FINITE HORIZON ECONOMIC LOT AND DELIVERY SCHEDULING PROBLEM: FLEXIBLE FLOW LINES WITH UNRELATED PARALLEL MACHINES AND SEQUENCE DEPENDENT SETUPS

M. Jenabi and S. M. T. Fatemi Ghomi

Department of Industrial Engineering, Amirkabir University of Technology
P.O. Box 15916-34311, Tehran, Iran
m.jenabi@aut.ac.ir - fatemi@aut.ac.ir

S. A. Torabi*

Department of Industrial Engineering, Faculty of Engineering
University of Tehran, P.O. Box 11155/4563, Tehran, Iran
satorabi@ut.ac.ir

*Corresponding Author

(**Received: September 8, 2007 – Accepted in Revised Form: November 22, 2007)**

Abstract This paper considers the economic lot and delivery scheduling problem in a two-echelon supply chains, where a single supplier produces multiple components on a flexible flow line (FFL) and delivers them directly to an assembly facility (AF). The objective is to determine a cyclic schedule that minimizes the sum of transportation, setup and inventory holding costs per unit time without shortage. We have developed a new mixed zero-one nonlinear mathematical model for the problem. Due to the difficulty of obtaining the optimal solution, especially in the instances of medium and large-sized problems, two meta-heuristic algorithms (HGA and SA) are proposed and evaluated over randomly generated problems. Computational results indicate that the proposed HGA outperforms the SA algorithm with respect to both the solution quality and computation times especially in large-size problems.

Keywords Flexible Flow Lines, Lot and Delivery-Scheduling, Unrelated Parallel Machines, Finite Planning Horizon, Hybrid Genetic Algorithm, Simulated Annealing

چکیده در این مقاله، مسئلهٔ زمان بندی تولید و تحویل انباشته‌های اقتصادی در یک زنجیرهٔ تأمین دو سطحی مورد بررسی قرار می‌گیرد. در این زنجیره، یک تأمین‌کننده چندین محصول را در یک سیستم تولید جریان خطی انعطاف‌پذیر تولید کرده و آنها را مستقیما" به یک مونتاژگر (مشتری) تحویل می‌دهد. هدف تعیین یک زمان بندی سیکلی به نحوی است که مجموع هزینه‌های حمل و نقل، راه‌اندازی و نگهداری موجودی‌ها در واحد زمان بدون بروز کمبود موجودی حداقل شود. یک مدل ریاضی غیرخطی ترکیبی صفر و یک برای مسئلهٔ فوق توسعه داده شده است. با توجه به پیچیدگی حل این مدل در مسائل با مقیاس متوسط و بزرگ، دو الگوریتم فوق ابتکاری (الگوریتم ژنتیک ترکیبی و شبیه‌سازی تبرید) طراحی شده است و بر تعدادی مسئله نمونه تست گردیده است. نتایج بدست آمده از آزمایش های عددی، برتری الگوریتم ژنتیک طراحی شده را هم به لحاظ کیفیت جواب‌های حاصله و هم زمان محاسباتی موردنیاز بر الگوریتم شبیه‌سازی تبرید به اثبات می‌رساند.

1. INTRODUCTION

Increasing global competition forces all members of supply chains to optimize their activities to achieve higher level of customer satisfaction. One of the main issues in supply chains is the efficient and effective management of material flow [1]. In this regard, a smooth and cost-efficient production and also components delivery between adjacent parties of a supply chain often depends on selecting appropriate lot-size, production and delivery schedules.

One of the most famous lot-sizing problems is the economic lot-scheduling problem (ELSP). Original ELSP concerns with lot sizing and scheduling of several items in a single stage production facility, so that corresponding demands are met without shortage, and the average inventory and setup costs per unit time are minimized [2]. Researches on the ELSP have usually focused on cyclic schedules (i.e., schedules that are repeated periodically) with three scheduling policies: common cycle, basic period and time varying lot size approaches. Several authors have extended the classical ELSP to more general systems such as single stage systems with parallel machines, flow shop, and job shop systems [3-5].

The economic lot and delivery scheduling problem (ELDSP) is an extension of ELSP into a two-stage supply chain environment, where a supplier produces several items for an assembly facility and delivers them directly. Hahm, et al [6] introduced the single item ELDSP, and then extended it to multiple items case through two other research works. In the first paper [7], they used the common cycle for all components and assumed that the time between deliveries is equal to the duration of the common production cycle. In the second one [8], they assumed that multiple deliveries within a global production cycle are allowed (i.e., the nested schedule case). Khouja [9] considered ELDSP for a supplier that uses a volume flexible production system where component quality depends on both lot sizes and unit production times and developed an algorithm to solve the problem. Jensen, et al [10] developed an optimal polynomial time algorithm for ELDSP under common cycle approach. Vergara, et al [11] extended ELDSP to multiple supplier, multiple components supply chain and proposed an evolutionary algorithm (EA) to obtain an optimal, or near optimal, synchronized delivery cycle time and suppliers' component sequences.

In all above works, it is assumed that the planning horizon is infinite and the production system of supplier(s) as a single production line or machine. However, these assumptions considerably reduce the usefulness of the proposed contributions, because in practice, planning horizons and production systems are often finite and multi-stage [4].

Literature review in finite horizon case reveals that there are few contributions. Ouenniche, et al [4] studied the finite horizon ELSP in job shops under common cycle approach and developed an optimal solution method to solve the problem. In another research work, Ouenniche, et al [12] considered this problem using the multiple cycle approach, and developed an efficient heuristic method to obtain a near optimal solution. Recently, Torabi, et al [13] presented a new model for the finite horizon ELSP in flexible job shops (i.e. job shop systems with parallel machines at least at one stage) under the common cycle approach, and developed an optimal solution method for this problem. Moreover, Torabi, et al [3] studied the finite horizon ELDSP in flexible flow lines with identical parallel machines and sequence-independent setup times/costs at each stage under the common cycle approach. They proposed an efficient hybrid genetic algorithm to obtain near optimal (or ideally optimal) solutions for the problem.

In this paper, we extend our previous research work [3] and study the finite horizon ELDSP in flexible flow lines with unrelated parallel machines (i.e., machines with different characteristics such as production rates and/or setup times), and sequence-dependent setup times/costs at each stage. The objective is to determine a cyclic schedule that minimizes the sum of transportation, setup and inventory holding costs per unit time in the supply chain without any stock-outs.

To solve the problem, the common cycle strategy is used for all components. That is, at each production cycle, one lot of each component at each stage is produced. Also, it is required that the planning horizon is an integer multiple (F) of the common cycle length (T). We have developed a new mixed zero-one nonlinear program whose optimal solution determines simultaneously the optimal assignment of components to machines at stages with multiple parallel machines, the optimal sequence of components on each machine at each stage, the optimal lot sizes and the optimal production and delivery schedule for each production run.

The rest of the paper is outlined as follows. Problem assumptions, notations and mathematical formulation are presented in Section 2. The proposed HGA and SA algorithms are introduced in Sections 3 and 4, respectively. In Section 5, numerical experiments and corresponding computational results are shown. Finally, Section 6 is devoted to some concluding remarks.

2. PROBLEM FORMULATION

2.1. Problem Assumptions

The following assumptions are considered for the problem formulation:

- The lots of each component are of equal size at different stages.
- Machines at stages, with multiple parallel machines that can be identical (in all characteristics such as production rates and setup times/costs) or non-identical (unrelated), but at least one stage must have unrelated parallel machines.
- Machines of different stages are continuously available and each machine can only process one component at a time.
- At stages with parallel machines, each component is processed entirely on one machine.
- The structure of setup times and costs at the supplier are sequence dependent.
- The production sequence on each machine at each stage is unique and is determined by the solution method.
- The supplier incurs linear inventory holding costs on semi-finished components.
- Both the supplier and the assembler incur linear holding costs on end components.
- Lot splitting and pre-emption are not allowed.
- There are unlimited buffers between adjacent stages.
- Total capacities of different stages are sufficient to meet the demands; thus there exists at least one feasible schedule.
- Zero switch rule is used which means the production of each component at each cycle begins when its inventory level reaches zero.

2.2. Model Notations

The notations used for the problem formulation are defined as follows:

2.2.1. Parameters

n	Number of components
m	Number of work centers (stages)
i, u	Component indices
j	Stage index
m_j	Number of parallel machines at stage j (may be unrelated or identical)
M_{kj}	K-th machine at stage j
d_i	Demand rate of component i
p_{ikj}	Production rate of component i on machine k at stage j
pt_{ikj}	Processing time for a lot of component i on machine k at stage j ($pt_{ikj} = d_i.T/p_{ikj}$)
s_{iukj}	Setup time from component i to component u on machine k at stage j
sc_{iukj}	Setup cost from component i to component u on machine k at stage j
h_{ij}	Holding cost per unit of component i per unit time between stages j and j + 1
h_i	Holding cost per unit of final component i per unit time (both at the supplier and at the assembler)
A	Transportation cost per delivery
H	Planning horizon length
M	A large real number

2.2.2. Decision variables

σ_{kj}	Production sequence vector at machine M_{kj}
n_{kj}	Number of components assigned to machine M_{kj}
T	Common production and delivery cycle length
Q_i	Production lot size of component i at different stages ($Q_i = d_i.T$)
F	The number of production cycles over the planning horizon
b_{ij}	Process beginning time of component i at stage j (after related setup operation)

It is noted that at stages with only one machine the value of m_j and index k would be only one. Since after processing each component at each stage, there would be a value added for the component, values of h_{ij} parameters will be non-decreasing. In other words, we have: $h_{i,j-1} < h_{ij}$; $i = 1, \ldots, n$, $j = 2, \ldots, m-1$.

2.3. Objective Function The objective of the problem (Problem P) is to minimize the average of transportation, setup, work-in-process and end component inventory holding costs per unit time. The average delivery cost per unit time is A/T. The setup cost expression consists of two parts: the first part computes the setup cost of the first component which comes after the last assigned component to M_{kj}. The second part computes the setup costs of the subsequent products which are dependent on the processing sequence on M_{kj}. This expression would be as follow:

$$C_{setup} = \frac{1}{T}\left\{\sum_{j=1}^{m}\sum_{k=1}^{m_j}\sum_{i=1}^{n}\sum_{u=1,u\neq i}^{n}\sum_{\ell=2}^{n} sc_{uikj}.x_{u\ell kj}.x_{i1kj} + \sum_{j=1}^{m}\sum_{k=1}^{m_j}\sum_{i=1}^{n}\sum_{u=1,u\neq i}^{n}\sum_{\ell=2}^{n} sc_{iukj}.x_{u\ell kj}.x_{i,\ell-1,kj}\right\} \quad (1)$$

The inventory holding costs are incurred at both the supplier and the assembler. The inventory level for final component i at each cycle at the AF have a simple saw tooth form. Therefore, the average inventory of component i per unit time at the AF is (1/T) $\{(d_iT).T/2\}$ and then the total average holding cost per unit time at the AF would be: (1/2) $(\sum_i d_ih_iT)$.

Two types of inventories are considered for the supplier: WIP inventory and finished product inventory. Figures 1 and 2 show the evolution of WIP inventory of component i between two successive stages j-1 and j, and the inventory level of final component i, respectively.

From Figure 1 it is obvious that the average WIP inventory of component i between two successive stages j-1 and j per unit time is:

$$I_{i,j-1} = \frac{1}{T}\left\{\frac{d_i^2T^2}{2}\sum_{k=1}^{m_{j-1}}\sum_{\ell=1}^{n}\frac{x_{i\ell k,j-1}}{p_{ik,j-1}} + d_iT\left(b_{ij} - b_{i,j-1} - d_iT\sum_{k=1}^{m_{j-1}}\sum_{\ell=1}^{n}\frac{x_{i\ell k,j-1}}{p_{ik,j-1}}\right) + \frac{d_i^2T^2}{2}\sum_{k=1}^{m_j}\sum_{\ell=1}^{n}\frac{x_{i\ell kj}}{p_{ikj}}\right\} =$$

$$d_i\left(b_{ij} + d_iT\sum_{k=1}^{m_j}\sum_{\ell=1}^{n}\frac{x_{i\ell kj}}{2p_{ikj}} - b_{i,j-1} - d_iT\sum_{k=1}^{m_{j-1}}\sum_{\ell=1}^{n}\frac{x_{i\ell k,j-1}}{2p_{ik,j-1}}\right) \quad (2)$$

Therefore, the total WIP holding cost for all components per unit time at the supplier is:

$$TC_{WIP} = \sum_{i=1}^{n}\sum_{j=2}^{m} h_{i,j-1}d_i\left\{b_{ij} + d_iT\sum_{k=1}^{m_j}\sum_{\ell=1}^{n}\frac{x_{i\ell kj}}{2p_{ikj}} - b_{i,j-1} - d_iT\sum_{k=1}^{m_{j-1}}\sum_{\ell=1}^{n}\frac{x_{i\ell k,j-1}}{2p_{ik,j-1}}\right\} \quad (3)$$

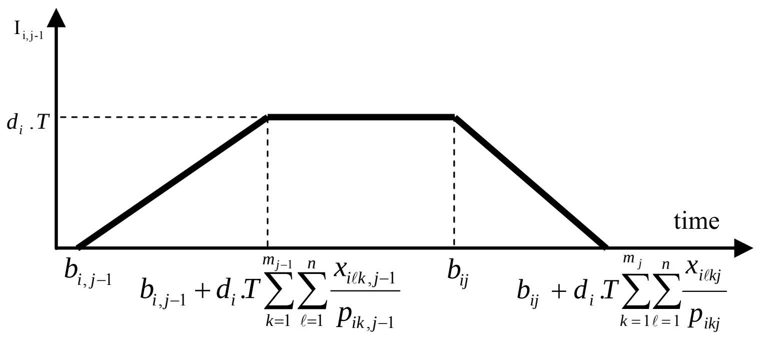
From Figure 2, the average inventory of final component i per unit time would be:

$$I_{im} = \frac{1}{T}\left\{\frac{d_iT}{2}.d_iT\sum_{k=1}^{m_m}\sum_{\ell=1}^{n}\frac{x_{i\ell km}}{p_{ikm}} + d_iT\left(T - b_{im} - d_iT\sum_{k=1}^{m_m}\sum_{\ell=1}^{n}\frac{x_{i\ell km}}{p_{ikm}}\right)\right\}$$

$$= d_i\left(1 - d_i.\sum_{k=1}^{m_m}\sum_{\ell=1}^{n}\frac{x_{i\ell km}}{2p_{ikm}}\right)T - d_ib_{im} \quad (4)$$

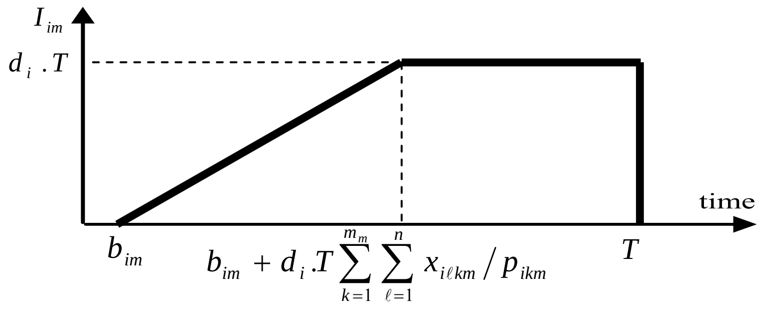
Thus, the total inventory holding cost for all final components per unit time is:

$$TC_{FI} = \sum_{i=1}^{n} h_i.d_i\left(1 - d_i\sum_{k=1}^{m_m}\sum_{\ell=1}^{n}\frac{x_{i\ell km}}{2.p_{ikm}}\right).T - \sum_{i=1}^{n} h_i.d_i.b_{im} \quad (5)$$



$I_{i,j-1}$
$d_i.T$
time
$b_{i,j-1}$
$b_{i,j-1}+d_i.T\sum_{k=1}^{m_{j-1}}\sum_{\ell=1}^{n}\frac{x_{i\ell k,j-1}}{p_{ik,j-1}}$
b_{ij}
$b_{ij}+d_i.T\sum_{k=1}^{m_j}\sum_{\ell=1}^{n}\frac{x_{i\ell kj}}{p_{ikj}}$

Figure 1. WIP inventory between stages j-1 and j.



I_{im}
$d_i.T$
time
b_{im}
$b_{im}+d_i.T\sum_{k=1}^{m_m}\sum_{\ell=1}^{n}x_{i\ell km}/p_{ikm}$
T

Figure 2. Final product inventory.

Therefore, the total cost per unit time (i.e. objective function of Problem P) would be:

$$
\begin{aligned}
\mathrm{TC} = {} & \\
& \frac{1}{T}\Big\{\sum_{j=1}^{m}\sum_{k=1}^{m_j}\sum_{i=1}^{n}\sum_{u=1,u\neq i}^{n}\sum_{\ell=2}^{n} sc_{uikj}.x_{u\ell kj}.x_{i1kj} \\
& + \sum_{j=1}^{m}\sum_{k=1}^{m_j}\sum_{i=1}^{n}\sum_{u=1,u\neq i}^{n}\sum_{\ell=2}^{n} sc_{iukj}.x_{u\ell kj}.x_{i,\ell-1,kj} \\
& + A\Big\} + \sum_{i=1}^{n}\left[\frac{1}{2}h_i.d_i.\left(3-d_i\sum_{k=1}^{m_m}\sum_{\ell=1}^{n}\frac{x_{i\ell km}}{p_{ikm}}\right)+\frac{1}{2}d_i^2.\right. \\
& \left.\sum_{j=2}^{m}h_{i,j-1}\left(\sum_{k=1}^{m_j}\sum_{\ell=1}^{n}\frac{x_{i\ell kj}}{p_{ikj}}-\sum_{k=1}^{m_{j-1}}\sum_{\ell=1}^{n}\frac{x_{i\ell k,j-1}}{p_{ik,j-1}}\right)\right] \\
& .T + \sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}.d_i\left(b_{ij}-b_{i,j-1}\right)-\sum_{i=1}^{n}h_i d_i b_{im}
\end{aligned}
\tag{6}
$$

2.4. Proposed Mathematical Model A mixed zero-one nonlinear model, Problem P, has been developed to solve the problem which is presented in Figure 3.

Constraints (8) state that no component can be processed before it is completed at the previous stage. Constraints (9) show that no product can be processed before the completion of its predecessor in the related production sequence (σ_{kj}). Constraints (10) secure that each product has a unique position in the sequence of one machine at each stage and Constraints (11) show that at each position of each machine, there is at most one component; because for each machine, it may be assigned less than n components. Constraints (12) state that one component can be assigned at one position of each machine; if another product is to be assigned at the previous position of this machine.

Constraints (13) show that if component i is the first component in the sequence of one machine at

$$\text{Min } Z = \frac{1}{T}\left\{ \sum_{j=1}^{m} \sum_{k=1}^{m_j} \sum_{i=1}^{n} \sum_{u=1, u \neq i}^{n} \sum_{\ell=2}^{n} sc_{uikj}.x_{u\ell kj}.x_{i1kj} + \right.$$

$$\left. \sum_{j=1}^{m} \sum_{k=1}^{m_j} \sum_{i=1}^{n} \sum_{u=1, u \neq i}^{n} \sum_{\ell=2}^{n} sc_{iukj}.x_{u\ell kj}.x_{i,\ell-1,kj} + A \right\} + \tag{7}$$

$$\sum_{i=1}^{n} \left[\frac{1}{2} h_i.d_i.\left(3 - d_i \sum_{k=1}^{m_m} \sum_{\ell=1}^{n} \frac{x_{i\ell km}}{p_{ikm}} \right) + \frac{1}{2} d_i^2 . \sum_{j=2}^{m} h_{i,j-1} \left(\sum_{k=1}^{m_j} \sum_{\ell=1}^{n} \frac{x_{i\ell kj}}{p_{ikj}} - \sum_{k=1}^{m_{j-1}} \sum_{\ell=1}^{n} \frac{x_{i\ell k,j-1}}{p_{ik,j-1}} \right) \right] .T +$$

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

Subject to :

$$b_{i,j\text{-}1} + d_i.T. \sum_{k=1}^{m_{j-1}} \sum_{\ell=1}^{n} \frac{x_{i\ell k,j-1}}{p_{ik,j-1}} \leq b_{ij} \; ; i = 1,...,n, \; j = 2,...,m \tag{8}$$

$$b_{ij} + \frac{d_i.T}{p_{ikj}} + s_{iukj} - b_{uj} \leq M\left(2 - x_{i\ell kj} - x_{u,\ell+1,kj} \right); \; i = 1,...,n, \; j = 1,...,m, \; u \neq i, \; k = 1,...,m_j, \; \ell < n \tag{9}$$

$$\sum_{k=1}^{m_j} \sum_{\ell=1}^{n} x_{i\ell kj} = 1 \; ; j = 1,...,m, \; i = 1,...,n \tag{10}$$

$$\sum_{i=1}^{n} x_{i\ell kj} \leq 1 \; ; \; j = 1,...,m, \; k = 1,...,m_j, \; \ell = 1,...,n \tag{11}$$

$$\sum_{i=1}^{n} x_{i,\ell+1,kj} \leq \sum_{i=1}^{n} x_{i\ell kj}; \; j = 1,...,m, \; k = 1,...,m_j, \; \ell < n \tag{12}$$

$$b_{ij} \geq \sum_{k=1}^{m_j} \sum_{u=1, u \neq i}^{n} \sum_{\ell=2}^{n} s_{uikj}.x_{u\ell kj}.x_{i1kj} \; ; j = 1,...,m, \; i = 1,...,n \tag{13}$$

$$b_{im} + d_i.T. \sum_{k=1}^{m_m} \sum_{\ell=1}^{n} \frac{x_{i\ell km}}{p_{ikm}} \leq T \; ; i = 1,...,n, \tag{14}$$

$$F.T = H \tag{15}$$

$$F \geq 1, \text{ and integer} \tag{16}$$

$$T \geq 0, \; b_{ij} \geq 0 \; ; \forall i,j, \; x_{i\ell kj} = \{0,1\}; \forall i,\ell,k,j. \tag{17}$$

Figure 3. Mathematical model of problem P.

stage j, it's processing cannot get started before setting up the corresponding machine.

Constraints (14) assures the resulting schedule is cyclic so that the process completion time for each component at the final stage is less than or equal to the cycle time. Constraint (15) implies that the planning horizon H is an integer multiple of common cycle T. Constraints (16) show that F is an integer greater than or equal to one. Finally, Constraints (17) indicate the type of variables.

It is noteworthy that finding the optimal solution of the original ELSP and consequently the ELDSP (i.e., considering a single machine with sequence-independent setup times/costs as supplier's production system) is known to be quite difficult due to the NP-hardness of the problem [14]. In this paper, we deal with an extended version of the ELDSP where supplier's production system is assumed to be a flexible flow line with unrelated parallel machines and sequence-dependent setup times/costs at each stage. Therefore, it is obvious that our generalized problem is definitely NP-Hard. This issue itself justify applying the heuristic or meta-heuristic approaches instead of optimal ones for the problem. In this paper, we develop two meta-heuristics i.e., a hybrid genetic algorithm (HGA) and a simulated annealing (SA) algorithm, and provide corresponding numerical results.

3. PROPOSED HYBRID GENETIC ALGORITHM

Genetic algorithms (GAs) have been proved to be highly successful in solving combinatorial optimization problems where the search space is highly unstructured or the standard techniques like the branch and bound method fail to provide efficient solutions. This intelligent stochastic optimization technique is based on the mechanism of natural selection and genetics. To improve the solution quality of GA and to overcome the problem of converging to local optima, various strategies of hybridization have been suggested to improve the performance of the simple GA [15-17]. Usually, in a typical HGA, a neighbourhood search (NS) heuristic, acts as a local improver into a basic GA loop.

3.1. Chromosome Representation Component sequences have been used for chromosome representation. For example, in a five components problem a chromosome could be [13245]. Such a vector by itself does not specify the complete solution for a discrete part of the problem. Thus, we need to develop an appropriate procedure to construct a complete discrete solution for every given permutation vector (i.e., determining the sequence vectors σ_{kj}). So, we have developed a simple heuristic approach to construct the complete discrete solution from a given permutation (for example V) in Figure 4. This heuristic is presented at following.

The basic idea behind this heuristic is to use the costs of our problem to construct a complete discrete solution from a known vector V. The products based on vector V and their dependent setup costs are assigned to machines of each stage then those are arranged in a non-decreasing order of their process completion times. These make fewer setup costs, flow time of products and consequently holding costs.

3.2. Initial Population Generation Several simple and effective constructive heuristics have been used to find good initial permutation vectors. Five constructive heuristics, i.e., the CDS (Campbell, Dudek, and Smith [18]) heuristic, the SPT heuristic, the LPT heuristic, the WSPT heuristic, and the HY heuristic [7], have been used. To apply the above heuristics it is required to estimate the processing time of each component at each stage. So, the following equation has been used:

$$p'_{uj} = \sum_{k=1}^{m_j} p_{ukj} \Big/ m_j \Rightarrow \; pt'_{uj} = \frac{d_u}{p'_{uj}} + \min_i \left\{ \sum_{k=1}^{m_j} s_{iukj} \Big/ m_j \right\} \; ; u = 1,\dots,n, \;\; j = 1,\dots,m \tag{18}$$

Through utilizing the above heuristics, we can obtain at most 5 m-1 different permutation vectors. If the initial population size generated by the above heuristics is greater than the tuned population size (in our algorithm, n), we can delete the worst excessive vectors (in terms of corresponding fitness values). Otherwise, the remaining distinct permutation vectors can be generated randomly.

```
for j = 1 to m
   if $m_j = 1$; $\sigma_{1j} = V$
   else
      while $V \neq \varnothing$
            u = first component at V.
            for k = 1 to $m_j$
               if $\sigma_{kj} = \varnothing$; $SC(k) = \min(sc_{iukj})$ $i = 1,\ldots, n$, $i \neq u$, $i \in V$ (product i' is result of this minimum)
               else; $SC(k) = sc_{u'ukj}$. u′ is the last component which previously assigned to $\sigma_{kj}$.
               end
            end
            min_SC = min {SC} $k = 1,\ldots, m_j$.
            if min_SC is related to a machine with empty sequence vector ($\sigma_{kj}$);
              assign components i' and u to that sequence vector and delete them from V.
            else; assign component u at the end of $\sigma_{kj}$. Delete component u from V.
            end
      end
   end
   V = non-decreasing order of components completion times at stage j.
end
```

Figure 4. Pseudo code for constructing a complete discrete solution.

3.3. Fitness Evaluation Function In order to mimic the natural process of survival of the fittest, a fitness value via a fitness evaluation function is assigned to each member of the population. In our problem, the fitness value of each chromosome has been obtained by solving the corresponding NLP model (Problem P1) presented in Figure 5. Note that the chromosomes with a lower cost imply the better solutions.

Problem P1 is derived from problem P by substituting $x_{i\ell kj}$ values by corresponding ones. Also, $\sigma_{kj}(i)$ indicates the i-th component on sequence vector of M_{kj}. This problem can be solved by the following iterative procedure:

3.3.1. Iterative step Let F = 1, and solve the corresponding linear program.

3.3.2. Iterative step Increase F by 1 and solve the corresponding linear program for this new value of F. If this model has no feasible solution, stop; else, if the objective function for current value of F (Z_F) is less than this value for previous F (Z), then set $Z = Z_F$ and $T^* = H/F$, and go to the next iteration.

3.4. Parent Selection The tournament selection approach has been adopted for choosing some parents [15]. It randomly chooses two individuals from the parent pool, and then chooses the fittest one if a random generated value (r) is smaller than a pre-set probability value φ ($0.5 < \varphi < 1$). Otherwise, the other one is chosen. The unselected individual is returned to the parent pool, and can be chosen again as a parent. This process is repeated until the mating pool is filled. We have also used the spouse duplication method to duplicate and to select pairs of chromosomes as a parent to undergo the crossover operation [19].

3.5. Crossover Operator The main purpose of the crossover operator is to exchange information between randomly selected parent chromosomes with the aim of producing better offspring. Based on the permutation representation of solutions, several crossover operators have been suggested [20,21,3]. In this problem, we have used the similar job 2-point order crossover (SJ2OX) operator based on our initial experiments. This crossover operator works as follows:

$$
\min\ Z = \frac{1}{T}\left[\sum_{j=1}^{m}\sum_{k=1}^{m_j} sc_{\sigma_{kj}\left(n_{kj}\right),\sigma_{kj}(1),kj} + \sum_{j=1}^{m}\sum_{k=1}^{m_j}\sum_{i=2}^{n_{kj}} sc_{\sigma_{kj}(i-1),\sigma_{kj}(i),kj} + A\right] +
$$

$$
\frac{T}{2}\left[\sum_{k=1}^{m_m}\sum_{i=1}^{n_{kj}} d_{\sigma_{km}(i)}\cdot\left(3-\frac{d_{\sigma_{km}(i)}}{p_{\sigma_{km}(i),km}}\right).h_{\sigma_{km}(i)} + \sum_{j=2}^{m}\sum_{k=1}^{m_j}\sum_{i=1}^{n_{kj}}\frac{d^2_{\sigma_{kj}(i)}}{p_{\sigma_{kj}(i),kj}} - \right. \tag{19}
$$

$$
\sum_{j=2}^{m}\sum_{k=1}^{m_{j-1}}\sum_{i=1}^{n_{kj}}\frac{d^2_{\sigma_{k,j-1}(i)}}{p_{\sigma_{k,j-1}(i),k,j-1}} + \sum_{i=1}^{n}\sum_{j=2}^{m} h_{i,j-1}.d_i\left(b_{ij}-b_{i,j-1}\right) - \sum_{i=1}^{n} h_i d_i b_{im}
$$

Subject to:

$$
b_{\sigma_{k,j-1}(i),j-1} + \frac{T.d_{\sigma_{k,j-1}(i)}}{p_{\sigma_{k,j-1}(i),k,j-1}} \le b_{\sigma_{k,j-1}(i),j};\ i \in J_{k,j-1}, j = 2,\ldots,m, k = 1,\ldots,m_{j-1} \tag{20}
$$

$$
b_{\sigma_{kj}(i-1),j} + \frac{T.d_{\sigma_{kj}(i-1)}}{p_{\sigma_{kj}(i-1),kj}} - s_{\sigma_{kj}(i-1),\sigma_{kj}(i),kj} \le b_{\sigma_{kj}(i),j};\ i = 2,\ldots,n_{kj}, j = 2,\ldots,m, k = 1,\ldots,m_{j-1} \tag{21}
$$

$$
b_{\sigma_{kj}(1),j} \ge s_{\sigma_{kj}\left(n_{kj}\right),\sigma_{kj}(1),kj};\ j = 1,\ldots,m, k = 1,\ldots,m_j \tag{22}
$$

$$
b_{\sigma_{km}(i),m} + \frac{T.d_{\sigma_{km}(i)}}{p_{\sigma_{km}(i),km}} \le T;\ i \in J_{km}, k = 1,\ldots,m_m \tag{23}
$$

$$
F.T = H \text{ and } F \ge 1 \tag{24}
$$

$$
T, b_{ij} \ge 0\quad \forall i,j. \tag{25}
$$

Figure 5. The NLP model for a given discrete solution.

First both selected parents are examined on a position-by-position basis and the similar building blocks of jobs are directly copied to the offspring. Then, two random cut points are drawn and the section between these points at each parent is directly copied to one of the offspring. Finally, the missing elements of each offspring are copied in the relative order of the other parent so as to maintain feasibility in the job permutation. Figure 6 represents a sample of this crossover operator.

3.6. Mutation Operator Mutation is a background operator that produces spontaneous random changes in the various chromosomes in order to implement diversification strategy. A simple way to achieve mutation is to alter one or more genes. There are several mutation operators for permutations such as swapping, inversion, insertion and shift mutation [15]. In our HGA, the swapping operator has been selected based on our initial tests as a mutation operator. Figure 7 represents a sample of this mutation operator.

3.7. Local Improvement Procedure Our local improvement procedure is based on an iterative neighborhood search (NS) so that within successive interchanges, a given offspring is replaced with an elite (dominating) neighbor.

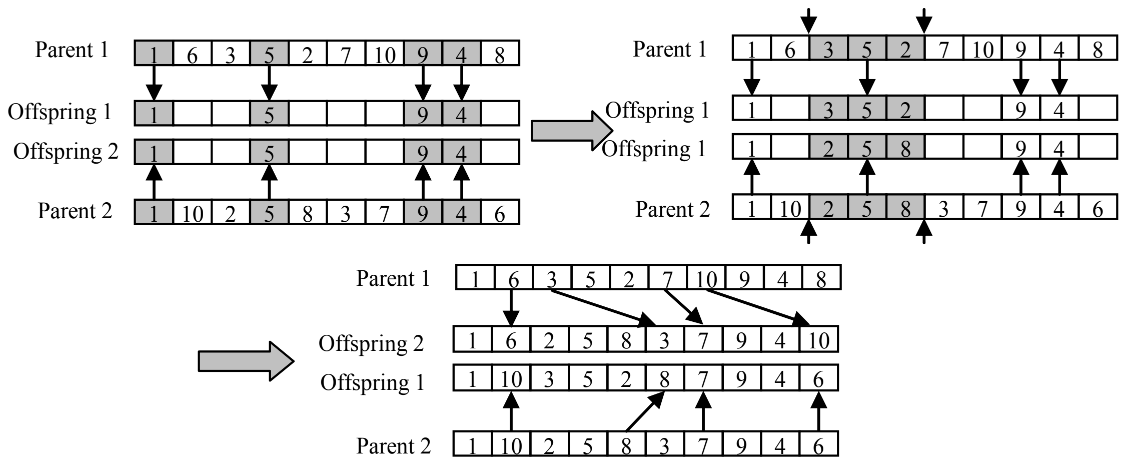


Figure 6. Illustration of SJ2OX operator

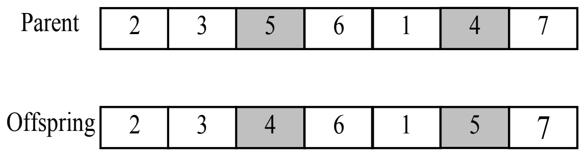


Figure 7. Illustration of swapping.

Among many definitions of NS [5,16], we have adopted the insertion neighborhood procedure. By this neighborhood structure, first two random positions are selected (say x < y), then the component x is removed and inserted at the initial position of y, and components between x and y are shifted by one unit to the left. Figure 8 shows an example of this NS.

3.8. Population Replacement Chromosomes for the next generation are selected from the enlarged population. After the generation of offspring via main GA operators (i.e., crossover and mutation) and improvement using the neighbourhood search procedure, the improved offspring are added to the current population. Then, n better chromosomes are chosen as the new population from the enlarged population, where n is the population size.

3.9. Termination Criteria The termination criterion determines when GA will stop. In our implementation we stop when pre-determined number of generations, max_gen, has been executed or when the algorithm has run for max_nonimprove generations without improvement.

4. PROPOSED SIMULATED ANNEALING ALGORITHM

SA is a random neighborhood search technique. A standard SA procedure starts by generating an initial solution. At each stage (temperature), the new solution taken from the neighborhood of the current solution is accepted as a new solution when it has a lower or equal cost; otherwise it is accepted with a probability depending on the difference between the current and new solution, and the current temperature. This temperature is reduced periodically with a temperature reduction scheme, so that it moves gradually from a relatively high value to near zero.

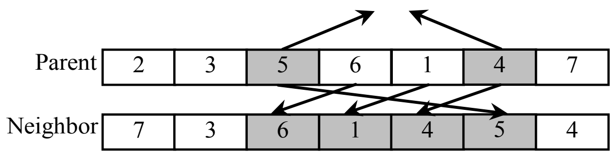


Figure 8. Illustration of SJ2OX operator.

4.1. Initial Solution Generation The quality of the initial solution has a major effect on the efficiency of a SA algorithm. Also, the solution space searched by HGA must be agreed with the solution space searched by SA until these two proposed meta-heuristics can fairly be compared with each other. Therefore, the initial solution of the SA is selected through the best solution generated by the approaches presented in Section 3.1.

4.2. Evaluation Function To determine the quality of each discrete solution, the NLP model presented in Figure 5 has been applied.

4.3. Neighborhood Structure Based on results of initial experiments, the insertion operator has been used in the implementation of the SA. An example of this operator has been shown in Figure 8.

4.4. Candidate List To improve the quality of the final solution, on the best out of these four candidate solutions has been selected: (1) the best solution in the annealing processes so far, S_G^*, (2) the best solution at the specific temperature T, S_T^*, (3) the current solution at the specific temperature T, S_T^0, (4) and a randomly generated solution at the specific temperature T, S_T^R.

4.5. Temperature Reduction Mechanism We have used linear cooling strategy: $T_{new} = T_{old} \times \alpha$, which α is the cooling rate.

4.6. Termination Criteria To increase search speed of the SA algorithm, we have considered three termination criteria. The first condition can terminate the algorithm, and the two other conditions can terminate the search process of the algorithm in a specific temperature.

Condition 1. If the temperature reaches a final predetermined temperature, the algorithm is terminated.

Condition 2. If the number of times that the best solution at a specific temperature, S_T^*, is replaced with the better solution, reaches a pre-set number, N_T, the search of the algorithm at that temperature is terminated.

Condition 3. If the number of iterations at a specific temperature exceeds a pre-set number, I_{max}, the search process at that temperature is terminated.

The steps of proposed SA algorithm are stated below:

Step 1. Initialization step

```
1.1.  set the parameters such as, T, $T_f$, cooling rate $\alpha$, $N_T$, $I_{max}$.
1.2.  obtain initial solution $S_T^0$ (from Section 4.3) and set $S_G^* = S_T^* = S_T^0$.
1.3.  set $num_I = 0$, $num_T = 0$.
```

Step 2. Iteration step

```
2.1.          while ($T > T_f$)
2.1.1.        while ($num_I < I_{max}$) and ($num_T < N_T$)
2.1.1.1.      select a neighbor solution, $S_T^1$ (Sections 4.3 and 4.4)
2.1.1.2.      compute $\Delta = f(S_T^1) - f(S_T^0)$.
2.1.1.3.      if $\Delta \le 0$, set $S_T^0 = S_T^1$.
2.1.1.3.1.    compute $\Delta = f(S_T^1) - f(S_T^*)$.
2.1.1.3.2.    if $\Delta \le 0$, set $S_T^* = S_T^1$ and $num_T = num_T + 1$.
2.1.1.3.2.1.  compute $\Delta = f(S_T^*) - f(S_G^*)$.
2.1.1.3.2.2.  if $\Delta \le 0$, set $S_G^* = S_T^*$.
2.1.1.4.      else if $\Delta > 0$, select a random variable $X \sim U(0,1)$;
               if $e^{-\Delta/T} > X$, set $S_T^0 = S_T^1$.
2.1.1.5.      set $num_I = num_I + 1$.
2.1.2.        set $T = T \times \alpha$ and $num_T = 0$, $num_I = 0$.
```

Step 3. Return the solution found for S_G^*.

5. COMPUTATIONAL EXPERIMENTS

To evaluate the efficiency of the proposed algorithms, in terms of the solution quality and the required computation time, some experiments have been conducted. All of the experimental tests have been implemented on a personal computer with an Intel Pentium IV 1800 MHz CPU and all of the proposed algorithms have been coded with MATLAB 6.5. Moreover, Lingo 8.0 has been used to solve the mixed zero-one non-linear models.

Parameters of the proposed HGA after initial tests have been adjusted as: population size: pop_size = n, maximum number of generations: max_gen = m × n, maximum number of generations without improvement: max_nonimprove = round (n/2), crossover probability: $P_c = 0.8$, mutation probability: $P_m = 0.2$, and the tournament selection parameter: $\varphi = 0.7$.

Moreover, parameters of the proposed SA after initial tests have been set as: initial temperature: T = 100, final temperature: $T_f = 1$, cooling rate: $\alpha = 0.9$, number of times that the current solution at a specific temperature is replaced with better solution: $N_T = 3$ or 5 depending on the size of the problem, and number of iterations at a specific temperature: $I_{max} = n$.

The parameters for each problem instance have been randomly generated from the following uniform distributions:

$d_i \sim U(100, 1000)$, $p_{ikj} \sim (5000, 15000)$, $s_{ikj} \sim (0.01, 0.25)$,

$h_{i1} \sim U(1, 10)$, $A \sim U(10000, 20000)$

Because processing at each stage has a value added on components; h_{ij} values should be non-decreasing with j. So, after random generation of h_{i1} for each component i from a uniform distribution between 1 and 10, other associated h_{ij} values are determined by adding a randomly generated number between [1,5] to $h_{i,j-1}$. Also there could be a correlation between sc_{iukj} and s_{iukj} values. Therefore, for each randomly generated s_{iukj}, its corresponding sc_{iukj} parameter has been computed using the following equation:

$$sc_{iukj} = 15000 \times s_{iukj} + 1000 \times U(0,1).$$

Moreover, the numbers of parallel machines at each stage are randomly set to 1 or 2. To evaluate the performance of the solutions obtained via proposed algorithms, we have compared the total cost obtained by HGA and SA algorithms for each problem instance, with an associated lower bound (LB) in medium and large-size problems, and with the solution obtained by LINGO for the small size problems, respectively. We have calculated an index $\lambda = (TC-LB)/LB$ where TC is the total cost of a problem instance obtained by each algorithm, and LB is the associated lower bound cost. This lower bound is obtained for each problem instance by eliminating the components assigning and sequencing constraints in problem P.

In our computational experiments, we have considered nine different problem sizes with 4, 5 and 10 components, and 2, 3, 5 and 10 production stages. For each problem size, 20 problem instances have been randomly generated. To identify the superiority among the proposed algorithms, we have divided our problem instances into two parts: problem instances with 4 and 5 components, 2 and 3 stages (small-sized problems), and with 5 and 10 components and 2, 5 and 10 stages (medium and large-sized problems). For the small size problem instances, the solutions of HGA and SA have been compared with the Lingo's solution. Moreover, for the medium and large size problem instances, the quality of solutions, CPU time and performance ratio λ obtained by the HGA and SA have been compared with each other. Table 1 shows the structure of the test problems. Table 2 represents the results for the small size problem instances, and Table 3 gives these results for medium and large size problem instances.

In summary, we have made the following observations from our numerical experiments:

1. The results shown in Table 2 indicate which for the small size problem instances, the solutions obtained by the proposed HGA and SA are 67 and 62 times better than the Lingo's solutions. Also, in average, the solutions quality obtained by the HGA and SA are 9.68 and 8.01 percent better than the solutions quality obtained by the Lingo, respectively. Moreover, the results shown in Table 2, indicate the superiority of proposed algorithms with respect to both CPU time

TABLE 1. Structure of the Test Problems.

Problem Set	Number of Components	Number of Stages	Number of Machines at Each Stage	Problem Size			
				Number of Integer Variables	Total Number of Variables	Number of Constraints	Number of Nonlinear Constraints
1	4	2	2-1	49	58	169	30
2	4	3	2-1-1	65	78	229	42
3	5	2	2-1	76	87	326	47
4	5	3	1-1-2	101	117	441	67
5	5	5	1-1-2-2-1	176	202	764	112
6	5	10	1-1-2-1-2-2-1-2-1-2	376	427	1618	227
7	10	2	2-1	301	322	2701	192
8	10	5	2-1-1-2-1	701	752	6329	472
9	10	10	2-2-1-1-2-1-2-1-1-2	1501	1602	13493	952

TABLE 2. Results of the Small Size Test Problems.

Problem Size (n×m)	The Number of Times Which the HGA's Solution was Better Than the Lingo's Solution	The Number of Times Which the SA's Solution is Better Than the Lingo's Solution	The Average Percentage of Decrease in the Total Cost of HGA's Solution Compared to Lingo's Solution (%)	The Average Percentage of Decrease in the Total Cost of SA's Solution Compared to Lingo's Solution (%)	Average CPU Time for Lingo (In Seconds)	Average CPU Time for HGA (In Seconds)	Average CPU Time for SA (In Seconds)
4 × 2	16	14	6.94	5.62	2348.75	19.9	33.92
4 × 3	17	16	10.42	8.6	4878.07	47.43	80.26
5 × 2	16	15	8.27	6.95	4952.13	50.86	87.59
5 × 3	18	17	13.12	10.9	8356.5	137.14	203.86

TABLE 3. Results of the Mdium and Large Size Test Problems.

Problem Size (n×m)	The Average Performance Ratio of the HGA (%)	The Average Performance Ratio of the SA (%)	The Average CPU Time of HGA (In Seconds)	The Average CPU Time of SA (In Seconds)	Comparison of the SA Versus HGA in Percent (%)
5 × 5	11.51	13.17	331.43	445.28	12.6
5 × 10	15.7	18.84	1364.33	1748.23	16.66
10 × 2	14.52	16.43	73.5	126.08	11.62
10 × 5	23.65	29.11	595.01	767.06	18.75
10 × 10	32.26	39.74	2384.99	4077.5	18.82

and the solution quality when compared to Lingo's solutions. It is noted that Problem P is a mixed integer non-linear program (MINLP). So, in finding an optimal solution by Lingo 8.0 solver, the solver traps in a local optimal solution in most of the time even in small-sized problems. On the other hand, because of good ability of the proposed HGA and SA algorithms in both local and global search, in the majority of test problems, the solutions obtained by heuristic methods are better than the corresponding solution found by Lingo 8.0.

2. For the medium and large size problem instances, performance ratio λ have been calculated and used as a measure to compare the proposed algorithms. In Table 3, we have compared the performance of the proposed algorithm with each other. The results indicate that the qualities of HGA's solutions in average are 15.69 % better than the quality of SA's solutions.
3. As it is mentioned before, the permutations in HGA are converted to complete discrete solutions via a proposed heuristic (see Section 3.1). This heuristic considers the setup costs, demand rates and the processing times when assigning and sequencing of components to machines. Computational results indicate which is an effective constructive heuristic in the context of the problem.

6. CONCLUSION REMARKS

In this paper, the common cycle approach has been used to solve the economic lot and delivery scheduling problem, in flexible flow lines with unrelated parallel machines, and sequence dependent setup times/costs over a finite horizon planning. First, a new mixed zero-one nonlinear model is developed to solve the problem to optimality. Providing an optimal solution is not a practical approach especially for medium and large size problems. Thus, two efficient meta-heuristic (HGA and SA) have been developed to obtain near optimal (or ideally optimal) solutions in a reasonable time.

In our HGA, the following approaches have been used for hybridization: incorporating simple and effective heuristics into initialization step of GA to generate a good initial population, developing a new simple heuristic to convert a permutation to a complete discrete solution, and incorporating an efficient neighborhood search method to the main GA loop of recombination and selection as a local optimizer.

Furthermore, in our SA, for improving the solution quality and efficiency of the algorithm, a specific candidate list for selecting a good neighbor has been developed. Also, to increase the searching speed, some additional termination criteria have been used.

Computational results indicate that due to nonlinearity nature of the original Problem P, the quality of solutions generated by the heuristic methods are usually better than Lingo's solutions even in small-sized problems. Moreover, when comparing two proposed HGA and SA algorithms, the HGA outperforms the SA algorithm with respect to both solution quality and computation time especially in large-size problem instances.

The problem considered here can be extended in different ways. Among them, we suggest the following ones as possible directions for further studies:

Considering the Problem P under multiple-cycle (i.e., basic period) cyclic approach and allowing the different items having different cycle times in order to find solutions with lower total cost.

Extending the current model to more complex supply chains, for example, a multiple supplier, multiple assembler case in which, each assembler could receive different items from different suppliers.

7. ACKNOWLEDGEMENT

This research has been supported by Tehran University which is acknowledged.

8. REFERENCES

1. Handfield, Jr. R. B. and Nichols, E. L., "Introduction to Supply Chain Management", Prentice-Hall, Upper Saddle River, New Jersey, U.S.A., (1999).
2. Eilon, S., "Scheduling for Batch Production", ***Institute of Production Engineering Journal***, Vol. 36, (1957), 549-579.
3. Torabi, S. A., Fatemi Ghomi, S. M. T. and Karimi, B., "A Hybrid Genetic Algorithm for the Finite Horizon Economic Lot and Delivery Scheduling in Supply Chains", ***European Journal of Operational Research***, Vol. 173, (2006), 173-189.
4. Ouenniche, J. and Boctor, F. F., "Sequencing, Lot Sizing and Scheduling of Several Components in Job Shops: The Common Cycle Approach", ***International Journal of Production Research***, Vol. 36, No. 4, (1998), 1125-1140.
5. Ouenniche, J., Boctor, F. F. and Martel, A., "The Impact of Sequencing Decisions on Multi-Item Lot Sizing and Scheduling in Flow Shops", ***International Journal of Production Research***, Vol. 37, No. 10, (1999), 2253-2270.
6. Hahm, J. and Yano, C. A., "The Economic Lot and Delivery Scheduling Problem: The Single Item Case", ***International Journal of Production Economics***, Vol. 28, No. 2, (1992), 235-252.
7. Hahm, J. and Yano, C. A., "The Economic Lot and Delivery Scheduling Problem: The Common Cycle Case", ***IIE Transactions***, Vol. 27, (1995a), 113-125.
8. Hahm, J. and Yano, C. A., "The Economic Lot and Delivery Scheduling Problem: Models for Nested Schedules", ***IIE Transactions***, Vol. 27, (1995b), 126-139.
9. Khouja, M., "The Economic Lot and Delivery-Scheduling Problem: Common Cycle, Rework, and Variable Production Rate", ***IIE Transactions***, Vol. 32, (2000), 715-725.
10. Jensen, M. T. and Khouja, M., "An Optimal Polynomial Time Algorithm for the Common Cycle Economic Lot and Delivery Scheduling Problem", ***European Journal of Operational Research***, Vol. 156, No. 2, (2004), 305-311.
11. Vergara, F. E., Khouja, M. and Michalewicz, Z., "An Evolutionary Algorithm for Optimizing Material Flow in Supply Chains", ***Computers and Industrial Engineering***, Vol. 43, (2002), 407-421.
12. Ouenniche, J. and Bertrand, J. W. M., "The Finite Horizon Economic Lot Scheduling Problem in Job Shops: The Multiple Cycle Approach", ***International Journal of Production Economics***, Vol. 74, (2001), 49-61.
13. Torabi, S. A., Karimi, B. and Fatemi Ghomi, S. M. T., "The Common Cycle Economic Lot Scheduling in Flexible Job Shops: The Finite Horizon Case", ***International Journal of Production Economics***, Vol. 97, (2005), 52-65.
14. Hsu, W., "On the General Feasibility Test for Scheduling Lot Sizes for Several Products on One Machine", ***Management Science***, Vol. 29, (1983), 93-105.
15. Cheng, R. and Gen, M., "Parallel Machine Scheduling Problems using Memetic Algorithms", ***Computers and Industrial Engineering***, Vol. 33, (1997), 761-764.
16. Taillard, E., "Some Efficient Heuristic Methods for the Flow Shop Sequencing Problem", ***European Journal of Operational Research***, Vol. 47, (1990), 65-79.
17. Wang, H. F. and Wu, K. Y., "Hybrid Genetic Algorithm for Optimization Problems with Permutation Property", ***Computers and Operations Research***, Vol. 31, No. 14, (2004), 2453-2471.
18. Brah, S. and Loo, L. L., "Heuristics for Scheduling in a Flow Shop with Multiple Processors", ***European Journal of Operational Research***, Vol. 113, (1999), 113-122.
19. Luu, D. T., Bohez, E. J. and Techanitisawad, A., "A Hybrid Genetic Algorithm for the Batch Sequencing Problem on Identical Parallel Machines", ***Production Planning and Control***, Vol. 13, No. 3, (2002), 243-252.
20. Ruiz, R., Maroto, C. and Alcaraz, J., "Solving the Flow Shop Scheduling Problem with Sequence Dependent Setup Times using Advanced Meta-Heuristics", ***European Journal of Operational Research***, Vol. 165, No. 1, (2005), 34-54.

21. Ruiz, R. and Maroto, C., “A Genetic Algorithm for Hybrid Flow Shops with Sequence Dependent Setup Times and Machine Eligibility”, ***European Journal of Operational Research***, Vol. 169, No. 3, (2006), 781-800.

22. Gen, M. and Cheng, R., “Genetic Algorithms and Engineering Design”, Wiley, New York, U.S.A., (1997).