A meta-heuristic approach for the ELDSP in flexible flow lines: the power-of-two policy

Seyed Ali Torabi*[1] and Masoud Jenabi[2]
[1]Assistant Professor, Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran
[2]Ph.D. Student, Department of Industrial Engineering, Amirkabir University of Technology, 424, Hafez Avenue, 15916-34311 Tehran, Iran
(Received 25.6.85, Accepted 27.8.88)

Abstract

In this paper, the problem of lot sizing, scheduling and delivery of several items in a two-stage supply chain over a finite planning horizon is studied. Single supplier via a flexible flow line production system (FFL) produces several items and delivers them directly to an assembly facility. Based on basic period (BP) strategy, a new mixed zero-one nonlinear programming model has been developed with the objective of minimization average setup, inventory-holding and delivery costs per unit time in the supply chain without any stock-out. The problem is very complex and it cannot be solved to optimality especially in real-sized problems. So, an efficient hybrid genetic algorithm (HGA) has been proposed based on applying the most applied BP approach i.e., power-of-two policy. Based on a number of problem instances, the solution quality of the algorithm has been evaluated and also compared with the common cycle approach. Numerical experiments demonstrate the superiority of the solutions of proposed HGA and indicate that is a very promising solution method for the problem.

Keyword: Flexible Flow Lines; Lot and Delivery-Scheduling, Basic Period Approach, Power-of-Two Policy, Hybrid Genetic Algorithm (HGA)

Introduction

Nowadays, there is a high tendency to develop integrated models in research community for simultaneously cost-effective planning of different activities in supply chains. Among them, integrated production and delivery planning between adjacent supply parties is in particular interest which can reduce the total logistics-related costs considerably.

Literature review reveals that one of the earliest studied problems in this area is the economic lot scheduling problem (ELSP). This problem addresses lot-scheduling of several items with static and deterministic demands over an infinite planning horizon at a single facility, where products are delivered to the customer continuously. Research on the ELSP usually has focused on cyclic schedules (i.e., schedules that are repeated periodically) with three well known policies: common cycle, basic period (or multiple cycle) and time varying lot size approaches (Torabi et al. 2005). Several authors have extended the ELSP to multistage production systems with common cycle production policy (Ouenniche and Boctor 1998, 1999, Fatemi Ghomi and Torabi 2002, Torabi et al. 2005).

The economic lot and delivery-scheduling problem (ELDSP) is an extension of ELSP to a two-stage supply chain where a supplier produces several items for an assembly facility and delivers them to it in a static condition. Hahm and Yano (1995a, b) provided an excellent review of models related to ELDSP and developed two efficient heuristic algorithms to solve it based on common cycle and nested schedule strategies, respectively. Jensen and Khouja (2004) developed an optimal polynomial time algorithm for the ELDSP under common cycle approach. Finally, Torabi et al. (2006) considered the ELDSP in flexible flow lines under common cycle approach and over a finite planning horizon. They developed an effective HGA to obtain near (or ideally optimal) solutions.

* Corresponding author: Tel& Fax: +98- 21- 88021067 Email: satorabi@ut.ac.ir

Regarding to the basic period (BP) approach, Bomberger (1996) assumed different production cycles for items in which each cycle time must be an integer multiple of a BP that is long enough to meet the demand of all items. The production frequency of each product during the global cycle is then determined as a multiple of the selected BP. In such a case, infeasibility results from the artificial restrictions imposed by the concept of BP. Elmaghraby (1978) provided a review of the various contributions to ELSP and presented an improvement upon the BP approach, named the extended basic period (EBP) method. Its main difference from Bomberger's BP method was that it allowed items to be loaded on two BPs simultaneously and at the same time relaxed the requirement that the basic period should be large enough to accommodate such simultaneous loading. Yao and Elmaghraby (2001) developed an evolutionary algorithm for ELSP under basic period policy. Also, Ouenniche and Boctor (2001a, b, c) proposed three efficient heuristic approaches, i.e., power of two, two group and G-group methods for the ELSP in flow shop systems over an infinite planning horizon under basic period approach.

In all above works, the planning horizon is assumed to be infinite. However, this assumption considerably reduces the usefulness of the proposed contributions, because in practice, planning horizon is often finite. In this regard, there are few research which have assumed the finite planning horizon (Ouenniche and Boctor 1998, Ouenniche and Bertrand 2001, Torabi et al. 2005, 2006).

Consequently, to the best of our knowledge, there is no research on ELDSP in flexible flow lines under basic period approach over a finite planning horizon so far. It is noted that the solutions obtained via the basic period approach are generally better than the common cycle's solutions, and this is our main motivation in this research.

The outline of the paper is as follows: problem definition and formulation has been presented in section 2. The proposed HGA have been explained in section 3. In section 4, an efficient procedure has been developed for determining upper bounds on k_i values. An efficient feasibility test for capacity checking along with an iterative repair procedure based on k_i values modifications to convert an infeasible solution to a feasible, are also proposed in section 5. Computational experiments are provided in section 6. Finally, section 7 is devoted to conclusion remarks.

Problem formulation

This paper studies the finite horizon economic lot and delivery scheduling problem in flexible flow lines under basic period approach. At first, a new mixed zero-one nonlinear program has been developed whose optimal solution determines simultaneously the optimal assignment of products in basic periods, optimal assignment of products to machines at stages with multiple parallel machines, the optimal products sequence for each machine at each stage, the optimal lot sizes and the optimal production and delivery schedule at each global cycle.

To solve the problem, we assume that the cycle time of product i, T_i, is an integer multiple, say k_i, of a basic period F; i.e., $T_i=k_i.F$ for all i. In addition, we required that the basic period F to be such that the planning horizon PH is an integer multiple of a global cycle $H.F$; that is $PH=r.H.F$ where r is an integer and H denotes the least common multiple (LCM) of the k_i's. Consequently, to solve the problem, a hybrid genetic algorithm (named *PT-HGA*) has been proposed based on power of two policy (the most applied BP strategy).

The following assumptions are considered for the problem formulation:

- Machines at stages with multiple parallel machines are identical (in all characteristics such as production rates and setup times/cost)
- Machines of different stages are continuously available and each machine can only process one product at a time.

- At stages with parallel machines, each product is processed entirely on one machine.
- Setup times/costs in supplier's production system are sequence independent.
- Production sequence at each basic period for each machine at each stage is unique and is determined by the solution method.
- The supplier incurs linear inventory holding costs on semi-finished products.
- Both the supplier and the assembler incur linear holding costs on end products.
- Preemption/Lot-splitting is not allowed.

Moreover, the notations used for the problem formulation are defined as follows:

Indices

Index	For	Scale
i	Products	{1, 2, …, n}
j	Work Centers	{1, 2, …, m}
l	Positions	{1, 2, …, n}
k	Basic Periods	{1, 2, …, H[1]}
k'	Machines	{1, 2, …, m_j}

Parameters:

n	number of products
m	number of work centers (stages)
m_j	number of parallel machines at stage j
$M_{k'j}$	k'-*th* machine at stage j
d_i	demand rate of product i
p_{ij}	production rate of product i at stage j
s_{ij}	setup time of product i at stage j
sc_{ij}	setup cost of product i at stage j
h_{ij}	inventory holding cost per unit of product i per unit time between stages j and $j+1$
h_i	inventory holding cost per unit of final product i per unit time
A	transportation cost per delivery
PH	planning horizon length
M	a large real number

Decision variables

σ_k	partial sequence vector of basic period k
$\sigma_{kk'j}$	sequence vector of machine k' at stage j in basic period k
r	number of production cycles over the finite planning horizon
$n_{kk'j}$	number of products assigned to machine k' at stage j in basic period k
F	basic period length
b_{ij}	production beginning time of product i at stage j (after related setup operation)
k_i	time multiple of product i
$x_{ilk'kj}$	1, if product i occupies position l on machine k' at stage j in basic period k; 0, otherwise

It is noted that based on above variables, the global cycle length is equal to the least common multiple of the k_i variables, in other words we would have: H = LCM ($k_1, k_2,...,k_n$). Also, the production cycle length, the production lot size of product i and the processing time for a lot of product i at stage j would be as follows:

$T_i = k_i.F,\ Q_i = d_i.T_i,\ pt_{ij}= Q_i /p_{ij} =d_i.k_i.F / p_{ij}$.

Moreover, at stages with only one machine the value of m_j and index k' would be only one. Since after processing each product at each stage, there would be a value added for the product, values of h_{ij} parameters will be non-decreasing; that is $h_{i,j-1}\leq h_{ij}$.

The objective function of this problem (Problem P) includes two fundamental expressions. First expression is related to the setup and transportation costs. This expression consists of two parts: the first part computes the setup cost of products with respect to their production cycle times. The second part computes the transportation cost of products on each basic period.

$$C=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_iF}+\frac{A}{F}. \tag{1}$$

The inventory holding costs are often more complicated which are incurred at both the supplier and the assembler. Figure 1 shows the inventory level of final product i in one cycle at the assembly facility is $1/T_i\left(d_iT_i\frac{T_i}{2}\right)=d_iT_i/2$. Therefore, the average inventory of component i per unit time at the assembly facility will be:

$$k_iF/2\sum_{i=1}^{n}h_id_i. \tag{2}$$

[1] H= LCM ($k_1, k_2,...,k_n$)

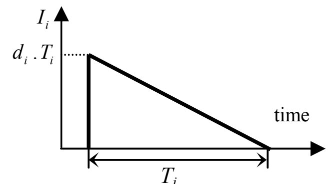

Fig. 1: Inventory level at the assembler in one cycle.

Two types of inventories i.e., *WIP* and finished product inventories are considered for the supplier. Figures 2 and 3 show the evolution of *WIP* inventory of product *i* between two successive stages *j-1* and *j*, and the inventory level of final product *i*, respectively.

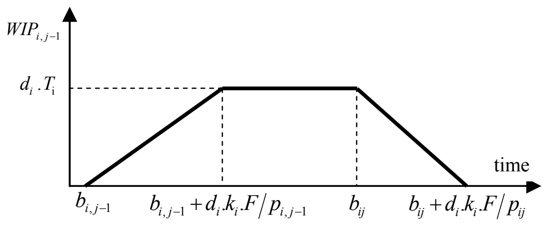

Fig. 2. WIP between stages *j-1* and *j* at the supplier.

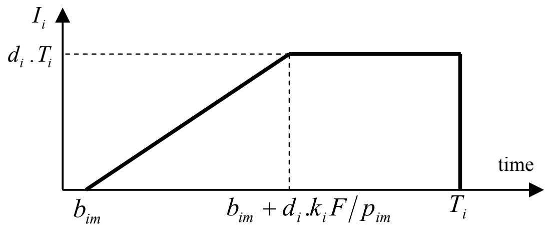

Fig. 3. Final product inventory at the supplier.

From Figure 2 it is obvious that the average WIP inventory of product *i* between two successive stages *j-1* and *j* per unit time is:

$$I_{i,j-1} = \frac{1}{T_i}\left\{\frac{d_iT_i}{2}\frac{d_iT_i}{p_{i,j-1}} + d_iT_i\left(b_{ij} - b_{i,j-1} - \frac{d_iT_i}{p_{i,j-1}}\right) + \frac{d_iT_i}{2}\frac{d_iT_i}{p_{ij}}\right\}$$

$$= d_i\left(b_{ij} + \frac{d_ik_iF}{2p_{ij}} - b_{i,j-1} - \frac{d_ik_iF}{2p_{i,j-1}}\right) \qquad (3)$$

Therefore, the total *WIP* inventory holding cost for all products per unit time at the supplier would be as follows:

$$TC_{WIP} = \sum_{i=1}^{n}\sum_{j=2}^{m} h_{i,j-1} d_i \left\{b_{ij} + \frac{d_ik_iF}{2p_{ij}} - b_{i,j-1} - \frac{d_ik_iF}{2p_{i,j-1}}\right\}.$$

(4)

Also, from Figure 3, we can derive the average inventory of final product *i* per unit time:

$$I_{i,j-1} = \frac{1}{T}\left\{\frac{d_iT}{2}.\frac{d_iT_i}{p_{im}} + d_iT_i\left(T_i - b_{im} - \frac{d_iT_i}{p_{im}}\right)\right\}$$

$$= d_i\left(1 - .\frac{d_i}{2p_{im}}\right)k_iF - d_ib_{im} \qquad (5)$$

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

$$TC_{FI} = \sum_{i=1}^{n} h_i d_i \left(1 - \frac{d_i}{2.p_{ikm}}\right).k_1F - \sum_{i=1}^{n} h_i d_i .b_{im} \qquad (6)$$

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

$$TC = \frac{A}{F} + \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_iF} +$$

$$\sum_{i=1}^{n} F.\left[\frac{h_i.d_i.k_i}{2}.\left(3 - \frac{d_i}{p_{im}}\right) + \frac{d_i^2}{2}.\sum_{j=2}^{m} h_{i,j-1}\left(\frac{1}{p_{ij}} - \frac{1}{p_{i,j-1}}\right)\right]$$

$$+ \sum_{i=1}^{n}\sum_{j=2}^{m} h_{i,j-1}.d_i\left(b_{ij} - b_{i,j-1}\right) - \sum_{i=1}^{n} h_i d_i b_{im} \qquad (7)$$

Regarding to this objective function and logical relationships between variables of Problem P (that some of them are recognizable from inventory level curves), a mixed zero-one nonlinear model is developed to obtain optimal solution of the problem.

Problem P:

$$Min\ Z = \frac{A}{F} + \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_iF} +$$

$$\sum_{i=1}^{n} F\left[h_i \frac{d_i.k_i}{2}\left(3 - \frac{d_i}{p_{im}}\right) + \frac{k_id_i^2}{2}\sum_{j=2}^{m} h_{i,j-1}\left(\frac{1}{p_{ij}} - \frac{1}{p_{i,j-1}}\right.\right.$$

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

(8)

subject to

$$b_{i,j-1} + \frac{d_ik_iF}{p_{i,j-1}} \le b_{ij} \ ; i = 1,...,n; \ j = 2,...,m \qquad (9)$$

$$b_{ij}+\frac{d_i k_i F}{p_{ij}}+s_{ij}-b_{uj}\le M\left(2-x_{i\ell k'kj}-x_{u(\ell+1)k'kj}\right)$$
$$i=1,\dots,n,u\ne i;\, j=1,\dots,m;\, k'=1,\dots,m_j;\, \ell<n;\quad (10)$$
$$k=1,\dots,H,H=lcm(k_1,\dots,k_n)$$

$$\sum_{i=1}^{n}x_{i\ell k'kj}\le 1\ ;\ j=1,\dots,m;\, k'=1,\dots,m_j;\quad (11)$$
$$\ell=1,\dots,n;\, k=1,\dots,H,H=lcm(k_1,\dots,k_n)$$

$$\sum_{i=1}^{n}x_{i(\ell+1)k'kj}\le\sum_{u=1}^{n}x_{u\ell k'kj};\ i=1,\dots,n;\, j=1,\dots,m;\quad (12)$$
$$k=1,\dots,m_j;\ \ell<n;\, k=1,\dots,H,H=lcm(k_1,\dots,k_n)$$

$$\sum_{k'=1}^{m_j}\sum_{\ell=1}^{n}\sum_{k=1}^{k_i}x_{i\ell k'kj}=1\ \ ;i=1,\dots,n;\, j=1,\dots,m\quad (13)$$

$$\sum_{k'=1}^{m_j}\sum_{\ell=1}^{n}x_{i\ell k'(t+bk_i)j}=\sum_{k'=1}^{m_j}\sum_{\ell=1}^{n}x_{i\ell k'(t+(b+1)k_i)j}\ ;\quad (14)$$
$$i=1,\dots,n;\, j=1,\dots,m;\, t=1,\dots,k_i;\, b=0,\dots,\frac{H}{k_i}-2$$

$$\sum_{k'=1}^{m_j}\sum_{\ell=1}^{n}x_{ilk'kj}=\sum_{k'=1}^{m_j}\sum_{\ell=1}^{n}x_{ilk'k,j+1}\ ;\quad (15)$$
$$i=1,\dots,m;\, j=1,\dots,m,j<m;\, k=1,\dots,H$$

$$b_{ij}\ge s_{ij}-M\left(1-\sum_{k'=1}^{m_j}x_{i1k'kj}\right);\, j=1,\dots,m,\quad (16)$$
$$i=1,\dots,n;\, k=1,\dots,H,H=lcm(k_1,\dots,k_n)$$

$$b_{im}+\frac{d_i k_i F}{p_{im}}\le F\,;i=1,\dots,n\quad (17)$$

$$H.F.r=PH;\, H=lcm(k_1,\dots,k_n)\quad (18)$$

$$r\ge 1,\ \textit{and intege}\quad (19)$$

$$F\ge 0;\ b_{ij}\ge 0\ \forall i,j;\ x_{i\ell k'kj}=\{0,1\};\quad (20)$$
$$\forall i,\ell,k',j,k$$

Problem P has the following set of constraints. Constraints (9) state that no product can be processed before it is completed at previous stage. Constraints (10) show that, no product can be processed before the completion of its predecessor in the related production sequence ($\sigma_{kk'j}$). Constraints (11) reveal that at each position of each machine, there is at most one product; because for each machine such as $M_{k'j}$, it may be assigned less than n products. Constraints (12) state that one product can be assigned at one position of machine $M_{k'j}$; if another product is to be assigned at the previous position of this machine. Constraints (13) ensure assignment of product i to one of the first k_i basic period and implies that each assigned product at each stage has a unique position in the sequence of one machine. Constraints (14) determine assignment of products in appropriate basic periods during the H basic periods. Constraints (15) denote that if product i have been assigned to basic period k at stage j, it must be assigned to that basic period at all stages. Constraints (16) show that if product i is the first product in the sequence vector of one machine at stage j, it's processing cannot be started before setting up the corresponding machine. Constraints (17) assure that the resulting schedule is cyclic so that the process completion time for each product at final stage is less than or equal to a basic cycle time F. Constraint (18) implies that the planning horizon PH is an integer multiple of $H.F$, where H=LCM $(k_1,\dots,k_n)$, and F is the basic period length. Constraints (19) show that r is an integer greater than or equal to one. Finally, Constraints (20) are the non-negativity constraints of variables.

It is noted that this model can be run for a set of known k_i variables. In other words, to run this model, at first the k_i values must be determined, and then the corresponding optimal basic period length, optimal assignments, sequence vectors and production and delivery schedule of products are obtained via solving Problem P.

Proposed hybrid genetic algorithm

During the last thirty years, there has been a growing interest in obtaining the optimal solutions for complex systems using genetic algorithms (GA). Genetic algorithms maintain a population of potential solutions and simulate evolution process using selection process based on fitness of chromosomes and genetic operators. To improve solution quality and to escape from converging to local optima, various strategies of hybridization have been suggested (Cheng and Gen 1997, Torabi et al. 2006). In designing a hybrid genetic algorithm (HGA), the neighborhood search (NS) heuristic usually acts as a local improver into a basic GA loop.

In our HGA, each solution is characterized with a set of k_i multipliers and the value of basic period F. Beside the cost minimization; we have to generate feasible schedules. Therefore, a capacity feasibility test has been developed in section 5 which is able to identify the infeasible solutions and converting them to feasible schedules.

Chromosome representation

The proposed HGA search in the solution space of k_i values, so that each chromosome is a binary (zero-one) string, and each k_i multiplier will be represented as a particular part of a chromosome. For instance, the first u_1 bits are used to encode the value of k_1 and the particular piece of chromosome from the (u_1+1)-*th* bit to the (u_1+u_2)-*th* bit represents the value of k_2 and so on. In order to represent all the possible values of k_i for each item i, we need an upper bound (see section 4) on the value of k_i (or v_i so that $k_i = 2^{v_i}$). Because of encoding the value of k_i into a binary string, we have to establish a mapping between each binary string and an integer k_i. In fact, we map a binary string consisting of u_i bits to an integer value k_i by using the following equations for the power of two and non-power of two cases, respectively:

$$\left\langle b_{u_i} b_{u_i-1} .. b_1 \right\rangle_2 = \left(\sum_{j=1}^{u_i} b_j 2^{j-1} \right)_{10} = (v_i)_{10} \Rightarrow k_i = 2^{v_i} \quad (21)$$

Determining the σ_k vectors

In assigning and sequencing of products in different basic periods i.e., determination of σ_k vectors, it is not easy to derive a simple necessary and sufficient condition to have a non-empty set of feasible solutions. Given a vector of multipliers k_i; $i=1,...,n$, the procedure starts to make a vector say V' by sorting the products in ascending order of k_i and, within the products having the same multiplier k_i , in the descending order of ρ_i where:

$$\rho_i = \sum_{j=1}^{m} \frac{k_i d_i}{p_{ij}}, \; i = 1,...,n. \quad (22)$$

Each product i in the vector V' is assigned to the basic period t within the first k_i periods of global cycle H which minimizes:

$$\underset{k=t,t+m_i,...}{\text{maximum}} \left\{ \underset{j=1,...,m}{\text{maximum}} \left(\sum_{u\in\sigma_k} \frac{k_u d_u}{p_{uj}} + \frac{k_i d_i}{p_{ij}} \right) \right\} \quad (23)$$

Finally, for each k, k=1,…,H, we determine the sequence of products within σ_k such that if $i, u \in \sigma_k$ and i is ordered before u in V', then i also is ordered before u in σ_k.

Determining the $\sigma_{kk'j}$ vectors

First available machine (FAM) rule has been employed to assign and sequence the products of each basic period to machines of different stages (Torabi et al. 2006). According to this procedure, for any given permutation vector V; the products are assigned to machines of first stage by using FAM rule (if $m_1 > 1$). Then, for each of subsequent stages, the products have been first sequenced according to increasing order of their process completion time at the previous stage, and then assigned to the machines at the current stage according to FAM rule.

Initial population

Initial population of binary chromosomes is generated randomly. According to feasibility test, each infeasible solution is converted to a feasible one and then is inserted into the initial population.

Evaluation function

Each chromosome in the population represents a potential solution to the problem. Evaluation function is responsible for rating these potential solutions by assigning a real number as a measure of their fitness. In our problem after determining the $\sigma_{kk'j}$ vectors for each chromosome, evaluation function is obtained by solving the following NLP model (Problem P1). This problem is derived from Problem P by substituting $x_{ilkk'j}$ values by corresponding ones. Also, $\sigma_{kk'j}(i)$ indicates the i-th product in the sequence vector of machine $M_{k'j}$ in basic period k. Problem P1 can be solved by the following iterative procedure:

Initial step: Let r =1, and solve the resultant linear program.

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

Problem P1:

$$Min\ Z = \frac{A}{F} + \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_i F} + \sum_{i=1}^{n}\left[h_i.k_i.\frac{d_i}{2}\left(3-\frac{d_i}{p_{im}}\right)+\sum_{j=2}^{m}h_{i,j-1}.k_i.\frac{d_i^2}{2}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right]F + \sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}.d_i\left(b_{ij}-b_{i,j-1}\right)-\sum_{i=1}^{n}h_i.d_i.b_{im} \tag{24}$$

subject to:

$$b_{i,j-1}+\frac{k_i F.d_i}{p_{i,j-1}} \le b_{ij};\ i=1,\dots,n,\ j=2,\dots,m \tag{25}$$

$$b_{\sigma_{kk'j}(i-1),j}+\frac{k_{\sigma_{kk'j}(i-1)}F.d_{\sigma_{kk'j}(i-1)}}{p_{\sigma_{kk'j}(i-1),j}} - s_{\sigma_{kk'j}(i),j} \le b_{\sigma_{kk'j}(i),j};$$
$$i=2,\dots,n_{kk'j};\ j=1,\dots,m;\ k'=1,\dots,m_j;$$
$$k=1,\dots,H, H=lcm\,(k_1,\dots,k_n) \tag{26}$$

$$b_{\sigma_{kk'j}(1),j} \ge s_{\sigma_{kk'j}(1),j};\ j=1,\dots,m;$$
$$k=1,\dots,m_j;\ k=1,\dots,H,H=lcm(k_1,\dots,k_n) \tag{27}$$

$$b_{im}+\frac{k_i F.d_i}{p_{im}} \le F;\ i=1,\dots,n \tag{28}$$

$$r.H.F=PH;\ \ r\ge 1 \text{and intege} \tag{29}$$

$$F, b_{ij} \ge 0\ \ \forall i,j. \tag{30}$$

Selection and crossover operators

In proposed HGAs, we have used the tournament selection approach. It randomly chooses two chromosomes from parent pool,

and then chooses the fittest one if a random value generated (τ) is smaller than a pre-set probability value φ ($0.5<\varphi<1$). Otherwise, the other one is chosen. Then, the selected chromosomes are duplicated and pairs of them are selected as parents to undergo the crossover operation.

The main purpose of crossover is to exchange genetic materials between randomly selected parents with the aim of producing better offspring. In this research we have used the classic two point crossover. According to this crossover, at first two positions are randomly selected, and then the genes between them in the parent chromosomes are exchanged (see Figure 4).

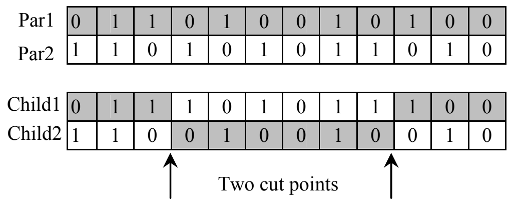
Par1 0 1 1 0 1 0 0 1 0 1 0 0
Par2 1 1 0 1 0 1 0 1 1 0 1 0
Child1 0 1 1 1 0 1 0 1 1 1 0 0
Child2 1 1 0 0 1 0 0 1 0 0 1 0
Two cut points

Fig. 4 Two point crossover

Mutation operator

Mutation introduces random variation (diversification) into the population. Most genetic algorithms incorporate mutation operator mainly to avoid convergence to local optima in the population and recovering lost genetic materials. In the proposed HGAs, we have used the swap mutation. Fig. 5 illustrates an example of this operator.

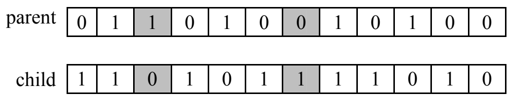

Fig. 5. Swap mutation.

Local improver

Our local improvement procedure is based on an iterative neighborhood search (NS) so that within successive interchanges, given offspring is replaced with an elite (dominating) neighbor. We have used inversion operator as local improver. Figure 6 is an example of this operator.

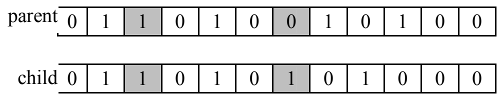

Fig. 6. Inversion operator.

Population replacement

Chromosomes for the next generation are

selected from the enlarged population. After offspring were generated from GA operators (crossover and mutation) and then improved by the neighborhood search procedure, the improved offspring are added to the current population. This population is called the enlarged population, and then about 60 percent of the new population is filled out by the best fittest chromosomes of the enlarged population. Remaining chromosomes are selected randomly from the remainder chromosomes in the enlarged population.

Termination criteria

Termination criterion determines when GA will stop. In other words, the genetic operations are repeated until one of the termination conditions is met. In our implementation, we stop HGAs, if pre-determined number of generations, *max_gen*, has been executed or the pre-set number of generations without any improvement in the last best solution*, max_nonimprove*, reaches.

Determining upper bounds on k_i values

In order to represent all possible and feasible values of each k_i multiplier, we determine an upper bound for each one. In our HGAs, we derive an upper bound on k_i through determining an upper bound on the objective value of each product *i*. Following steps describe how we can compute an upper bound for each k_i (k_i^{UB}):

Step1: For each product *i*, assume $k_i=1$ and calculate $\sum_j d_i/p_{ij}$, arrange the products in ascending order of these values. Assign the products and sequence them to machines at all stages via FAM rule. Finally, find the corresponding common cycle solution and its objective function, TC_{cc}.

Step2: Calculate the cost share of each product *i*, $TC^i_{cc} = TC_{cc} - \sum_{u=1,u\neq i}^{n} TC^u_{cc}$. As it had been mentioned by Yao and Elmaghraby (2001), we would have: $TC^i_{BP} \leq TC^i_{cc}$, where TC^i_{BP} is cost share of product *i* under basic period approach. So, TC^i_{BP} values must be determined.

Step3: Assume there is only one product (say product *i*) with following objective function:

$$TC^i_{BP} = \sum_{j=1}^{m} \frac{sc_{ij}}{k_i F} +$$
$$F\left[\frac{h_i.d_i.k_i}{2}.\left(3-\frac{d_i}{p_{im}}\right)+\frac{k_i d_i^2}{2}.\sum_{j=2}^{m} h_{i,j-1}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right.\right.$$
$$+\sum_{j=2}^{m} h_{i,j-1}.d_i\left(b_{ij}-b_{i,j-1}\right)-h_i d_i b_{im}. \tag{31}$$

Obviously, to obtain the optimal solution, we have to determine the starting times so that minimize $(b_{ij}-b_{i,j-1})-b_{im}$. Also, the smallest feasible value of $b_{ij}-b_{i,j-1}$ is equal to $k_iF.d_i/p_{i,j-1}$. Therefore, the best value of TC^i_{BP} is equal to:

$$TC^i_{BP} = \sum_{j=1}^{m} \frac{sc_{ij}}{k_i F} + k_i F.$$
$$\underbrace{\left[\frac{h_i.d_i}{2}.\left(3-\frac{d_i}{p_{im}}\right)+\frac{d_i^2}{2}.\sum_{j=2}^{m} h_{i,j-1}\left(\frac{1}{p_{ij}}+\frac{1}{p_{i,j-1}}\right)-h_i d_i^2 \sum_{j=1}^{m-1}\frac{1}{p_{ij}}\right]}_{H_i}. \tag{32}$$

Finally for a given value of *F*, we can derive an upper bound on v_i, ($k_i = 2^{v_i}$), denoted by v_i^{UB} by using following equations:

$$TC^i_{BP} \leq TC^i_{cc} \Rightarrow \sum_{j=1}^{m} \frac{sc_{ij}}{k_i F} + k_i F.H_i \leq TC^i_{cc}$$
$$\Rightarrow k_i^2 F^2 H_i - k_i F.TC^i_{cc} + \sum_{j=1}^{m} sc_{ij} \leq 0$$
$$\Rightarrow v_i^{UB} = log_2\left[\frac{TC^i_{cc} + \sqrt{\left(TC^i_{cc}\right)^2 - 4H_i\left(\sum_{j=1}^{m} sc_{ij}\right)}}{2H_i.F_{min}}\right]. \tag{33}$$

Moreover, to determine the minimum value of *F*, F_{min}, assume that *F* must be large enough so that at least one product with $k_i=1$, can be produced during it. Consequently, F_{min} is obtained from the following equation:

$$\underset{i=1,\ldots,n}{maximum}\left\{\sum_{j=1}^{m} s_{ij} + \sum_{j=1}^{m}\frac{d_i F}{p_{ij}}\right\} \leq F$$
$$\Rightarrow F \geq \underset{i=1,\ldots,n}{maximum}\left\{\frac{\sum_{j=1}^{m} s_{ij}}{1-\sum_{j=1}^{m} d_i/p_{ij}}\right\}. \tag{34}$$

Feasibility test and repair procedure

For a given chromosome and related k_i, σ_k and $\sigma_{kk'j}$ vectors, a simple test for capacity feasibility can be carried out. To do so, the process completion times of products for all *H* basic periods are first calculated. We can use the following procedure for doing so.

Then, if at least at one basic period, the related completion time would be greater than or equal to 1, the corresponding chromosome is infeasible and otherwise, it is feasible. In other words, if at least one of the ft_k values, $k=1,\ldots, H$, be grater than or equal to 1, this solution would be infeasible.

```
for $k=1,\ldots,H$
    for each $i \in \sigma_k$
        for $j=1,\ldots,m$
            $process\ 1 = \sum_{u\in\sigma_{kk'j}} k_u d_u / p_{uj}$;
                    product u is before i at basic
                    period k on machine $M_{k'j}$,
            $process\ 2 = \sum_{u\in\sigma_{kk',j-1}} \frac{k_u d_u}{p_{u,j-1}} + \frac{k_i d_i}{p_{i,j-1}}$;
                    product u is before i at basic
                    period k on machine $M_{k',j-1}$
            $fin=\max\{process1,process2\}+k_i d_i/p_{ij}$
        end
      $ft_{ik}=fin$
    end
$ft_k=\max_i\{ft_{ik}\}$
end
```

For converting an infeasible solution to a feasible one, we have also proposed following iterative repair procedure based on k_i values modifications.

Step1: Choose the basic period with maximal value of ft_k, e.g. basic period *k1*.

Step2: Among the products of basic period *k1*, select the product with maximum amount of process time ($\max_i \{\sum_{j=1,\ldots,m} k_i d_i / p_{ij}\}$; $i \in \sigma_{k1}$), e.g. product *i*.

Step3: If $v_i \neq 0$; set $v_i=v_i-1$, and obtain σ_k and $\sigma_{kk'j}$ vectors for this new set of multiples. If this solution is feasible, stop, otherwise go to *step1*. If $v_i=o$; select the product with subsequent maximum amount of process time and go to *step3*.

It is noteworthy that each chromosome obtained via genetic operators (crossover, mutation and local improver) is checked with aforementioned feasibility test.

Computational experiments

To verify efficiency of the proposed algorithm, in terms of the solution quality and the required computation time, several numerical experiments have been conducted. In this regard, all of the experimental tests have been implemented on a personal computer with an Intel Pentium IV 1800 MHz CPU and HGA has been coded with MATLAB 6.5. Moreover, to solve the mixed zero-one non-linear models, LINGO 6.0 optimization software has been used.

Parameter setting and data set

The tuned parameters of the HGA after initial tests have been adjusted as: population size *pop_size* = *n*, maximum number of generations *max_gen* = $m \times n$, maximum number of generations without improvement *max_nonimprove* =*n*, crossover probability P_c= 0.8, mutation probability P_m= 0.2, and tournament selection parameter φ = 0.7.

Furthermore, the parameters of each problem instance have been randomly generated from the following uniform distributions:

$$d_i \sim U(100,1000),\quad p_{ij} \sim U(5000,15000),$$
$$s_{ij} \sim U(0.01,0.025),\quad h_{i1} \sim U(1,10),$$
$$A \sim U(10000,20000).$$

Because processing at each stage has a value added on products, h_{ij} values should be non-decreasing with *j*. So, after random generation of h_{i1} for each product *i*, other associated h_{ij} values are determined by randomly generating incremental additions i.e., $h_{ij}=h_{i,j-1}+U(1,3)$. Also there could be a correlation between sc_{ij} and s_{ij} values. Therefore, for each randomly generated s_{ij}, its corresponding sc_{ij} parameter has been computed using the following equation: $sc_{ij}=15000\times s_{ij}+1000\times U(0,1)$.

Performance evaluation

To verify efficiency of proposed solution method, we have considered eight different problem sizes. For each problem size, 20 problem instances have been randomly generated. Also, we have divided our problem instances in two parts: problem instances with 4 and 5 products, and 2 and 3 stages (as the small-sized problems), and problem instances with 5 and 10 products and 5 and 10 stages (as the medium and large-sized problems). For small size problems, the solutions of HGA have been compared with the solution of LINGO software. Also, for medium and large-sized problems, we have calculated an index λ by the λ= (TC-LB)/LB equation and used it as a base of comparisons. In index λ, the TC is the total cost of a problem instance obtained by each algorithm and LB is the associated lower bound.

To calculate the LB, we must obtain minimum value of the following equation:

$$Z = \frac{A}{F} + \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_i F} + \sum_{i=1}^{n}\left[h_i.k_i.\frac{d_i}{2}\left(3-\frac{d_i}{p_{im}}\right)+\sum_{j=2}^{m}h_{i,j-1}.k_i.\frac{d_i^2}{2}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right]F + \sum_{i=1}^{n}\sum_{j=2}^{m}h_{i,j-1}.d_i\left(b_{ij}-b_{i,j-1}\right)-\sum_{i=1}^{n}h_i.d_i.b_{im} \quad (35)$$

If we assign to b_{ij} the values that minimize (b_{ij}-$b_{i,j-1}$), then the above equation is minimized. According to constraints 2, the minimum possible amounts of (b_{ij}-$b_{i,j-1}$) are $d_i.k_iF/p_{i,j-1}$. Also, if products can be scheduled as soon as possible, we can substitute b_{im} with $\sum_{j=1,\dots,m-1} d_i.k_iF/p_{ij}$. Therefore, a good lower bound can de computed as follows:

$$F = \sqrt{\frac{A+\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_i}}{\sum_{i=1}^{n}\left[\begin{array}{l} h_i.k_i.\frac{d_i}{2}\left(3-\frac{d_i}{p_{im}}\right)+\sum_{j=2}^{m}h_{i,j-1}.k_i.\frac{d_i^2}{2}\left(\frac{1}{p_{ij}}+\frac{1}{p_{i,j-1}}\right) \\ -h_i d_i^2 k_i \sum_{j=1}^{m-1}\frac{1}{p_{ij}} \end{array}\right]}} \quad (36)$$

$$LB = \frac{A}{F} + \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_i F} + \sum_{i=1}^{n}\left[\begin{array}{l} h_i.k_i.\frac{d_i}{2}\left(3-\frac{d_i}{p_{im}}\right)+\sum_{j=2}^{m}h_{i,j-1}.k_i.\frac{d_i^2}{2}\left(\frac{1}{p_{ij}}+\frac{1}{p_{i,j-1}}\right) \\ -h_i d_i^2 k_i \sum_{j=1}^{m-1}\frac{1}{p_{ij}} \end{array}\right]F \quad (37)$$

Table 1 represents the computational results for the small sized problem instances and Table 2 and 3 gives these results for medium and large sized problem instances.

Table 1. Results of small-sized test problems

problem size (n×m)	*	**	average CPU time for LINGO (in seconds)	average CPU time for PTHGA (in seconds)
4×2	17	3.44	2736.98	34.43
4×3	16	5.05	5684.45	53.47
5×2	16	5.35	5770.77	54.69
5×3	18	9.07	9737.29	133.65

*: number of times that the PTHGA's solution was better than the LINGO's solution
**: the average of percentage decrease in PTHGA's solution compared to LINGO's solution (%)

In summary, we find out the following results from our numerical experiments:

1) In Table 1, solution qualities of the proposed algorithm have been compared with the related optimal solution obtained via LINGO 6.0. As we can see, for the small size problem instances, the solutions of the *PT-HGA* are 67 times better than the solutions of LINGO. It seems that having a mixed zero-one and nonlinear nature of the proposed mathematical model makes the LINGO can not obtain good results. Also, in average, the solutions qualities obtained by the *PT-HGA* are 4.3 percent better than the solutions of LINGO. Totally, the results shown in Table 1 indicate superiority of the proposed algorithm with respect to both CPU time and solution quality in compare to solutions of the LINGO.

Table 2. Results of medium and large-sized test problems (comparison with LB)

problem size (n×m)	the average performance ratio of PTHGA (%)	the average CPU time of PTHGA (in seconds)
5×5	10.26	149.34
5×10	18.86	787.66
10×5	16.4	1825.9
10×10	23.63	2126.58

Table 3. Results of medium and large size test problems (Comparison with the common cycle approach)

problem size (n×m)	the average improvement in PTHGA's solution compare to common cycle approach (%)
5×5	9.76
5×10	6.49
10×5	8.35
10×10	6.82

2) For the medium and large-sized problem instances, performance ratio λ has been calculated and used as a measure to evaluate the proposed algorithms. In Table 2, we have calculated the performance of the proposed algorithm. We observe that the average performance ratio for the problem instances increases when the problem size increases. However, this increase can be either due to an increase in the difference between the lower bound and the corresponding optimal cost, or due to reduction of effectiveness (performance) of the proposed algorithms due to increase in corresponding solution space.
3) Table 3 reports the average of cost differences between the solutions obtained through proposed algorithm and the solutions of common cycle approach. To do this, at first, for each problem of medium and large-sized test problems, time multiplier of all products was equated to 1. Then, the solution of common cycle approach has been calculated (see sections 3.2, 3.3 and 3.5). Finally, the solution of proposed PT-HGA has been calculated and compared with the common cycle approach. The solution of common cycle can be considered as an upper bound for solution of basic period approach (Yao and Elmaghraby, 2001). These results reveal the average improvement of 7.85 percent in solutions of the *PT-HGA* over the common cycle's solutions, respectively. From Table 3, it can be observed that when the problem sizes increases, the distance between common cycle and basic period solutions decreases. The main reason of this observation is that when the problem size increases the infeasibility of obtained solution; make it impossible to have different time multiplier (k_i) for products. Therefore, the solutions of two approaches become similar. Totally, these results indicate superiority of applying the basic period policy versus common cycle approach in the problem.

Conclusion remarks

In this paper, the basic period approach has been applied to solve the economic lot and delivery-scheduling problem in flexible flow lines over a finite planning horizon. To do so, a new mixed zero–one nonlinear model has been developed to solve the problem to optimality. Providing an optimal solution is not a practical approach for routine decision-making in case of the medium and large-sized problems. Thus, an efficient meta-heuristic (*PT-HGA*) based on power-of-two policy of basic period approach has been developed.

Since there is not another solution method for the problem, we have compared the solutions of *PT-HGA* with the solution of LINGO software in small-sized problems. Also, for medium and large-sized problems, we have calculated an index λ which calculates the distance of solutions obtained by *PT-HGA* from a lower bound. Computational results are very promising and indicate the superiority of *PT-HGA* over the common cycle approach with respect to the solution quality.

References:

1- Torabi, S. A., Karimi, B. and Fatemi Ghomi, S. M. T. (2005). "The common cycle economic lot scheduling in flexible job shops: The finite horizon case." *International Journal of Production Economics*, Vol. 97, PP.52-65.

2- Ouenniche J. and Boctor F. F. (1998). "Sequencing, lot sizing and scheduling of several components in job shops: The common cycle approach." *International Journal of Production Research*, Vol.36, No. 4, PP. 1125–1140.

3- Ouenniche J., Boctor F. F. and Martel A. (1999). "The impact of sequencing decisions on multi-item lot sizing and scheduling in flow shops." *International Journal of Production Research*, Vol. 37, No. 10, PP. 2253–2270.

4- Fatemi Ghomi, S. M. T. and Torabi, S. A. (2002). "Extension of common cycle lot size scheduling for multi-product, multi-stage arborscent flow-shop environment." *Iranian Journal of Science and Technology, Transaction B*, Vol. 26 (B1), PP. 55-68.

5- Hahm J. and Yano C. A. (1995a). "The economic lot and delivery-scheduling problem: The common cycle case."*IIE Transactions*, Vol. 27, PP. 113–125.

6- Hahm J. and Yano C. A. (1995b). "The economic lot and delivery scheduling problem: Models for nested schedules." *IIE Transactions*, Vol. 27, PP. 126–139.

7- Jensen M. T. and Khouja M. (2004). An optimal polynomial time algorithm for the common cycle economic lot and delivery scheduling problem, *European Journal of Operational Research*, Vol. 156, No. 2, PP. 305–311.

8- Torabi, S. A., Fatemi Ghomi, S. M. T. and Karimi, B. (2006). "A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains." *European Journal of Operational Research*, Vol. 173, PP. 173-189.

9- Bomberger, E. E. (1966). "A dynamic programming approach to the lot size scheduling problem."*Management Science*, Vol. 12, PP. 778-784.

10- Elmaghraby, S. E. (1978). "The economic lot scheduling problem: review and extensions." *Management Science*, Vol. 24, PP. 587- 598.

11- Yao, M. J. and Elmaghraby, S. E. (2001). "On the economic lot scheduling problem under power-of-two policy. "*Computers and Mathematics with Applications*, Vol. 41, PP. 1379–1393.

12- Ouenniche J. and Boctor F. F. (2001a). "The multi-product, economic lot-sizing problem in flow shops: the powers-of-two heuristic." *Computers and Operations Research*, Vol. 28, PP. 1165-1182.

13- Ouenniche J. and Boctor F. F. (2001b). "The two-group heuristic to solve the multi-product, economic lot-sizing and scheduling problem in flow shops." *European Journal of Operational Research*, Vol. 129, PP. 539-554.

14- Ouenniche J. and Boctor F. F. (2001c). "The G-group heuristic to solve the multi-product, sequencing, lot-sizing and scheduling problem in flow shops." *International Journal of Production Research*, Vol. 39, No. 1, 89-98.

15- Ouenniche J. and Bertrand, J. W. M. (2001). "The finite horizon economic lot sizing problem in job shops: the multiple cycle approach." *International Journal of Production Economics*, Vol. 74, PP. 49-61.

16- Cheng, R. and Gen, M. (1997). "Parallel machine scheduling problems using memetic algorithms." *Computers and Industrial Engineering*, Vol. 33, PP. 761–764.