JJMIE

Volume 3, Number 1, March. 2009
ISSN 1995-6665
Pages 9 - 16

Jordan Journal of Mechanical and Industrial Engineering

Fuzzy Genetic Approach to Economic Lot – Size Scheduling Problem

V.Durga Prasada Rao [a,*], K.Venkata Subbaiah [b], V.Ramachandra Raju [c]

[a] *Dept. of Mech. Engg., S.R.K.R.Engg., College, Bhimavaram–534204, India*

[b] *Dept. of Mechanical Engineering, College of Engineering, Andhra University, Visakhapatnam – 530003, India.*

[c] *J.N.T.U. College of Engineering, Vizianagaram – 535002, India.*

Abstract

The aim of this paper is to consider formulation of Economic Lot size Scheduling Problem (ELSP) in fuzzy environment with fuzzy inventory costs and objective goal. A Genetic Algorithm (GA) is used to solve the problem in the sense that it is computationally simple, yet powerful in its search for improvement. This approach is defined as Fuzzy Genetic Approach (FGA). The ELSP is a problem of scheduling the production of several different items over the same facility on a repetitive manner. The facility is such that only one item can be produced at a time. The ELSP formulation in-turn is considered under the Basic Period (BP) approach with the cycle time, T_i, of each item modified and expressed as a real multiple k_i of a fundamental cycle T. As the typical inventory analysis in the real world situations is sensitive to reasonable errors in the measurement of relevant inventory costs, the inventory costs are assumed to be vague and imprecise in this paper. The objective of minimizing the total inventory cost is also imprecise in nature. The impreciseness in these variables has been represented by fuzzy linear membership functions. The bench mark problem of Bomberger's ELSP has been worked out to highlight the method, and the results are compared with those of corresponding crisp model results. The results indicate that the FGA gives good results and works better even for higher utilization levels of the ELSP.

Keywords: Inventory; Economic lot size scheduling; fuzzy variable; membership function; Genetic Algorithm

1. Introduction

The Economic Lot Size Scheduling Problem (ELSP) is practically important; and is a problem of scheduling production of multiple items over the same facility or machine on a repetitive basis. Examples of ELSP situations include producing several different colors of paint on the same equipment and producing several types of stamped metal parts with the same stamping press. Since items must all be made on the same facility, production of an item will be in lots or batches. The production cycle time T_i of the ith item, therefore, is the elapsed time between productions of consecutive lots of the same item. The facility is such that only one item can be produced at a time, there is a setup cost and setup time associated with each item. The demand rate for each item is known and constant, no shortages are allowed, and for each item the total variable cost is the sum of its setup cost and a time and quantity dependent inventory holding cost.

As the ELSP is NP hard, due to the difficulty of checking the feasibility of schedule, some researchers developed approaches in which additional constraints that guarantee feasibility are added to the problem. Such approaches include the common cycle (CC) approach attributed to Hanssmann [1], and the BP approach attributed to Bomberger [2]. Jones and Inman [3] have provided a detailed analysis of conditions under which the CC approach provides optimal or near – optimal solutions. Doll and Whybark [4] used heuristic approaches which have been effective in approaching the optimal solution to the benchmark problem of Bomberger than analytical approaches. However, they lack a systematic way to test for feasibility and efficient procedures to escape from infeasibility.

The Basic period (BP) approach guarantees feasibility by making the cycle time of each product an integer multiple of a basic cycle time known as the fundamental cycle. Thus the BP approach results in a problem that has one continuous decision variable (the fundamental cycle) and a number of integer decision variables (the integer multipliers) equal to the number of products. Bomberger [2] formulated the problem as a Dynamic Programming (DP) problem, and solved the ELSP under the BP approach.

An excellent review of the literature on the ELSP up to 1976 is given by Elmaghraby [5]. Since then, many new approaches to the problem have been proposed. Khouja et

* Corresponding author : vdp9@rediffmail.com

al. [6] investigated the use of GAs for solving the ELSP. The problem is formulated using the BP approach and solved by using a binary coded GA. The GA is tested on Bomberger's classical problem. Ben-Daya and Hariga [7] modeled the effect of imperfect production processes on the ELSP by taking into account the imperfect quality and process restoration. Torabi et al. [8] addressed the common cycle multi-product lot-scheduling problem in deterministic flexible job shops where the planning horizon is finite and fixed by management. To solve the problem, a mixed integer nonlinear program is developed which simultaneously determines machine allocation, sequencing, lot-sizing, and scheduling decisions.

Yao and Huang [9] solved the ELSP with deteriorating items using the extended basic period approach under Power-of-Two (PoT) policy. Teunter et al. [10] studied the ELSP with two sources of production: manufacturing of new items and remanufacturing of returned items. For both cases, a mixed integer programming formulation is presented for a fixed cycle time, and simple heuristics are proposed. Jenabi et al. [11] addressed ELSP in flexible flow lines with unrelated parallel machines over a finite planning horizon. A mixed zero-one nonlinear mathematical programming method has been developed for solving the problem.

Chatfield [12] developed a Genetic Lot Scheduling (GLS) procedure which combines an extended solution structure with a new item scheduling approach, allowing a greater number of potential schedules to be considered. It maintains solution feasibility determination by employing simple but effective sequencing rules that create nested schedules. He created a binary representation of formulation and utilized a GA to search for low cost ELSP solutions. Yao et al. [13] solved the economic lot scheduling problem with fuzzy demands in order to cope with the uncertainty in demand of items. Chang et al. [14] presented fuzzy extension of ELSP for fuzzy demands via the extended basic period approach and power-of-two policy.

In this paper the ELSP, under the BP approach with cycle time T_i of each item modified and expressed as a real multiple k_i of a fundamental cycle T, is formulated in both crisp and fuzzy environments. The fuzzy concept is considered for ordering costs, carrying costs, and the limitation on total cost. The impreciseness in these variables is represented by linear membership functions. A real coded GA (RGA) is used in search for the near – optimal solution to both the crisp and fuzzy ELSPs. The methodology is illustrated with a bench mark ELSP of Bomberger [2], and the fuzzy results (by FGA) are compared with those obtained by crisp analysis.

2. Problem Formulation

The following assumptions apply to the ELSP under consideration:

1. Only one item can be produced at a time on the machine.
2. Production rates are deterministic and constant.
3. Product demand rates are deterministic and constant.
4. No shortages are allowed.
5. Product setup costs and times are independent of production sequence.
6. Inventory costs are directly proportional to inventory levels.
7. The total variable cost consists of the sum of setup costs and inventory holding costs of all products.

The notations followed are given by:
i = a product or item index.
n = Number of products or items.
Z_i = Annual demand for item-i (units / unit time)
P_i = Annual production rate for item-i (units / unit time)
c_{ci} = Holding cost for item-i-per unit per unit time
c_{ri} = Setup cost of item-i per setup.
t_i = Setup time per setup for item-i.
T_i = cycle time of ith item.

2.1. Crisp Formulation

Under the BP approach, the cycle time T_i for every item is an integer multiple k_i of a fundamental cycle T. Thus, the cycle time T_i for item – i is $T_i = k_iT$. In this paper, each T_i is considered as a real multiple k_i of fundamental cycle time T. Then the crisp formulation of ELSP is given by:
Minimize: Total cost,

$$fbp = \sum_{i=1}^{n}\left[\frac{c_{ri}}{k_i T} + \frac{Z_i T k_i C_{ci}}{2}\left(1 - \frac{Z_i}{P_i}\right)\right] \quad (1)$$

$$\text{Subject to : } \sum_{i=1}^{n}\left[t_i + \frac{Z_i T k_i}{P_i}\right] \le T \quad (2)$$

And $k_i > 0$; $T > 0$

The ELSP thus becomes a problem with k_i and T as decision variables. The constraint (eq.2) ensures that the fundamental cycle time is long enough to accommodate the production of all items even though not every item has to be produced during every fundamental cycle. This formulation is ideally suited for using real coded GA as all k_is are expressed as real variables. The bounds of each k_i are set by using the procedure outlined below:
Step-1: Determine T_i^* for each product by independent solution (IS) method. T_i^* is obtained by substituting Ti=kiT in eq.(1), and then differentiating and equating it to zero.

$$T_i^* = \sqrt{\frac{2 c_{ri}}{Z_i c_{ci}\left(1 - \frac{Z_i}{P_i}\right)}}$$

Step-2: Select the smallest T_i^* as the initial estimate of the fundamental cycle time.
i.e., $T = \text{Min.}[T_i^*]$
Step-3: Determine the possible integer bounds of each k_i defined by:

$$k_i^{(l)} \le \frac{T_i^*}{T} \le k_i^{(u)}$$

Where $k_i^{(l)}$, $k_i^{(u)}$ are lower and upper bounds of k_i.

2.2. Linear Membership Function

A membership function $\mu_{Ai}(x)$, assumed to be linearly increasing over the tolerance interval p_i can be expressed according to Zimmermann [15] as:

$$\mu_{Ai}(x)=\begin{cases}1 & if\ x<d_i\\ 1-\dfrac{x-d_i}{p_i} & if\ d_i\le x\le(d_i+p_i)\\ 0 & if\ x>(d_i+p_i)\end{cases} \quad (3)$$

where d_i and $(d_i + p_i)$ are the tolerance limits for x.
Introducing a new variable, α, which corresponds essentially to $\mu_{Ai}(x)$, the corresponding fuzzy variable ′x′ at the defined aspiration level ′α′ is given by:

$$\mu_{Ai}^{-1}(\alpha)=d_i+(1-\alpha)p_i \quad (4)$$

Similarly a membership function $\mu_{Bj}(x)$, assumed to be linearly decreasing over the tolerance interval p_j can be expressed as:

$$\mu_{Bj}(x)=\begin{cases}1 & if\ x>d_j\\ 1-\dfrac{d_j-x}{p_j} & if\ (d_j-p_j)\le x\le d_j\\ 0 & if\ x<(d_j-p_j)\end{cases} \quad (5)$$

Hence, $\mu_{Bj}^{-1}(\alpha)=d_j-(1-\alpha)p_j$

2.3. Fuzzy Formulation

The fuzzy set concepts are adopted for ordering costs, holding costs, and limitation on total cost. The impreciseness in these variables has been expressed by linear membership functions. Considering the nature of the variables, the membership functions are assumed to be non-decreasing for fuzzy inventory costs, and non-increasing for fuzzy total cost. On applying fuzzy non-linear programming approach to the crisp model, the formulation is:
Maximize: α
Subject to:

$$\sum_{i=1}^{n}\left[\frac{\mu_{cri}^{-1}(\alpha)}{k_iT}+\frac{Z_iTk_i}{2}\mu_{cci}^{-1}(\alpha)\left(1-\frac{Z_i}{P_i}\right)\right]-\mu_{fbp}^{-1}(\alpha)\le 0$$

i.e.,

$$\sum_{i=1}^{n}\left[\frac{(c_{ri}-(1-\alpha)\,p_{cri})}{k_iT}+\frac{Z_iTk_i}{2}(c_{ci}-(1-\alpha)\,p_{cci})\left(1-\frac{Z_i}{P_i}\right)\right]-(fbp+(1-\alpha)\,p_{fbp})\le 0 \quad (6)$$

and $$\sum_{i=1}^{n}\left[t_i+\frac{Z_iTk_i}{P_i}\right]\le T \quad (7)$$

$0 \le \alpha \le 1$

3. Genetic Algorithm

A GA performs a multi directional search by maintaining a population of potential solutions and encourages information formation and exchange between these directions. The population undergoes a simulated evolution: at each generation the relatively "good" solutions reproduce, while the relatively "bad" solutions die. To distinguish between different solutions, we use an objective (evaluation) function which plays the role of an environment.

The initial population of solutions is created by random selection of a set of chromosomes (solutions). Once a chromosome is created, it is necessary to evaluate the solution, particularly in the context of the underlying objective and constraint functions. The evaluation of a solution means calculating the objective function value and constraint violations. After assigning a relative merit to the solutions (called the fitness), the population of solutions is modified to create hopefully a better population. In this process the three main operators, viz., reproduction, crossover, and mutation are used. This completes the generation of the GA. Then a new population of solutions is created and the above procedure is repeated until the required conditions are satisfied.

The binary representation of decision variables used in genetic algorithms has some drawbacks when applied to multi-dimensional, high precision numerical problems. Real coded or floating – point representation, on the other hand, has a rising usage because of the empirical findings that real codings have worked well in a number of practical problems. The procedure of proposed GA is shown in figure 1. The Components of the developed system is discussed in the following sections.

3.1. Representation and Initialization of Population of Solutions

As a real parameter GA is used, the variables are represented by floating point numbers over whatever range is deemed appropriate. That is, the process of finding an optimal solution to a problem starts by defining a chromosome (solution) as an array of variable values to be optimized. If the chromosome has n variables (an n-dimensional optimization problem) given by $k_1, k_2, \ldots .k_{10}$, T,α, a chromosome is written as an array with $1 \times n$ elements so that

Chromosome = $[k_1, k_2, \ldots .k_{10}, T, \alpha]$ (8)

```
Procedure Genetic Algorithm
Begin
        t ← 0
        initialize population (t)
        evaluate population (t)
        while (not terminate – condition) do
        begin
                t ← t + 1
                reproduction
                crossover
                mutation
                evaluate population (t + 1)
        end
end
```

Figure 1: Procedure of GA

All variables are normalized to have values between 0 and 1, the range of a uniform random number generator. Then the values of a variable are "un-normalized" in the fitness function. If the range of values of an i^{th} variable is between k_i^l and k_i^u, then the un-normalized value is given by:

$$k_i = \left(k_i^u - k_i^l\right) k_i^{norm} + k_i^l \quad (9)$$

where k_i^{norm} = Normalized value of variable, $0 \le k_i^{norm} \le 1$

Now to begin the GA, a population of s_p chromosomes is defined by a matrix with each row in the matrix being 1 × 12 array of continuous values.

3.2. Evaluation of Solutions

Once a chromosome (or a solution) is created, it is necessary to evaluate the solution, particularly in the context of the underlying objective and constraint functions. The evaluation function is called the fitness function. In most of the constrained optimization problems, the fitness function is obtained by adding a penalty proportional to the constraints' violations to the objective function value. The penalty term with respect to a constraint violation is nothing but a user defined penalty parameter multiplied by some function of the constraint. However, this method has two difficulties – fixing a penalty parameter, convergence of fitness function to an artificial local optimum. A modified version of this method, which does not need any penalty parameter, is followed in the present paper for the evaluation of fitness function. The method is based on feasible over infeasible solutions.

The method of feasible over infeasible solutions [16] employs a tournament selection operator in which two solutions are compared at a time, and the following scenarios are always assured:

1. Any feasible solution is preferred to any infeasible solution.
2. Among two feasible solutions, the one having smaller constraint violation is preferred.
3. Among two infeasible solutions, the one having a smaller constraint violation is preferred.

Motivated by these arguments, the following fitness function (F_i) is used for any ith solution of the population.

$$F_i = \begin{cases} fbp_i, & \text{if solution is feasible} \\ fbp_{max} + \sum_{j=1}^{2} \langle g_j(X) \rangle, & \text{otherwise} \end{cases} \quad (10)$$

where fbp_i is the objective function value of ith solution. The parameter fbp_{max} is the objective function value of the worst feasible solution in the population. The g_j is jth constraint of greater than or equal to type, and the bracket operator < > denotes the absolute value of the operand, if the operand is negative. Otherwise, if the operand is non negative, it returns a value of zero. It is to be noted that the objective function value is not computed for any infeasible solution. Since all feasible solutions have zero constraint violation and all infeasible solutions are evaluated according to their constraint violations only, both the objective function value and constraint violation are not combined in any solution in the population. Thus, there is no need to have any penalty parameter for this approach.

After all, the solutions in the population are evaluated in terms of their fitness value. They are ranked in the order of best to worst solutions, and the termination condition is checked. The termination criterion is maximum number of generations to be used. If the termination criterion is not satisfied, then the three genetic operators are applied to improve the population of solutions.

3.3. Genetic Operators

The three genetic operators to be applied to improve the population of solutions are selection, crossover, and mutation operators.

3.3.1. Selection Operator

The primary objective of the selection or reproduction operator is to make duplicates of good solutions and eliminate bad solutions in a population, while keeping the population size constant. A tournament selection scheme is used where two solutions at a time are compared, and the best in terms of objective function value is selected [17]. That is, the scheme works in such a way that it picks randomly 2 individual solutions from the population and copies the best individual (in terms of fitness value) into the intermediate population; called the mating pool. This process is repeated population number of times. The mating pool, being comprised of tournament winners, has a higher average fitness than the average population fitness. This fitness difference provides the selection pressure, which drives the GA to improve the fitness of each succeeding generation.

3.3.2. Crossover Operator

Crossover using SBX operator is performed [18]. The procedure of computing offspring solutions $y^{(1)}$and $y^{(2)}$ from two parent solutions $x^{(1)}$, $x^{(2)}$ are as follows:

1. Create a random number between 0 and 1.
2. Find a parameter $\bar{\beta}$ using a polynomial probability distribution as given by:

$$\bar{\beta} = \begin{cases} (\alpha u)^{\frac{1}{p+1}}, & \text{if } u \le 1/\alpha \\ \left(\frac{1}{2-\alpha u}\right)^{\frac{1}{p+1}}, & \text{otherwise} \end{cases} \quad (11)$$

where p = Distribution index for SBX (non-negative).
$\alpha = 2 - \beta^{-(p+1)}$

$$\beta = 1 + \frac{2}{x^{(2)} - x^{(1)}} \min\left[\left(x^{(1)} - x^{l}\right), \left(x^{u} - x^{(2)}\right)\right]$$

and x^l, x^u = lower & upper bounds of variable.

It is assumed that $x^{(1)} < x^{(2)}$. This procedure allows a zero probability of creating any offspring solution outside the prescribed range [x^l, x^u]. A small value of p allows solutions for away from parents to be created as offspring solutions and a large value restricts only near– parent solutions to be created, as offspring solutions. The SBX operator is applied variable by variable and in all simulation results, p = 2 has been used.

3. The offspring solutions are then calculated as follows:

$$y^{(1)} = 0.5[(x^{(1)} + x^{(2)}) - \bar{\beta} \mid x^{(2)} - x^{(1)}|], \quad (12)$$

$$y^{(2)} = 0.5[(x^{(1)} + x^{(2)}) + \bar{\beta} \mid x^{(2)} - x^{(1)}|] \quad (13)$$

3.3.3. Mutation Operator

A polynomial probability distribution is used to create a solution y in the vicinity of parent solution x [19]. The following procedure is used where lower and upper bounds (x^l and x^u) are specified.

1. Create a random number u between 0 and 1.
2. Calculate the parameter $\bar{\delta}$ as follows:

$$\bar{\delta} = \begin{cases} \left[2u + (1-2u)(1-\delta)^{q+1}\right]^{\frac{1}{q+1}} - 1, & \text{if } u \le 0.5, \\ 1 - \left[2(1-u) + 2(u-0.5)(1-\delta)^{q+1}\right]^{\frac{1}{q+1}}, & \text{otherwise} \end{cases} \quad (14)$$

Where q = Distribution index for mutation (non negative).
δ = min $[(x - x^l), (x^u - x)]/ (x^u - x^l)$
This ensures that no solution would be created out side the range $[(x^l, x^u)$.

3. Calculate the mutated child as follows:

$$y = x + \bar{\delta}\, \delta_{max}. \quad (15)$$

Where δ_{max} = maximum perturbance allowed in the parent solution = $x^u - x^l$.
In all simulation results, q = 100 has been used.

3.4. GA Parameters

Selecting GA parameters like population size (s_p), crossover probability (p_c), mutation probability (p_m), and number of generations (n_g) is very difficult due to many possible variations in the algorithm and objective function. A real parameter GA relies on random number generators for creating the population, crossover and mutation. A different random number seed produces different results.

As far as population size is considered, traditionally large numbers of population of solutions have been used to thoroughly explore complicated objective surfaces. The number of generations is something like termination criteria, which indicates how many times the trials (iterations) are to be made.

Crossover probability is used to find the probable number of solutions (s_p, p_c) to be crossed over to produce an equal number of offspring solutions. In order to preserve some good solutions selected during reproduction operator, ((1-p_c). s_p) number of solutions are simply copied to the new population. This process helps in exploiting promising regions of objective space by combining information from promising solutions.

Mutation probability is used in finding the probable number of variables to be mutated (p_m x s_p×n). This process helps in exploring different areas of the objective space by randomly introducing changes, or mutations, in some of the variables.

Table 1: Data of Bomberger's metal stamping problem

Part No.,i	Demand, Z_i (Units/year)	Production rate, P_i (units/day)	Setup cost, c_{ri} ($/setup)	Setup time, t_i (hours)	Holding cost, c_{ci} ($/unityear)
1	24,000	30,000	15	1	0.00065
2	24,000	8,000	20	1	0.01775
3	48,000	9,500	30	2	0.01275
4	96,000	7,500	10	1	0.01000
5	4,800	2,000	110	4	0.27850
6	4,800	6,000	50	2	0.02675
7	1,440	2,400	310	8	0.15000
8	20,400	1,300	130	4	0.59000
9	20,400	2,000	200	6	0.09000
10	24,000	15,000	5	1	0.00400

The production was on a one shift (8 hour day) basis. Daily costs were based on a 240 day year.

4. Computational Results

To illustrate the proposed methodology, the practical example originally given by Bomberger [2] is considered. The basic data for this problem were developed from metal stamping data. Table 1 shows the Bomberger's problem with machine utilization (Σ Z_i / P_i) equal to 0.22 (22%). It is to be noted that in the real world, the decision makers should be faced with the uncertainty and fluctuation problems in estimating the inventory costs and in setting limit on total cost of inventory. Therefore, a decision maker may employ the concept of fuzzy demands with the membership functions to cope with variations in inventory costs and total cost before planning the production strategies (i.e., solving the ELSP). Besides 22% machine utilization, the levels of utilization of 44%, 50%, 60%, 66%, 70%, 80%, 88%, 92%, and 95% are also solved by the present formulations. The different machine utilizations are obtained in such a way that Z_i = a times Z_i given in table 1, where a = possible positive constant (2, 2.266, 2.719, 3, 3.173, 3.626, 4, 4.1703, 4.306). In all the problems with respect to different machine utilizations, the following maximum acceptable violations of c_{ri} and c_{ci} are assumed:

p_{cri} = 5, 5, 7.5, 2.5, 25, 12.5, 100, 25, 50, 1
p_{cci} = 0.0002, 0.004, 0.003, 0.0025, 0.07, 0.006, 0.03, 0.15, 0.02, 0.001.

Also, the values of fbp and their corresponding maximum acceptable violations with respect to different machine utilizations are assumed to be as given in table 3. Table 2 shows the comparison of results by the proposed fuzzy and crisp models of ELSP for different cases of Bomberger's problem with different machine utilizations. The total cost of the crisp solution increases as the utilization increases. The total cost of the corresponding fuzzy model solution also increases with respect to increase in utilization. It is to be noted that, the difference between the total cost of crisp and fuzzy models is as low as $14 for 88% utilization problem. This is owing to the fuzzy costs that the constraints could actually be more binding. The difference between total cost of crisp and fuzzy models is as high as $55.42 for 60% utilization problem.

Table 2: Comparison of crisp and fuzzy ELSP results

%Utili-zation	fbp ($/year)		T(days)		k_1		k_2		k_3		k_4		k_5		k_6		k_7		k_8		k_9		k_{10}		α of fuzzy models
	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	Cri	Fuz	
22.06	4073.56	4025.43	47.29	43.16	7.04	8.14	1.56	1.74	1.60	1.74	0.75	0.83	2.06	2.23	4.48	5.01	8.60	9.41	0.77	0.84	2.42	2.66	1.64	1.81	0.9491
44.12	5633.65	5612.80	33.08	35.75	7.13	7.29	1.59	1.48	1.64	1.51	0.78	0.74	2.10	1.94	4.54	4.22	8.71	7.95	0.80	0.74	2.50	2.32	1.66	1.58	0.9839
50.00	5983.67	5935.04	32.21	32.89	7.00	9.99	1.53	1.50	1.58	1.82	0.76	0.91	2.03	2.12	4.38	5.16	8.40	9.58	0.78	0.71	2.43	2.38	1.06	1.55	0.9610
60.00	6493.50	6438.08	32.41	36.43	7.00	7.97	1.39	1.83	1.44	1.39	0.70	0.71	1.84	1.63	3.97	4.31	7.63	7.43	0.72	0.65	2.22	1.97	1.00	2.27	0.9619
66.18	6740.27	6698.85	34.47	36.48	7.00	8.01	1.24	1.31	1.29	1.40	0.63	0.66	1.65	1.58	3.54	4.49	7.00	7.21	0.65	0.59	2.00	1.72	1.30	2.16	0.9511
70.00	6951.11	6902.56	33.53	35.05	7.00	7.59	1.24	1.20	1.28	1.24	0.62	0.60	1.64	1.59	3.51	3.53	7.00	7.06	0.66	0.62	1.99	1.95	1.00	1.35	0.9974
80.00	7378.37	7336.75	35.79	39.10	7.00	8.86	1.04	1.46	1.08	1.14	0.51	0.48	1.43	1.25	3.00	3.21	7.00	7.13	0.58	0.49	1.72	1.49	1.00	1.89	0.9632
88.24	7628.41	7614.41	38.46	38.98	7.00	7.06	1.00	1.00	1.00	1.03	0.44	0.44	1.27	1.24	3.00	3.17	7.00	7.01	0.52	0.50	1.50	1.46	1.00	1.03	0.9855
92.00	7774.18	7733.20	39.49	43.37	7.00	7.49	1.00	1.45	1.00	1.13	0.40	0.27	1.20	1.27	3.00	3.21	7.00	7.17	0.50	0.48	1.39	1.21	1.00	1.46	0.8889
95.00	7891.07	7839.65	40.56	41.05	7.00	7.06	1.00	1.01	1.00	1.00	0.37	0.36	1.14	1.16	3.00	3.00	7.00	7.00	0.48	0.46	1.33	1.32	1.00	1.01	0.9669

Cri: Crisp result; Fuz: Fuzzy result by FGA

Table 3: Values of fbp & p_{fbp} for different machine utilizations

$\Sigma Z_i / P_i$	22%	44%	50%	60%	66%	70%	80%	88%	92%	95%
fbp($)	4000	5600	5900	6400	6650	6900	7300	7600	7600	7800
p_{fbp}($)	500	800	900	1000	1000	1000	1000	1000	1200	1200

Table 4: GA Parameters used in crisp and Fuzzy models

% utilization	Crossover probability (p_c)	Mutation probability (p_m)	Population size (s_p)	Number of generations (n_g)	Best generation and Run No.
22.06	0.92	0.045	100	150	148,16
50.00	0.955	0.025	20	250	231,8
60.00	0.98	0.045	20	250	249,7
66.18	0.96	0.05	20	250	239,27
75.00	0.945	0.065	20	250	246,22
80.00	0.96	0.075	20	220	220,5
88.24	0.905	0.05	110	250	244,20
92.00	0.93	0.05	20	250	245,12
95.00	0.89	0.05	110	240	233,24

Table 5 : Fuzzy ELSP solution costs at increasing utilization levels and comparison with other approaches.

	88%	92%	95%
Independent solution: IS	$7589	$7714	$7812
Khouja et.al [6]: BGA	$8782	$9746	$12018
Chatfield [12]: GLS	$7697	$7974	$9140
Crisp solution: RGA	$7628	$7774	$7891
Fuzzy Genetic Approach: FGA	$7614	$7733	$7839
FGA distance from IS: (FGA–IS) / IS	0.3294%	0.7907%	0.3456%

Table 5 shows the comparison of ELSP solution costs of Bomberger's stamping problem at higher utilization levels (88%, 92%, and 95%) with those of corresponding fuzzy models. For all these levels of utilization, the total costs by FGA on the average are 0.4885% above that of independent solution (IS). It definitely indicates that the perturbations of inventory costs and total cost should be taken into account for the ELSPs in making the decision more proper and optimal. The present formulation reveals its usefulness.

The real parameter GA was coded in C-language and a parametric study of GA is carried out in solving each of the crisp models. The study is carried out by varying different GA parameters, viz., crossover probability (p_c), mutation probability (p_m), population size (s_p), and number of generations (n_g). By this study, the best set of GA parameters which gives the minimum value of the objective function can be found. The same sets of GA parameters are used in solving the corresponding fuzzy models.

Now for the crisp problem with the % utilization of 88.24%, initially one parameter, viz., crossover probability is varied from 0.89 to 0.99, Keeping the other parameters fixed to the values of p_m = 0.01, s_p = 20 , and n_g =20. Then all the objective function values are compared, and the crossover probability corresponding to minimum value of objective function value is selected as the best p_c. It was found to be 0.905. The above study is then repeated for different values of mutation probability from 0.005 to 0.11 in steps of 0.005, keeping the other parameters fixed to the values of p_c = 0.905, s_p = 20, n_g = 20. Then the best mutation probability corresponding to minimum value of objective function was found to be 0.05. By keeping the parameters p_c = 0.905, p_m =0.05, ng =20, the population size is varied from 20 to 220 in steps of 10. The best population size was found to be 110. Finally the number of generations are varied from 20 to 250 in steps of 10, keeping the other parameters constant at p_c = 0.905, p_m = 0.05, s_p = 110. The best number of generations was found to be 250. Thus the best GA parameters after the study are: p_c =0.905, p_m = 0.05, s_p =110, ng =250. The results of parametric study, i.e., best GA parameters for different % utilizations are furnished in table 4.

5. Conclusions

This paper has addressed a useful formulation of ELSP under fuzzy environment. The ELSP in-turn was formulated via the BP approach with the cycle time T_i of each item modified and expressed as a real multiple k_i of a fundamental cycle T. This modified formulation along with the fuzzification of inventory costs and total cost is ideally suited for using real coded GA. It is noticed that the FGA produced good results as compared to the corresponding crisp results of all cases of the Bomberger's stamping problem. The total cost of crisp and fuzzy models is as low as $14 for 88% utilization problem and as high as $55.42 for 60% utilization problem (Table-2). Also FGA has given better results at higher utilization levels of the problem (88%, 92%, and 95%). For all these levels of utilization, the total costs by FGA on the average are 0.4885% above that of independent solution (Table-5). It showed that the perturbation of inventory costs and total

cost should be taken into account in the ELSP by the use of proposed model. Future research may consider the fuzzification of other parameters, viz., demand and fundamental cycle time. The GA may also explore the solutions for other approaches of ELSP. Also in this paper, linear membership functions are only considered to represent the nature of variations of fuzzy variables. The membership functions like parabolic, exponential, hyperbolic, etc. can also be considered.

References

[1] Hanssmann F. Operation research in production and inventory. New York: Wiley; 1962.

[2] E.E. Bomberger, "A dynamic programming approach to a lot size scheduling problem". Management Science, Vol. 12, 1966, 778-784.

[3] P.C. Jones, R.R. Inman, "When is the economic lot scheduling problem easy". IIE Transactions, Vol. 21, 2004, 11-20.

[4] L. Doll, C. Whybark, "An iterative procedure for the single-machine, multi-product lot scheduling problem". Management Science, Vol. 20, 1973, 50-55.

[5] S.E. Elmaghraby, "The economic lot scheduling problem (ELSP): review and extensions". Management Science, Vol. 24, 1978, 587-598.

[6] M. Khouja, Z. Michalewicz, M. Wilmost, "The use of genetic algorithms to solve the economic lot scheduling problem". European Journal of Operational Research, Vol. 110, 1998, 509-524.

[7] M. Ben-Daya, M. Hariga, "Economic lot scheduling problem with imperfect production processes". Journal of the Operational Research Society, Vol. 51, 2000, 875-881.

[8] S.A. Torabi, B. Karimi, S.M.T. Ghomi, "The common cycle economic lot scheduling in flexible job shops: the finite horizon case". International Journal of Production Economics, Vol. 97, 2005, 52-65.

[9] M.J. Yao, J.X. Huang, "Solving the economic lot scheduling problem with deteriorating items using genetic algorithms". Journal of Food Engineering, Vol. 70, 2005, 309-322.

[10] R. Teunter, O. Tang, K. Kaparis, "Heuristics for the economic lot scheduling problem with returns". 14th International Symposium on Inventories, Budapest, Hungary, 2006.

[11] M. Jenabi, S.M.T. Ghomi, S.A. Torabi, B. Karimi, "Two hybrid meta-heuristics for the finite horizon ELSP in flexible flow lines with unrelated parallel machines". Applied Mathematics and Computation, Vol. 186, 2007, 230-245.

[12] D.C. Chatfield, "The economic lot scheduling problem: a pure genetic search approach". Computers & Operations Research, Vol. 34, 2007, 2865-2881.

[13] M.J. Yao, P.T. Chang, S.F. Huang, "On the economic lot scheduling problem with fuzzy demands". International Journal of Operations Research, Vol. 2, 2005, 58-71.

[14] P.T. Chang, M.J. Yao, S.F. Huang, C.T. Chen, "A genetic algorithm for solving fuzzy economic lot-size scheduling problem". International Journal of Production Economics, Vol. 102, 2006, 265-288.

[15] Zimmermann H J. Fuzzy set theory and its applications. Boston: Kluwer Academic Publishers; 1991.

[16] K. Deb, "An efficient constraint handling method for genetic algorithms". Computer Methods in Applied Mechanics and Engineering, Vol. 186, 2000, 311-338.

[17] D.E. Goldberg, K. Deb, "A comparison of selection schemes used in genetic algorithms". Foundations of Genetic Algorithms, I, 1991, 69-93.

[18] K. Deb, R.B. Agrawal, "Simulated binary crossover for continuous search space". Complex Systems, Vol. 9, 1995, 115-148.

[19] K. Deb, M. Goyal, "A combined genetic adaptive search (Gene AS) for engineering design". Computer Science and Informatics, Vol. 26, 1996, 30-45.