1	G111	0.727%	255	22.856%	1.072%	1.782%	12026	4376
2	G46	0.922%	327	26.809%	1.312%	2.480%	11569	3477
3	G89	0.652%	186	22.804%	1.772%	2.290%	12245	7032
4	G103	0.886%	214	22.858%	1.607%	2.364%	12318	6295
5	G105	0.661%	175	22.804%	1.414%	1.954%	12272	6142
6	G92	0.758%	201	22.839%	1.830%	2.414%	12175	7036
7	G99	0.760%	201	22.808%	2.157%	2.634%	12216	7523
8	G102	0.873%	184	22.819%	1.584%	2.111%	12149	7365
9	G104	0.934%	231	22.835%	1.641%	2.340%	12321	7274
10	G106	0.988%	269	22.814%	3.063%	3.563%	12213	11896
11	G90	1.087%	187	26.767%	2.115%	2.848%	11473	7470
12	G76	0.923%	241	27.166%	3.049%	3.694%	11725	9870
13	G70	0.699%	325	27.263%	1.500%	5.952%	12034	5088
14	G97	0.714%	375	22.787%	1.443%	6.557%	12175	6464
15	G83	0.758%	365	22.825%	1.487%	6.594%	12144	6526
16	G75	0.692%	421	22.808%	1.410%	6.415%	12193	6601
17	G80	0.747%	386	22.798%	1.498%	8.004%	12219	6741
18	G81	0.779%	339	22.833%	1.558%	8.185%	12223	6429
19	G82	0.845%	300	22.783%	1.561%	8.412%	12258	5896
20	G88	0.740%	401	22.849%	1.888%	6.994%	12232	7725
21	G84	0.873%	245	22.783%	1.609%	8.731%	12046	6251
22	G85	1.109%	177	22.824%	1.848%	9.406%	12209	5378
23	G86	0.648%	419	22.808%	1.401%	7.850%	12191	6713
24	G87	0.800%	434	22.803%	1.548%	8.096%	12092	6300
25	G63	1.160%	428	26.897%	1.584%	7.911%	11753	4113
26	G65	0.979%	574	27.174%	1.953%	11.320%	11884	5393
27	G67	0.821%	503	27.227%	1.661%	9.224%	12330	5423
28	G69	0.762%	415	27.149%	1.206%	7.361%	11921	4409
29	G36	1.063%	202	26.639%	3.174%	3.824%	11209	24727
30	G38	1.132%	267	27.923%	3.421%	4.600%	11459	24469
31	G71	0.961%	301	26.877%	3.157%	3.963%	11747	26727
32	G73	0.928%	302	26.111%	1.395%	8.817%	11367	4161
33	G74	0.950%	526	26.183%	1.413%	8.611%	11408	4469
34	G95	0.772%	347	22.785%	1.545%	6.570%	12224	6567
35	G96	0.742%	378	22.777%	1.165%	6.272%	12171	5032
36	G98	0.784%	403	22.786%	1.197%	7.634%	12142	5042
37	G91	0.610%	424	22.821%	1.347%	7.819%	12145	6253
38	G101	0.629%	385	22.804%	1.386%	7.984%	12282	6592
39	G107	0.758%	332	22.801%	1.468%	8.162%	12177	5919
40	G109	0.831%	298	22.808%	1.554%	8.376%	12337	6424
41	G110	0.937%	245	22.818%	1.613%	8.748%	12105	5634

Table 6. Run times (for each run of each problem).

Row	Algorithm	Average Run Time (minutes)	Maximum Run Time (minutes)
1	G111	0.3075	0.85
3	G89	0.285	0.7
4	G103	0.2875	0.752
5	G105	0.2925	0.753
6	G92	0.2825	0.7
7	G99	0.2825	0.75
8	G102	0.3025	0.75
9	G104	0.2925	0.71
10	G106	0.3025	0.72
14	G97	0.4	1
15	G83	0.3975	1
16	G75	0.5075	1.2
17	G80	0.3775	0.95
18	G81	0.3225	0.75
19	G82	0.2625	0.65
21	G84	0.205	0.5
22	G85	0.145	0.35
23	G86	0.4118	1
24	G87	0.4325	1.05
34	G95	0.4	1
35	G96	0.4125	1
36	G98	0.4375	1.15
37	G91	0.435	1.1
38	G101	0.3775	0.9
39	G107	0.325	0.85
40	G109	0.2625	0.65
41	G110	0.205	0.52

- Generations Number: 700.

CONCLUSION

This work presents a new and original mathematical model for lot sizing in a two-level assembly system. This model determines lot sizes and the periods within which supplies will start. This model shows better performance in production systems where supply and manufacturing lead times are relatively high and stochastic. The structure of the model was based on a manufacturing system which produces construction equipment. Construction equipment manufacturing is an industry with high (and stochastic) lead times. The results show that stochastic lead times could not be ignored in such cases.

A genetic algorithm is developed to solve the model. An innovative mutation operator (non-uniform) is combined with single-bit mutation to avoid the local minimum trap. On the basis of the test runs, the genetic algorithm works satisfactorily.

Possible extensions to this work could be classified in modeling and solution methods. In the modeling view, the proposed model could be extended to consider capacities as a constraint. In the solution method view, a parallel genetic algorithm may show better performance than a standard genetic algorithm.

REFERENCES

1. Sox, C.R. et al. “A review of the stochastic lot scheduling problem”, *Int. J. Production Economics*, **62**, pp. 181-200 (1999).
2. Potts, C.N. and Kovalyov, M.Y. “Scheduling with batching: A review”, *European Journal of Operational Research*, **120**, pp. 228-249 (2000).

3. Wagner, H.M. and Whitin, T.M. "Dynamic version of the economic lot size model", *Management Science*, **5**(1), pp. 89-96 (1958).

4. Zangwill, W.I. "A deterministic multi-period production scheduling model with backlogging", *Management Science*, **13**(1), pp. 105-119 (1966).

5. Brahimi, N. et al. "Single item lot sizing problems", *European Journal of Operational Research*, **168**, pp. 1-16 (2006).

6. Karimi, B. et al. "The capacitated lot sizing problem: A review of models and algorithms", *Omega*, **31**(5), pp. 365-378 (2003).

7. Bahl, H.C. et al. "Determining lot sizes and resource requirements: A review", *Operations Research*, **35**(3), pp. 329-345 (1987).

8. Berretta, R. and Rodrigues, L.F. "A memetic algorithm for a multistage capacitated lot-sizing problem", *Int. J. Production Economics*, **87**, pp. 67-81 (2004).

9. Dellaert, N. and Jeunet, J. "Randomized multi-level lot-sizing heuristics for general product structures", *European Journal of Operational Research*, **148**, pp. 211-228 (2003).

10. Dellaert, N. et al. "A genetic algorithm to solve the general multi-level lot-sizing problem with time-varying costs", *Int. J. Production Economics*, **68**, pp. 241-257 (2000).

11. Clark, A.R. and Armentano, V.A. "A heuristic for a resource-capacitated multi-stage lot-sizing problem with lead times", *The Journal of the Operational Research Society*, **46**(10), pp. 1208-1222 (1995).

12. Dellaert, N. and Jeunet, J. "An alternative to safety stock policies for multi-level rolling schedule MRP problems", *European Journal of Operational Research*, **163**, pp. 751-768 (2005).

13. Hadley, G. and Whitin, T.M. "A family of dynamic inventory models", *Management Science*, **8**, pp. 458-469 (1962).

14. Liberatore, M.J. "Planning horizons for a stochastic lead time inventory model", *Operations Research*, **25**(6), pp. 977-988 (1977).

15. Nevison, C. and Burstein, M. "The dynamic lot-size model with stochastic lead times", *Management Science*, **30**(1), pp. 100-109 (1984).

16. Gupta, S.M. and Brennan, L. "MRP systems under supply and process uncertainty in an integrated shop floor control", *International Journal of Production Research*, **33**, pp. 205-220 (1995).

17. Bragg, D.J. et al. "The effects of partial order release and component reservation on inventory and customer service performance in an MRP environment", *International Journal of Production Research*, **37**, pp. 523-538 (1999).

18. Dolgui, A. and Ould-Louly, M.A. "A model for supply planning under lead time uncertainty", *Int. J. Production Economics*, **78**, pp. 145-152 (2002).

19. Ould-Louly, M.A. and Dolgui, A. "The MPS parameterization under lead time uncertainty", *Int. J. Production Economics*, **90**, pp. 369-376 (2004).

20. Xia, Y. et al. "Job sequencing and due date assignment in a single machine shop with uncertain processing times", *European Journal of Operational Research*, **184**, pp. 63-75 (2008).

21. Shabtay, D. and Steiner, G. "Two due date assignment problems in scheduling a single machine", *Operations Research Letters*, **34**, pp. 683-691 (2006).

22. Gordon, V. et al. "A survey of the state-of-the-art of common due date assignment and scheduling research", *European Journal of Operational Research*, **139**, pp. 1-25 (2002).

23. Cheng, T.C.E. and Kovalyov, M.Y. "Batch scheduling and common due-date assignment on a single machine", *Discrete Applied Mathematics*, **70**, pp. 231-245 (1996).

24. Veral, E.A. and Mohan, R.P. "A two-phased approach to setting due-dates in single machine job shops", *Computers & Industrial Engineering*, **36**, pp. 201-218 (1999).

25. Veral, E.A. "Computer simulation of due-date setting in multi-machine job shops", *Computrs & Industrial Engineering*, **41**, pp. 77-94 (2001).

26. Cheng, T.C.E. and Gupta, M.C. "Survey of scheduling research involving due date determination decisions", *European Journal of Operational Research*, **38**, pp. 156-166 (1989).

27. Bertrand, J.W.M. and Ooijen, H.P.G.V. "Customer order lead times for production based on lead time and tardiness costs", *Int. J. Production Economics*, **64**, pp. 257-265 (2000).

28. Hegedus, M.G. and Hopp, W.J. "Due date setting with supply constraints using MRP", *Computers & Industrial Engineering*, **39**, pp. 293-305 (2001).

29. Ooijen, H.P.G.V. and Bertrand, J.W.M. "Economic due-date setting in job-shops based on routing and workload dependent flow time distribution functions", *Int. J. Production Economics*, **74**, pp. 261-268 (2001).

30. Song, D.P. et al. "Product due date assignment for complex assemblies", *Int. J. Production Economics*, **76**, pp. 243-256 (2002).

31. Sabuncuoglu, I. and Comlekci, A. "Operation-based flowtime estimation in a dynamic job shop", *Omega*, **30**, pp. 423-442 (2002).

32. Wein, L.M. "Due-date setting and priority sequencing in a multiclass M/G/1 queue", *Management Science*, **37**(7), pp. 834-850 (1991).

33. Duenyas, I. "Single facility due date setting with multiple customer classes", *Management Science*, **41**, pp. 608-619 (1995).

34. Slotnick, S.A. and Sobel, M.J. "Manufacturing lead-time rules: Customer retention versus tardiness costs", *European Journal of Operational Research*, **163**, pp. 825-856 (2005).

35. Krarup, J. and Bilde, O. "Plant location, set covering and economic lot size: an o(mn) algorithm for structured problems", *International Series of Numerical Mathematics*, **36**, pp. 155-180 (1977).

36. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company (1989).

37. Hertz, A. and Widmer, M. "Guidelines for the use of meta-heuristics in combinatorial optimization", *European Journal of Operational Research*, **151**, pp. 247-252 (2003).

38. Jans, R. and Degraeve, Z. "Meta-heuristics for dynamic lot sizing: A review and comparison of solution approaches", *European Journal of Operational Research*, **177**, pp. 1855-1875 (2007).

39. Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag (1992).