International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:2, No:11, 2008

The Economic Lot Scheduling Problem in Flow Lines with Sequence-Dependent Setups

M. Heydari, and S. A. Torabi

***Abstract*—**The problem of lot sizing, sequencing and scheduling multiple products in flow line production systems has been studied by several authors. Almost all of the researches in this area assumed that setup times and costs are sequence –independent even though sequence dependent setups are common in practice. In this paper we present a new mixed integer non linear program (MINLP) and a heuristic method to solve the problem in sequence dependent case. Furthermore, a genetic algorithm has been developed which applies this constructive heuristic to generate initial population. These two proposed solution methods are compared on randomly generated problems. Computational results show a clear superiority of our proposed GA for majority of the test problems.

***Keywords*—**Economic lot scheduling problem, finite horizon, genetic algorithm, mixed zero-one nonlinear programming, sequence-dependent.

I. Introduction

SMOOTH and cost-efficient running of a production line often depends on selecting appropriate lot-sizes and production schedules. One of the most difficult lot-sizing problems which is known to be NP-hard is ELSP [6]. We encounter with this problem when one machine is used to meet deterministic and stationary demand of several products over an infinite horizon [11]. In addition to the discrete parts manufacturing, multi-products or multi-purpose processors are common features in many chemical plants such as those producing pharmaceuticals, cosmetics, polymers, biochemical, food and beverages, etc. Hence any methodology for solving the ELSP (in particular with sequence-dependent setups) has enormous potential of applicability for industry.

The ELSP has been studied extensively from the time that Rogers [15] applied the economic order quantity (EOQ) formula. Following Rogers's work, researchers have used one of following three cyclic policies: The common cycle, basic or fundamental cycle and time-varying approaches. These include the OEM of Torabi et al [19], the golden section search of Ouenniche [13] for the first policy, dynamic approach by Bomber [2], marginal analysis by Fujita [5] for the second policy and lagrangian relaxation method by Dobson [3] for the last policy.

M. Heydari is with Department of Industrial Engineering, National Iranian Oil Refinery & Distribution Company (NIORDC), Tehran, Iran (corresponding author to provide phone: 0098-21-88941474; fax: 0098-21-66155115; e-mail: moh_hei_2004@yahoo.com).

S. A. Torabi is with Department of Industrial Engineering, Tehran University, Tehran, Iran (e-mail: sa.torabi@ut.ac.ir).

The ELSP with sequence-dependent setups (ELSPSD) was investigated by Maxwell [12]. He showed that the problem can be modeled as minimizing a quadratic criterion subject to a set of linear constrains. He presented an approximate solution technique by using implicit rules for specifying of the saw-tooth pattern of inventory and using of heuristic rules for changing the length of the production cycle. Other early papers also include those of Geoffrion and Graves [7] and Driscoll and Emmons [4]. Geoffrion and Graves (1976) used a combined quadratic linear programming approach. The method of Driscoll and Emmons (1977) is based on using dynamic programming recursion to solve the problem with specified due dates. Singh and Foster [17] developed a heuristic with an imbedded quadratic programming problem. Irani and Gunasena [10] employed family structure setups. Dobson (1989) applied time-varying policy to solve the sequence-dependent ELSP. He used a lagrangian relaxation of the formulation which leads to a partial separation of the embedded lot sizing and traveling salesman problem. The relaxation results in a new combinational problem related to the minimum spanning tree problem. The information about frequency of production, obtained from this relaxation, is used to find heuristic solution for the entire problem.

Talor et al [18] developed a heuristic algorithm to find a good solution for the sequence dependent lot scheduling problem. Eliminating the need for creating new artificial problems and implementing feasibility tests, were preferences of their algorithm. Wagner et al [21] developed a heuristic procedure that provides a range of solutions from that a manager can choose one which is useful in an actual stochastic production environment. Their heuristic outperforms Dobson's heuristic (1987) when the utilization is high and the sequence dependent setup times and costs are significant. Honng-choun oh et al [8,9] in two papers decomposed the entire complex problem into two sub-problems; lot scheduling and sequencing. For the lot scheduling, they presented a novel mixed integer non linear programming approach. Then, for sequencing sub-problem, they proposed an efficient tabu search to determine the sequence and a linear programming approximation to construct a schedule. However, to the best of our knowledge, there is no contribution to economic lot scheduling problem in flow lines with sequence dependent setups.

The text is organized as follow: first we state the problem at hand in section II. Then in section III we present a mixed non linear integer programming for both finite and infinite planning horizon cases. For the finite horizon case of the problem, a heuristic search is proposed in section IV. In order

to obtain an optimal or near optimal solution in medium and large scale problems, a GA is also developed in section V. The effectiveness and consistent performance of the GA is demonstrated by some randomly simulated test problems in section VI. Final section is devoted to conclusions.

II. Problem Statement

Assumptions

- Only one machine is available at each stage and the machines are continuously available in each stage.
- Each product requires at most *m* operations at different stages.
- All parameters such as demand and production rates, setup times, setup costs and inventory holding costs are deterministic and constant over planning horizon.
- External demands occur only for end products.
- Production rates are different for different products and different stages or machines.
- Inventory holding costs directly are proportional to the inventory levels and holding times.
- Shortages are not allowed.
- Preemption is not allowed, that is, at a given stage, once the processing of a lot has started, it must be completed without interruption.
- No inter-stage overlapping is allowed; that is, a lot is not transferred to the next stage until the entire lot is processed at the current stage.
- The buffers between stages are unlimited; hence, in-process inventory is allowed.
- Delivery of finished products is continuous.
- Production capacity is sufficient to meet total demand; thus there exists at least one feasible solution.

Parameters

n: number of products.
m: number of machines (stages or work centers).
i, k: component indices.
j: stage index.
d_i : demand rate of product *i*.
p_{ij} : production rate of product *i* at stage *j*.
t_{ij} : processing time for a lot of product *i* at stage *j*.
s_{kij} : setup time to switch from production of product *k* to product *i* at stage *j*.
c_{kij} : setup cost to switch from production of product *k* to product *i* at stage *j*.
h_{ij} : inventory holding cost per unit of product *i* per unit time between stage *j* and *j+1*.
h_i : inventory holding cost per unit of finished product *i* per unit time.
M: a large real number.
H: length of finite planning horizon.

Decision Variables

σ_j : production sequence vector at stage *j*.
T: common production cycle length.
F: the number of production cycles over the finite planning horizon.
Q_i : production lot size of component *i* at different stages ($Q_i = d_i T$).
b_{ij} : process beginning time of component *i* at stage *j* (after setup operation).

$$x_{ilj}\begin{cases}1 & \text{if product i is assigned to l-th position in stage j}\\ 0 & \text{otherwise}\end{cases}$$

Since after processing each component at each stage, there would be a value added for the component, therefore values of h_{ij} will be non-decreasing, that is:

$$h_i > h_{ij}\,; h_{ij} \ge h_{i,j-1}\,; \quad i = 1,...,n\,; \qquad j = 2,...,m-1.$$

The problem can be formulated as a mixed zero-one non linear program. As mentioned before, we restrict our attention to the common cycle context where all products are assigned the same cycle time *T*. The objective of the problem is to minimize sum of setup costs, work-in-process inventory costs and finished products inventory costs per unit of time. The first cost element, the setup cost per time unit is as follows (see Fig. 1):

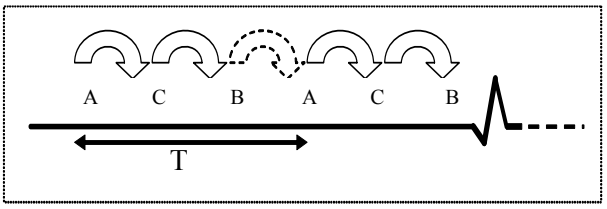


Fig. 1 Frequency setups of machine *j* to produce the A-C-B sequence

The setup costs for the non-initial products in the sequence are equal to:

$$C_{k,i,j} * x_{k,l,j} * x_{i,l+1,j}\,; \qquad i,k=1,2,..,n; i \ne k;\ l=1,2,..,n-1;\ j=1,2,..,m.$$

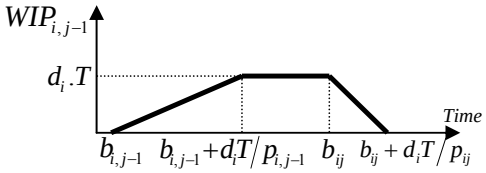
and the setup cost at switch from last product to first one in the sequence is equal to:

$$C_{k,i,j} * x_{k,n,j} * x_{i,1,j}\,; \qquad i,k=1,2,..,n;\ i \ne k\,;\ j=1,2,..,m.$$

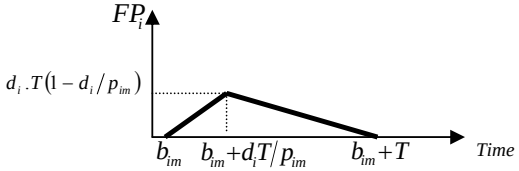
Without lose of generality, we assume that the setup cost of the first position product in the first run is equal to its setup cost in which it is produced after *n-th* position product. Therefore, the total setup costs per unit of time is:

$$TC_{SETUP} = \frac{1}{T}\left[\left(\sum_{j=1}^{m}\sum_{l=1}^{n-1}\sum_{i=1}^{n}\sum_{\substack{k=1\\k\ne i}}^{n} C_{k,i,j} * x_{k,l,j} * x_{i,l+1,j}\right) + \left(\sum_{j=1}^{m}\sum_{i=1}^{n}\sum_{\substack{k=1\\k\ne i}}^{n} C_{k,i,j} * x_{k,n,j} * x_{i,1,j}\right)\right]$$

Moreover, two types of inventory are considered: work-in-process inventory and finished products inventory which are indicated in Fig. 2.



(a)



(b)

Fig. 2 Inventory levels: (a) WIP between stages *j-1* and *j* (b) Inventory of finished product *i*

Final stages and adding them up over all components and all stages, we will have:. (For more details see (Ouenniche, [13])

$$HC=\sum_{i=1}^{n}\left[h_i.\frac{d_i}{2}\left(1-\frac{d_i}{p_{im}}\right)+\frac{d_i^2}{2}\sum_{j=2}^{m}h_{i,j-1}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right].T$$
$$+\sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}.d_i\left(b_{ij}-b_{i,j-1}\right)$$

Therefore, the sum (over all products and all stages) of setup and inventory holding cost per time unit is:

$$\mathrm{TC}=\frac{1}{T}\left[\left(\sum_{j=1}^{m}\sum_{l=1}^{n-1}\sum_{\substack{i=1\\k=i}}^{n}\sum_{k=1}^{n}C_{k,i,j}*x_{k,l,j}*x_{i,l+1,j}\right)+\left(\sum_{j=1}^{m}\sum_{\substack{i=1\\k=i}}^{n}\sum_{k=1}^{n}C_{k,i,j}*x_{k,n,j}*x_{i,1,j}\right)\right]$$

$$\sum_{i=1}^{n}\left[h_i.\frac{d_i}{2}\left(1-\frac{d_i}{p_{im}}\right)+\frac{d_i^2}{2}\sum_{j=2}^{m}h_{i,j-1}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right]T+\sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}d_i\left(b_{ij}-b_{i,j-1}\right)$$

Problem P1, demonstrate the finite horizon version of the problem:

PROBLEM P1

$$\text{Min } Z=\frac{1}{T}\left[\left(\sum_{j=1}^{m}\sum_{l=1}^{n-1}\sum_{\substack{i=1\\k=i}}^{n}\sum_{k=1}^{n}C_{k,i,j}*x_{k,l,j}*x_{i,l+1,j}\right)+\left(\sum_{j=1}^{m}\sum_{\substack{i=1\\k=i}}^{n}\sum_{k=1}^{n}C_{k,i,j}*x_{k,n,j}*x_{i,1,j}\right)\right]+$$

$$(1)\ \sum_{i=1}^{n}\left[h_i.\frac{d_i}{2}\left(1-\frac{d_i}{p_{im}}\right)+\frac{d_i^2}{2}\sum_{j=2}^{m}h_{i,j-1}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right].T+\sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}d_i\left(b_{ij}-b_{i,j-1}\right)$$

Subject to:

$$(2)\ b_{i,j-1}+\frac{d_i.T}{p_{i,j-1}}\le b_{ij};\ i=1,...,n,\ j=2,...,m$$

$$(3)\ b_{kj}+\frac{d_k.T}{p_{kj}}+s_{kij}-b_{ij}\le M\left(2-x_{k\ell j}-x_{i,\ell+1,j}\right);\ i=1,...,n;j=1,...,m;k\ne i;\ \ell<n$$

$$(4)\ \sum_{\ell=1}^{n}x_{i\ell j}=1;\quad j=1,..,m;\ i=1,...,n$$

$$(5)\ \sum_{i=1}^{n}x_{i\ell j}=1;\quad j=1,...,\mathrm{m};\ \ell=1,...,n$$

$$(6)\ b_{ij}\ge s_{kij}.x_{i1j}.x_{knj};\ j=1,...,m;\ i=1,...,n$$

$$(7)\ b_{im}+\frac{d_i.T}{p_{im}}\le T;\ i=1,...,n$$

$$(8)\ F*T=H$$

$$(9)\ T\ge 0,\ b_{ij}\ge 0;\ x_{i\ell j}\in\{(0,1\};\forall i,j,l;i\ne l$$

$$(10)\ F>=1;F\in Z$$

Constraints (2) state that at each stage, no product can be produced before it is completed at the previous stage. Constraints (3) stipulate that at each stage, no product can be processed before completion of its predecessor in the production sequence. Constraints (4) and (5) state that each product has a unique position in the sequence. Constraints (6) state that, at each stage, processing of the first product in the sequence cannot start before setting up the corresponding machine. Constraints (7) assure that the resulting schedule is cyclic so as the process completion time for each component at the final stage is less than or equal to the cyclic time. Constraints (8) implies that the common cycle is such that the planning horizon *H* is an integer multiple of *T*. Finally constraints (9) and (10) are non-negativity constraints.

Another mixed zero-one non linear program developed in this paper corresponds to the problem in infinite planning horizon case. It is worthy to mention that this model and the previous one are similar in both objective and constraints unless constraints (8) that must be eliminated and constraints (7) that should change to the following constraints:

$$(11)\ b_{kj}+\frac{d_{kj}T}{p_{kj}}+s_{kij}-b_{ij}-T\le M\left(2-x_{i,1,j}-x_{k,n,j}\right);i=1,...,n;j=1,...,m;k\ne i;$$

Constraints(11) imply that, at each stage the time that elapses between the production starting time of the first product in the sequence and the completion time of the last product, is less than or equal to the common cycle length *T*.

Unfortunately, it is difficult to solve this problem optimality. Hereafter we propose a heuristic search to solve it approximately.

III. Heuristic Approach

In this section we propose some simple heuristics to determine the production sequence vectors at different stages.

- **Neighborhood Search (NEH)**

First, all products at each stage are arranged once in a descending order and once more in an ascending order of $\partial_{ij} = d_i / p_{ij}$. This criterion represents the fraction of cycle time in which directly the *i-th* product is produced by the *j-th* machine. Therefore, this sequencing rule can be seen as generalization of the shortest and longest processing time for descending and ascending order respectively. Then the first two products will be put in the sequence directly. For remainder *i* products, *i=3,4,..,n*, the best order will be found according to the setup costs or setup times, by placing it in all the possible *i* positions in the sequence of products that are already ordered. For example, if *i=4*, the previously built sequence would contain the first three products of the sorted list calculated earlier. Then the fourth product could be placed either in the first, in the second, in the third or in the last position. The best sequence of the four products would be selected for the next step. This procedure can make many similar sequences based on this NEH heuristic and in the following modification. After having ordered the products, we pick randomly two products from the ordered list and exchange them with the two first products. Then we proceed with the rest of the NEH search. In this paper this procedure is repeated twice and therefore 48 sequences would be generated.

In this manner, we can use the following rules in constructing the initial sequence vector of each stage (see Table I).

TABLE I
Criterions for Arranging the Products in Every Stage

Criterion	Abbreviation	Description
$\partial_{ij} = \frac{d_i}{p_{ij}}$	LDR/C	Longest Demand to capacity ratio according to setup cost
	LDR/S	Longest Demand to capacity ratio according to setup time
	SDR/C	Shortest Demand to capacity ratio according to setup cost
	SDR/S	Shortest Demand to capacity ratio according to setup time
$\eta_{ij} = \frac{d_i}{h_{ij}}$	LDH/C	Longest Demand to holding cost according to setup cost
	LDH/S	Longest Demand to holding cost according to setup time
	SDH/C	Shortest Demand to holding cost according to setup cost
	SDH/S	Shortest Demand to holding cost according to setup time
$\Psi_{ij} = \frac{h_{ij}}{\sum h_{ij}}$	LAH/C	Longest Average holding cost according to setup cost
	LAH/S	Longest Average holding cost according to setup time
	SAH/C	Shortest Average holding cost according to setup cost
	SAH/S	Shortest Average holding cost according to setup time
$\Phi_{ij} = \frac{p_{ij}}{h_{ij}}$	LRH/C	Longest cap_ratio to holding cost according to setup cost
	LRH/S	Longest cap_ratio to holding cost according to setup time
	SRH/C	Shortest cap_ratio to holding cost according to setup cost
	SRH/S	Shortest cap_ratio to holding cost according to setup time

- **Holding Cost Base Approach (HCB)**

Since inventory holding costs have a big share in the total cost, following algorithm tries to reduce this costs by making an appropriate sequence.

Begin

Set the products as desending order of h_i.

Let $\sigma_m(r)=n-r+1, \quad \sigma_{m-1}(r)=r.$

Until $j>1$ *do*: $j-1 \rightarrow j$.

Set the products as desending order of $h_{i,j-i} * \sum_k \frac{d_k}{p_{kj}}$ *where* $\sigma_j(k)<\sigma_j(i)$.

Let $\sigma_{j-1}(r)=n-r+1.$

End

Now all generated sequences are considered to evaluate their effects on total cost. This is done by solving the common cycle non linear model presented in Fig. 5. To solve this model, we use an optimal enumeration method (OEM) within which the mixed non linear problem (P1) is replaced by its linear counterpart as the variable *T* is supersede with *H/F* and *F=1*. In an iterative process, by increasing *F* by 1 and solving the resulting mixed linear program we try to obtain the optimal or near optimal solution (Torabi et al., [20]). This procedure repeats until total costs of all sequences are calculated and the minimum of them will be the solution of the original problem.

IV. Genetic Algorithm

To evaluate previous heuristic approach and to obtain good solution in large scale problems we present a GA as follows:

- **Chromosomes Representation**

A chromosome by our definition contains of *m* sub-chromosomes with *n* genes in each of them and *j-th* sub-chromosome represents the sequence of products in *j-th* stage. In Fig. 3, a sample chromosome for a problem with *n=4* and *m=3* is depicted.

3	2	1	4	2	4	1	3	1	2	3	4

Fig. 3 A sample chromosome

According to this chromosome representation the sequence vectors at different stages are:

$$\sigma_1 = \{3,2,1,4\} \quad \sigma_2 = \{2,4,1,3\} \quad \sigma_3 = \{1,2,3,4\}$$

This representation of chromosomes has two privileges, one that there is a unique string associated with each solution for the problem and another that all of these chromosomes are feasible because it shows a feasible sequence. Therefore, based on the above chromosome representation, for a problem with *n* products and *m* stages the length of the associated string (number of genes) is equal to *m*n*. Also the number of solutions or chromosomes is $(n!)^m$**.** This value for a problem with *n=6* and *m=4* is equal to 207,360,000 and for a problem with *n=10* and *m=10* is equal to 13,168,189,440,000. The large number of feasible points or chromosomes is a good reason for using an intelligent search tool such as GA.

- **Initial Population**

Chromosomes that are constructed by NEH and HCB heuristics will be used as initial population in GA.

- **Fitness Function**

According to the problem structure, the fitness function is the objective function of the problem except that in this case the original mixed zero-one nonlinear problem is replaced by a *NLP* one which the binary variables are now fixed according to the chosen production sequence. Therefore the setup cost is a fixed amount represented by *A*. (see Fig. 4).

Problem P2:

$$\text{Min } Z = \frac{A}{T} +$$

$$(12)\ \sum_{i=1}^{n}\left[h_i \cdot \frac{d_i}{2}\left(1-\frac{d_i}{p_{im}}\right)+\frac{d_i^2}{2}\sum_{j=2}^{m}h_{i,j-1}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right].T+\sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}d_i\left(b_{ij}-b_{i,j-1}\right)$$

Subject to:

$$(13)\ b_{i,j-1}+\frac{d_i.T}{p_{i,j-1}} \le b_{ij};\ i=1,...,n,\ j=2,...,m$$

$$(14)\ b_{\sigma_j(i-1)}+\frac{d_{\sigma_j(i-1)}.T}{p_{\sigma_j(i-1)}}+s_{\sigma_j(i-1),\sigma_j(i)} \le b_{\sigma_j(i)};\ i=1,...,n;\quad j=1,...,m;\quad k \ne i;\ \ell < n$$

$$(15)\ b_{\sigma_j(1)j} \ge s_{\sigma_j(n)\sigma_j(1)j};\ j=1,...,m;\quad i=1,...,n$$

$$(16)\ b_{im}+\frac{d_i.T}{p_{im}} \le T;\ i=1,...,n$$

$$(17)\ F*T = H$$

$$(18)\ T \ge 0, b_{ij} \ge 0;\ x_{ijl} \in \{(0,1\};\ \forall i,l,j;\ l \ne i$$

$$(19)\ F \succ = 1;\ F \in Z$$

Fig. 4 The NLP model for evaluating the fitness of chromosomes

- **Crossover Operation**

The crossover operator creates offspring sequencing by coalescing two other sequences called parents. The aim is to generate better children. Many different general and specific crossover operators have been proposed for the permutation based representation (Torabi et al., [19]; Poon and Carter, [14]). Some typical are: PMX or *partially mapped crossover*, OP-TP or *one-two point crossover*, LOX or *linear order cross over*, all of them tend to disrupt building blocks in the latter stages of the algorithm. Using these operators may create offsprings that are consistently worse than their progenitors. Due to this fact Ruis and Maroto [16] who used GA for solving the flow shop scheduling problem with sequence dependent setup times, proposed following four crossover operators in order to overcome this problem:

1. Similar Job Order crossover or (SJOX)
2. Similar Block Order crossover or (SBOX)
3. Similar Job 2- point order cross over or (SJ2OX)
4. similar block2-point order cross over or (SB2OX)

Their study revealed that the last operator (SB2OX) was superior to all other considered operators. So we used this operator with some changes in our GA. The detailed descriptions are given as follows:

Regarding to the non permutation chromosomes of our problem, each of them with *m* Permutation sub chromosomes, first of all, *j-th* sub chromosome *(j=1,…m)* of both parents are examined on a position-by-position basis. We consider blocks of at least two consecutive identical Jobs and only those identical blocks on both parents are directly copied to the children. Then, two random cut points are taken and the sections between these two points are directly copied to the children. Finally, the missing elements of each offspring are copied in the relative order of the other parents. (see Fig. 5)

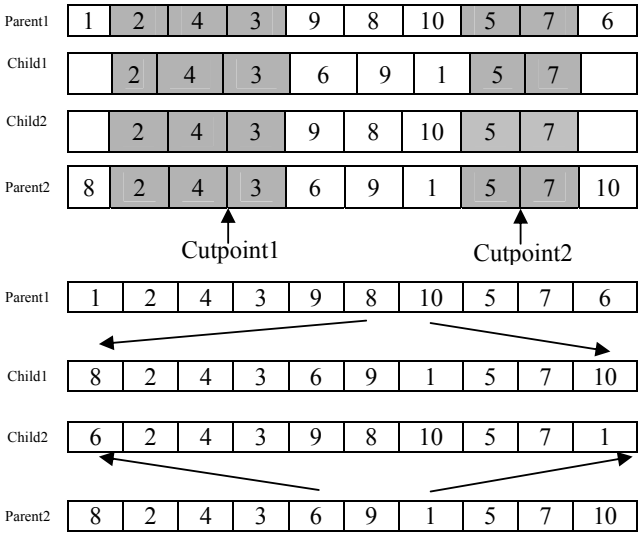


Fig. 5 SB2OX crossover operator

Now depending on having identical or different cut points for sub chromosomes, we will have two specific crossover operators for these non permutation chromosomes.

- **Mutation Operator**

A genetic algorithm includes a mutation operator to avoid convergence trapping in local optima and to introduce lost genetic material and variability in the population.

For the proposed GA, we used the SHIFT mutation where for every sub-chromosome a randomly picked position in the sequence is relocated to another randomly picked position and the jobs between two positions moving along. This operator can be changed as a cost based one if the two randomly picked positions are selected as follows:

In each stage *j, (j=1,..., m),* position of product *k* is the first position if $c_{ikj} = \max c_{ikj}$; $\forall i, j$ and position of product *l* is the second position if $c_{lkj} = \min c_{ikj}$; $\forall i$. Thus we have two randomly and cost based shift mutation operators. (see Fig. 6)

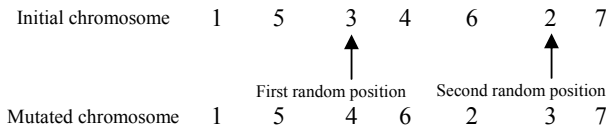


Fig. 6 Shift mutation operator

- **Population Replacement**

Next generation chromosomes are selected from the enlarged population in which all parents and offsprings have same chance for the selection. The mechanism of sampling is ($\mu + \lambda$) proposed by Batch [1]. In this strategy λ offsprings strive with μ parents for survival. It avoids from entering similar chromosomes. First generate pop_size numbers of random numbers between zero and one. Then, for each random number, if its value is smaller than crossover rate ($0 < p_c < 1$), the corresponding chromosomes will be selected as a parent. If the number of chromosomes is odd, the last chromosome won't be considered. Finally, by eliminating the similar chromosomes and replacing them with random produced chromosomes and ordering them according to the fitness function the best pop_size number of them will be selected. This process will be repeated for mutation operator. Let cross_size and mut_size denote the number of chromosomes selected to undergo the crossover and the mutation operation, respectively. Then we have:

$$Cross\_size = [\frac{\text{round}(\text{pop}\_\text{size} \times \text{P}_\text{c})}{2}] \text{ and}$$

$$mut\_size = \text{round}(\text{pop\_size} \times p_m)$$

Where P_c and P_m are crossover and mutation rates respectively and both of them are between zero and one.

Therefore in each iteration (generation) pop_size + cross_size + mut_size number of chromosome are evaluated and the best pop_size number of them will be transferred to the next generation. The termination criterion determines when the GA will stop. In other words, the genetic operations are repeated until a termination condition is met. In our implementation, we stop the GA, if the maximum number of generations or max_no_improve, is met.

- **Proposed GA**

The following is our proposed GA for the finite horizon ELSP in flow lines:

Step 1: set the GA parameters, including the population size, pop_size, the maximum number of generations, max_gen, the crossover rate, p_c, the mutation rate, P_m, and the number of generations without improvement for the last best solution max_no_imp as stopping condition.

Step 2: Generate initial population using NEH and HCB.

Step 3: Apply the optimal enumeration method (OEM) for each chromosome to calculate the related fitness value.

Step 4: follow the chromosome selection procedure to select cross_size pairs of chromosomes and perform the crossover operation.

Step 5: Evaluate the children, eliminate the similar chromosomes and select the best pop_size number of them according to the fitness value.

Step 6: Repeat the chromosome selection procedure to select mut_size number of chromosomes from the revised population and perform the mutation operation.

Step 7: Evaluate the muted chromosomes, eliminate the similar ones and select the best pop_size number of them.

Step 8: Repeat stages 4 to 7 until termination condition is met.

V. Performance Evaluation

- **Test Problems**

We used randomly generated problems to evaluate our GA. Each randomly generated problem in this paper has following range of parameters as uniform distribution:

$$di \approx U(50,500) \qquad C_{kij} \approx U(10,20)$$
$$P_{ij} \approx U(10000,20000) \quad S_{kij} \approx U(0.01,0.025)$$

To generate inventory holding cost for a given product *i* first we generate its total value from a uniform distribution between 10 and 100. Then h_i the annual inventory holding cost per unit of final product is taken to be equal to 10% of its total value. The inter-stage inventory holding costs h_{ij} are also taken to be equal to 10% of product value after stage *j*. we assume that raw materials count for 40% of the product total value and the remaining 60% (the added value) are randomly distributed among *m* stages (for more details see (Ouenniche, [1]).

- **Parameters Setting**

The different operators and levels of the parameters clearly affect the quality of the solutions obtained by a genetic algorithm. Therefore, in following three sub-sections, we explain parameters setting of our GA.

- *Selecting the Crossover and Mutation Operations*

In order to determine the GA operations, we test three types of the problem $(3\times3, 5\times2, 3\times4)$ for three times. Other parameters have been fixed as follows:

$$\text{Max-no-imp=5, number of generations=5, } P_c = 0.7, P_{m=} 0.2$$

IR: Using **I**dentical SBO2X crossover and **R**andom shift mutation

IC: Using **I**dentical SBO2X crossover and **C**ost based shift mutation

VR: Using **V**ariant SBO2X crossover and **R**andom shift mutation
VC: Using **V**ariant SBO2X crossover and **C**ost based shift mutation

Running these 36 problems by a PC with 2.0 GH speed of CPU and determining the objective functions and running times of them, now we can select our parameters.

Sum of weighted relative errors (SWRE) that is denoted follow is a weighted criteria that can Justify the results.

$$SWRE = w\sum_{i=1}^{|j|}\left(\frac{z_{ij}-z_i^*}{z_i^*}\right)+(1-w)\sum_{i=1}^{|j|}\left(\frac{t_{ij}-t_i^*}{t_i^*}\right)$$

Where:

z_{ij} = objective function of *j-th* problem in *i-th* case.

$z_i^* = \min_j\{z_{ij}\}$

t_{ij} = Run time of algorithm for solving the ***j-th*** problem in *i-th* case $t_i^* = \min_j\{t_{ij}\}$ w= superiority of time to cost

The minimum value of SWRE among IR, IC, VR and VC will be the best case.

TABLE II
SWRE CRITERION FOR SELECTING THE OPERATORS

A. CASE	**IR**	**IC**	**VR**	**VC**
w = 0.1	0.538233	0.114212	0.393396	0.877933
w = 0.3	0.564991	0.206287	0.48882	0.966872
w = 0.5	0.591749	0.298362	0.581368	1.055811

As it is shown in Table II the minimum value of SWRE in each case of w=0.1, w=0.3 and w=0.5 correspond to identical SBO2X and cost based shift mutation. These results were expected about the mutation operator. About the crossover operator selecting the identical cut points for all of the sub-chromosomes will cause decrease of $(b_{ij}-b_{i,j-1})$ which has direct effect on the objective function.

o *Determining the Initial Population and Number of Generations*

R (20): In this case 20 chromosomes are produced randomly as initial population.

R (50): In this case 50 chromosomes are produced randomly as initial population.

Heu: In this case only chromosomes are produced heuristically as initial population

RH (100): In this case in addition to 48 heuristic produced chromosomes, 52 chromosomes are produced randomly as initial population.

Also, max_gen and max_no_imp will be evaluate in four fallowing cases respectfully: (10,1), (30,5), (50,10), (nxm, m)

Running these 64 problems for SWRE criteria we found that best parameters setting is as follows: max_gen=10 and max_no_imp=1 and the initial population is heuristic produced chromosome.

Finally crossover and mutation rates after initial tests are set as $P_m = 0.7$ and $P_c = 0.2$.

- **A Lower Bound for the Problem**

In order to evaluate our GA, in this sub-section, we present a lower bound (LB) by solving the following model:

Problem P3:

$$\text{Min } Z_{LB} = \frac{A}{T} +$$

$$(20)\ \sum_{i=1}^{n}\left[h_i.\frac{d_i}{2}\left(1-\frac{d_i}{p_{im}}\right)+\frac{d_i^2}{2}\sum_{j=2}^{m}h_{i,j-1}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right].T+\sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}d_i\left(b_{ij}-b_{i,j-1}\right)$$

Subject to:

$$(21)\ b_{i,j-1}+\frac{d_i.T}{p_{i,j-1}}\le b_{ij};\quad i=1,...,n,\ j=2,...,m$$

$$(22)\ b_{k,j}+\frac{d_k.T}{p_{k,j}}+s_{i,j}-bi,j\le M(2-x_{k,l}-x_{i,l+1});\ \forall i,j\ ;k\ne i\ ;\ell<n$$

$$(23)\ b_{i,j}\ge s_{i,j}*x_{i,1};\ j=1,...,m;\quad i=1,...,n$$

$$(24)\ b_{im}+\frac{di.T}{p_{im}}\le T;\ i=1,...,n$$

$$(25)\ F*T=H$$

$$(26)\ T\ge 0, b_{ij}\ge 0; x_{ij}\in\{(0,1\}\ ;\forall i,j$$

$$(27)\ F>=1; F\in Z$$

This model has following specifications:

1- By replacing the following relations, the model will be relaxed from sequence dependent setups to sequence independent setups:

$$A = n*m*average\{C_{k,i,j}\} \qquad S_{i,j} = \underset{k}{average}\{S_{k,i,j}\}$$

2- Sequencing Variables have been eliminated from the model

3- The problem statement has been changed from non permutation to the permutation case. This will reduce the zero_one variables to $\frac{1}{m}$ comparing to Problem P1 and is no more in quadratic forms.

- **Computational Results**

Finally to evaluate the proposed GA six different problem sizes with 5 or 10 products and 2, 5 or 10 stages have been considered. The parameters for each problem have been generated three times. These 18 test problems are solved by the heuristic method. Furthermore they are solved by GA three times and the best has been considered. Finally their

lower bound has been determined using LINGO 6.0 as an optimal solver for the mathematical models. All of 12 results are presented in Table III:

TABLE III
COMPUTATIONAL RESULTS

Pro_size	B. Best Heuristic Solution			Best GA in 3 runs		LB(lingo)
n*m_Run	Heuristc Name	Obj value	Running Time	Obj value	Running Time	Obj value
5*2_1	LAH/S	23239	93.04	23239	1929.4	22107
5*2_2	LDR/C	13603	92.032	12870	2163.4	12113
5*2_3	LAH/S	20195	86.585	18878	2018.8	18063
5*5_1	HCB	54309	97.59	52212	2565.3	51775
5*5_2	HCB	41472	118.46	37969	2836.3	37002
5*5_3	HCB	46846	107.475	37800	2819.5	36219
5*10_1	HCB	68500	94.897	65331	2853.1	63593
5*10_2	HCB	57668	127.663	55668	2623.8	52834
5*10_3	HCB	55538	111.371	53347	3228.9	51024
10*2_1	LAH/C	515200	99.804	48362	2393.7	46823
10*2_2	LDR/S	44512	109.407	42050	2426.1	41002
10*2_3	LDR/S	42618	100.818	41730	2019	40197
10*5_1	HCB	101700	87.92	95193	2803.2	87314
10*5_2	HCB	94469	102.63	87232	2768.3	80217
10*5_3	HCB	126340	91.11	120240	2834.2	109504
10*10_1	HCB	162200	260.29	188370	6923.5	*
10*10_2	HCB	121560	251.35	119450	7542.3	*
10*10_3	HCB	141060	261.12	129010	8381.4	*

Let Z denotes the total cost obtained via GA, we can calculate its corresponding performance ratio as the $\theta = (z - z_{LB} / z_{LB}) * 100$

TABLE IV
COMPARING θ RATIO FOR THE TEST PROBLEMS

Problem numbers	1	2	3	4	5	6	7	8	9
θ	4.9	6.2	4.5	0.8	2.6	4.3	2.7	5.3	4.5
Problem numbers	10	11	12	13	14	15	16	17	18
θ	3.2	0.2	3.8	9.02	8.7	9.8	*	*	*

Largest performance ratio for the collection of instances is 9.8%. It should be noted that the solutions obtained from the problem P3 are often infeasible due to eliminating sequencing constrains, which means that the optimal cost of the original problem P1 often would be greater than the lower bound.

Although with consuming a considerable time we can obtain better solution by GA, computational results obviously show that the heuristic method obtains a feasible solution very soon without much running time even with increasing the problem size.

VI. CONCLUSION

In this paper, the common cycle approach has been used to formulate the economic lot scheduling problem in flow line systems where the machines setup times and costs are sequence dependent. First, two new mixed zero-one nonlinear models are developed to solve the problem in finite and infinite horizon cases. Then, to avoid solving the complex mixed non-linear program directly, a heuristic method is suggested to determine its near optimal solution. Finally, a genetic algorithm with special operations is suggested to improve the heuristic solution. Computational results indicate that the heuristic could give a good feasible solution in a short time, and the performance of GA is very promising.

REFERENCES

[1] T. Batch, Selective Pressure in Evolutionary Algorithms. IEEE press1, NJ, 57-62(1994).
[2] E.A. Bomberger, Dynamic programming approach to a lot size scheduling problem. Management science. 12:778-784 (1996).
[3] G. Dobson, The ELSP: Achieving feasibility using time-varing lot sizes. Operations research. 35:764-771 (1987).
[4] W.C. Driscoel, H. Emmoss, Scheduling production on the one machine with change over costs. AIIE Transactions. 9:388-395 (1977)
[5] S. Fujita, The application of marginal analysis to the ELSP. AIIETransactrion. 10:354-361 (1978).
[6] G. Gallego, D.X. Show, Complexity of the ELSP with general cyclic schedules. IEEE Transactions. 29:109-113 (1997).
[7] A.M. Geoffrion, G.W. Graves, Scheduling parallel production lines with change over costs. Operations Research. 24:595-610 (1976).
[8] O.H. Hong-Choon, I.A. Karimi, Planning production on a single processor with sequence dependent setups: part1, computer and chemical enginnering. 5:1021-1030 (2001).
[9] O.H. Hong-Choon, I. Karimi, A planning production on a single processor with sequence dependent setups: part2, Computer and chemical enginnering. 25:1031-1043 (2001).
[10] S.A. Irani, U. Gunasena, Single machine setup dependent sequencing using a setup complexity ranking scheme. Journal of manufacturing systems. 7:11-23 (1988)
[11] B. Karimi, S.M.T. Fatemi Ghomi, J.M. Wilson, The capacitated lot sizing problem: a review of models and algorithms. Omega. 31:365-378 (2003)
[12] W.L. Maxwell, The scheduling of economic lot sizes. Navel Research logistics Quartery.11:89-124 (1964).
[13] J. Ouenniche, The impact of sequencing decisions on multi-item lot sizing and scheduling in flowshops. International Journal of operational research. 37: 2253-2270 (1999).
[14] P.W. Poon, J.N. Carter, Genetic algorithm crossover operators for ordering applications. Computers and Operations Research. 22:135-147 (1995).
[15] J. Roger, A computational approach to the economic lot scheduling problem. Journal of Management Science. 4: 264-291 (1958).
[16] R. Ruis, C. Maroto, Solving the flowshop scheduling problem with sequence dependent setup times using advanced metaheuristic. European Journal of Operational Research. Article in press (2004).
[17] H. Singh, J.B. Foster, Production scheduling with sequence dependent setup costs. IEE Transaction. 19:43-49 (1987).
[18] G. Taylor, M. Taha, A heuristic model for the sequence-dependent lot scheduling problem. Production planning and control. Vol8. 3: 213-225(1997).
[19] S.A. Torabi, S.M.T. Fatemi Ghomi, B. Karimi, A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chain. European journal of operational research. Article in press (2005).
[20] S.A. Torabi, B. Karimi, S.M.T. Fatemi Ghomi, The common cycle economic lot scheduling in flexible job shops: the finite horizon case. International Journal of Production Economics. 97: 52-65 (2004).
[21] J. Wagner, J. Davis, A search heuristic for the sequence-dependent economic lot scheduling problem. European Journal of Operational Research. 141:133-146 (2001).