A Hybrid GA for A Supply Chain Production Planning Problem

Masoud Jenabi
Department of Industrial Engineering,
Amirkabir University of Technology, Tehran, Iran.
m_jenabi@aut.ac.ir

S. Ali Torabi
Department of Industrial Engineering, Faculty of Engineering,
University of Tehran, Tehran, Iran.
satorabi@ut.ac.ir

S. Afshin Mansouri
CREST, Department of Computer Science,
King's College London, Strand, London WC2R 2LS, UK.
Afshin.Mansouri@klc.ac.uk

ABSTRACT

The problem of production and delivery lot-sizing and scheduling of set of items in a two-echelon supply chain over a finite planning horizon is addressed in this paper. A single supplier produces several items on a flexible flow line (FFL) production system and delivers them directly to an assembly facility. Based on the well-known basic period (BP) policy, a new mixed zero-one nonlinear programming model has been developed to minimize 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 could not be solved to optimality especially in real-sized problems. So, an efficient hybrid genetic algorithm (HGA) using the most applied BP approach (i.e. power-of-two policy) has been proposed. The solution quality of the proposed algorithm called PT-HGA has been evaluated and compared with the common cycle approach in some problem instances. Numerical experiments demonstrate the merit of the PT-HGA and indicate that it is a very promising solution method for the problem.

Categories and Subject Descriptors: J.2 [PHYSICAL SCIENCES AND ENGINEERING]: *Engineering*

General Terms: Algorithms, Experimentation, Performance

Keywords: Flexible flow lines; Lot and delivery-scheduling; Basic Period approach; Hybrid Genetic Algorithm (HGA)

1. INTRODUCTION

Nowadays, there is a high tendency to develop integrated models for simultaneously cost-effective planning of activities in supply chains. Among them, integrated production and delivery planning between adjacent supply parties is of particular interest.

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. Researches on the ELSP usually have focused on cyclic schedules with three well known policies: common cycle, basic period (or multiple cycle) and time varying lot size approaches [15]. Several authors have extended the ELSP to multistage production systems with common cycle production policy [9, 10, 4, and 14].

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 [5, 6] provided an excellent review of models related to ELDSP and developed two efficient heuristics to solve it based on the common cycle and nested schedule strategies. Jensen and Khouja [7] developed an optimal polynomial time algorithm for the ELDSP under common cycle approach. Finally, Torabi et al. [14] considered the ELDSP in flexible flow lines under the common cycle approach over a finite planning horizon. They developed an effective HGA to obtain near (or ideally) optimal solutions.

Regarding the basic period (BP) approach, Bomberger [1] 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 are imposed by the concept of BP. Elmaghraby [3] provided a review of the various contributions to ELSP and presented an improvement upon the BP approach, i.e. the extended basic period (EBP) method. Its main difference with the 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 [15] developed an evolutionary algorithm for ELSP under basic period policy. Ouenniche and Boctor [11, 12 and 13] 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 restricts applicability of the proposed techniques, because in practice, the planning horizon is often finite. In this regard, there are few research works which have assumed the finite planning horizon [8, 9 and 14].

Consequently, to the best of our knowledge, there has not been a research work on the 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 those of the common cycle [3] and this has been our main motivation in this research work.

The outline of the paper is as follows: problem formulation is presented in section 2. Section 3 explains the proposed PT-HGA. In section 4, an efficient procedure is developed for determining upper bounds on product's cycle time coefficients. An efficient feasibility test for capacity checking along with an iterative repair procedure to convert an infeasible solution to a feasible one, are proposed in section 5. Computational experiments are provided in section 6. Finally, section 7 is devoted to conclusion remarks.

2. PROBLEM FORMULATION

The following assumptions are considered for the problem formulation:

- Parallel machines are identical.
- Machines are continuously available and each machine can only process one product at a time.
- At the 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:

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 for 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	sequence vector in basic period k
$\sigma_{kk'j}$	sequence vector of machine $M_{k'j}$ related to the basic period k
r	number of production cycles over the finite planning horizon
$n_{kk'j}$	number of products assigned to machine $M_{k'j}$ related to the basic period k
F	basic period length
b_{ij}	production beginning time of product i at stage j (after related setup operation)
k_i	cycle time coefficient of product i

$$x_{i\ell k'kj} = \begin{cases} 1, & \textit{If product } i \textit{ at basic period } k \textit{ is assigned} \\ & \textit{to the } \ell - th \textit{ position of } M_{k'j} \\ 0, & \textit{Otherwise} \end{cases} .$$

It should be noted that the global cycle length is equal to the least common multiple of the k_i variables. In other words we 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 are 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 the value of the items are increased as they are processed at each stage, the h_{ij} values are non-decreasing; that is $h_{i,j-1} \leq h_{ij}$.

The objective function of this problem (Problem P) includes two fundamental elements. The first element is related to the setup costs and the second part computes the transportation cost of products on each basic period.

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

The inventory holding costs are often more complicated which are incurred at both the supplier and the assembler. Figure 1 shows the inventory curve of final product i in one cycle at the assembly facility. Therefore, the average inventory of component i per unit time at the assembly facility is: $k_i F / 2 \sum_{i=1}^{n} h_i d_i$.

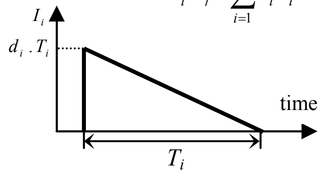

Fig. 1. Inventory level at the assembler in one cycle.

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

Referring to, Figure 2, it could be derived that the average WIP inventory of product i between two successive stages j-*1* and j per unit time is:

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

$$= d_i \left(b_{ij} + \frac{d_i k_i F}{2 p_{ij}} - b_{i,j-1} - \frac{d_i k_i F}{2 p_{i,j-1}} \right).$$

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

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

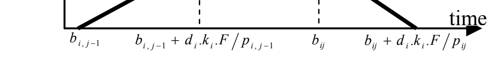

Fig. 2 WIP between stages *j-1* and *j* at the supplier

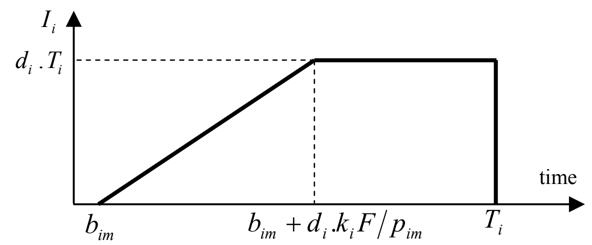

Fig. 3 Final product inventory at the supplier

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

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

$$= d_i\left(1 - .\frac{d_i}{2 p_{im}}\right) k_i F - d_i b_{im} .$$

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

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

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

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

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

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

Regarding this objective function and logical relationships between the variables of Problem P, a mixed zero-one nonlinear model is developed to obtain optimal solution of the problem.

Problem P has the following set of constraints. Constraints (2) state that no product can be processed on a given stage before it has been completed on its preceding stage. Constraints (3) guarantee that no product can be processed before the completion of its predecessor product in the production sequence ($\sigma_{kk'j}$). Constraints (4) reveal that at each position of each machine, there could be at most one product in process. Constraints (5) state that one product can be assigned at one position of machine $M_{k'j}$ only when another product is to be assigned at the preceding position of the same machine. Constraints (6) ensure the assignment of product *i* to one of the first k_i basic periods and imply that each assigned product at each stage has a unique position in the sequence of one machine. Constraints (7) determine the assignment of products in appropriate basic periods during the *H* basic periods. Constraints (8) denote that if product *i* has been assigned to the basic period *k* at stage *j*, it must be assigned to this basic period at all stages. Constraints (9) 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 the corresponding machine has been set up. Constraints (10) 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 (11) implies that the planning horizon *PH* is an integer multiplication of *H.F*, where *H*=LCM $(k_1,...,k_n)$, and *F* is the basic period length. Constraints (12) show that *r* is an integer number greater than or equal to one. Finally, Constraints (13) guarantee non-negativity of variables.

It must be 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. Then the corresponding optimal basic period length, optimal assignments, sequence vectors and the production and delivery schedule of products could be obtained via solving the Problem P.

3. 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 some selection process based on fitness of chromosomes and some genetic operators. To improve solution quality and to escape from converging to a 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 proposed PT-HGA, each solution is characterized by 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 which identifies the infeasible solutions and converts them to feasible schedules. This will be discussed in Sec 5.

3.1. Chromosome Representation

The proposed PT-HGA searches in the solution space consisting of k_i values, encoded as binary (zero-one) strings. Each k_i multiplier is represented by 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 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 $k_i = 2^{v_i}$). Due to the encoding of the k_i values into binary strings, we have to establish a mapping between each binary string and its corresponding integer k_i. We use the following equations to map a binary string consisting of u_i bits to an integer value k_i for the power of two and non-power of two cases, respectively:

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

3.2. 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.

Problem P :

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

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

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

subject to:

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

$$(3)\quad b_{ij} + \frac{d_i k_i F}{p_{ij}} + s_{ij} - b_{uj} \le M\left(2 - x_{i\ell k'kj} - x_{u(\ell+1)k'kj}\right);$$

$$i = 1,...,n, u \ne i; j = 1,...,m; k' = 1,...,m_j; \ell < n;$$

$$k = 1,...,H, H = lcm(k_1,...,k_n)$$

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

$$\ell = 1,...,n; k = 1,...,H, H = lcm(k_1,...,k_n)$$

$$(5)\quad \sum_{i=1}^{n} x_{i(\ell+1)k'kj} \le \sum_{u=1}^{n} x_{u\ell k'kj};\ i = 1,...,n; j = 1,...,m;$$

$$k = 1,...,m_j;\ \ell < n; k = 1,...,H, H = lcm(k_1,...,k_n)$$

$$(6)\quad \sum_{k'=1}^{m_j}\sum_{\ell=1}^{n}\sum_{k=1}^{k_i} x_{i\ell k'kj} = 1\ ; i = 1,...,n; j = 1,...,m$$

$$(7)\quad \sum_{k'=1}^{m_j}\sum_{\ell=1}^{n} x_{i\ell k'(t+bk_i)j} = \sum_{k'=1}^{m_j}\sum_{\ell=1}^{n} x_{i\ell k'(t+(b+1)k_i)j}\ ;$$

$$i = 1,...,n; j = 1,...,m; t = 1,...,k_i; b = 0,..., \frac{H}{k_i} - 2$$

$$(8)\quad \sum_{k'=1}^{m_j}\sum_{\ell=1}^{n} x_{ilk'kj} = \sum_{k'=1}^{m_j}\sum_{\ell=1}^{n} x_{ilk'k,j+1}\ ;$$

$$i = 1,...,m; j = 1,...,m, j < m; k = 1,...,H$$

$$(9)\quad b_{ij} \ge s_{ij} - M\left(1 - \sum_{k'=1}^{m_j} x_{i1k'kj}\right); j = 1,...,m,$$

$$i = 1,...,n; k = 1,...,H, H = lcm(k_1,...,k_n)$$

$$(10)\quad b_{im} + \frac{d_i k_i F}{p_{im}} \le F\ ; i = 1,...,n,$$

$$(11)\quad H.F.r = PH\ ; H = lcm(k_1,...,k_n)$$

$$(12)\quad r \ge 1,\ and\ integer$$

$$(13)\quad F \ge 0;\ b_{ij} \ge 0\ \forall i,j;\ x_{i\ell k'kj} = \{0,1\}; \forall i,\ell,k',j,k.$$

Given a vector of multipliers k_i; *i=1,...,n*, the procedure starts to create a vector say *V'* by sorting the products in ascending order of k_i. Ties are broken by sorting the products having the same multiplier k_i in descending order of ρ_i where:

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

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:

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

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

3.3. 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 the machines of different stages (Torabi et al. 2006). According to this procedure, for any given permutation vector *V*; the products are assigned to the machines of the first stage by using the FAM rule (if m_l> 1). Then, for each of the subsequent stages, the products are first sequenced according to the increasing order of their process completion times at the preceding stage. The products are then assigned to the machines at the current stage according to the FAM rule.

3.4. Initial Population

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

3.5. Evaluation Function

Each chromosome in the population represents a potential solution to the problem. The evaluation function assigns a real number to the individuals as their measure of fitness. In our problem, once the $\sigma_{kk'j}$ vectors are determined for each chromosome, the fitness value is obtained by solving the Problem P1 which is a NLP model. This problem is derived from Problem P by substituting the $x_{ilkk'j}$ variables by their 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 associated linear problem.

Iterative step: Increase *r* by 1 and solve the corresponding linear problem 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.

$$\begin{aligned}
&\textit{Problem}\quad P1:\\
&\textit{Min}\ \ Z = \frac{A}{F} + \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{sc_{ij}}{k_i F} +\\
&\quad \sum_{i=1}^{n}\left[h_i.k_i.\frac{d_i}{2}\left(3-\frac{d_i}{p_{im}}\right) + \sum_{j=2}^{m} h_{i,j-1}.k_i.\frac{d_i^2}{2}\left(\frac{1}{p_{ij}}-\frac{1}{p_{i,j-1}}\right)\right]F\\
&\quad + \sum_{i=1}^{n}\sum_{j=2}^{m} h_{i,j-1}.d_i\left(b_{ij}-b_{i,j-1}\right) - \sum_{i=1}^{n} h_i.d_i.b_{im}
\end{aligned}$$

$$\begin{aligned}
&\textit{subject to:}\\
&b_{i,j-1} + \frac{k_i F.d_i}{p_{i,j-1}} \le b_{ij};\ i=1,\dots,n,\ \ j=2,\dots,m\\
&b_{\sigma_{kk'j}(i-1),j} + \frac{k_{\sigma_{kk'j}(i-1)}F.d_{\sigma_{kk'j}(i-1)}}{p_{\sigma_{kk'j}(i-1),j}} - s_{\sigma_{kk'j}(i),j} \le b_{\sigma_{kk'j}(i),j};\\
&\qquad i=2,\dots,n_{kk'j};\ j=1,\dots,m;\ \ k'=1,\dots,m_j;\\
&\qquad k=1,\dots,H, H=lcm(k_1,\dots,k_n)\\
&b_{\sigma_{kk'j}(1),j} \ge s_{\sigma_{kk'j}(1),j};\ j=1,\dots,m;\\
&\qquad k=1,\dots,m_j; k=1,\dots,H, H=lcm(k_1,\dots,k_n)\\
&b_{im} + \frac{k_i F.d_i}{p_{im}} \le F;\ i=1,\dots,n\\
&r.H.F = PH;\ \ r \ge 1\,\textit{and}\ \ \textit{integer}\\
&F, b_{ij} \ge 0\ \ \forall i,j.
\end{aligned}$$

3.6. Selection and Crossover Operators

In the proposed PT-HGA, the tournament selection approach is used. It randomly chooses two chromosomes from the parent pool and selects the fitter one with the probability of φ ($0.5<\varphi<1$). Otherwise, the other one is selected. 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 the classic two point crossover scheme is employed. According to this scheme, two positions are randomly selected and the genes between them in the parent chromosomes are exchanged (see Figure 4).

3.7. 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 PT-HGA the swap mutation is used. Fig. 5 illustrates an example of this operator.

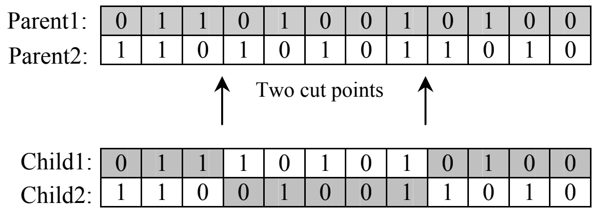
Parent1: 0 1 1 0 1 0 0 1 0 1 0 0
Parent2: 1 1 0 1 0 1 0 1 1 0 1 0
Two cut points
Child1: 0 1 1 1 0 1 0 1 0 1 0 0
Child2: 1 1 0 0 1 0 0 1 1 0 1 0

Fig. 4. Two-point crossover.

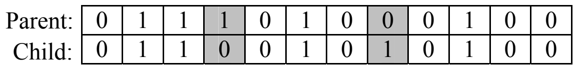
Parent: 0 1 1 1 0 1 0 0 0 1 0 0
Child: 0 1 1 0 0 1 0 1 0 1 0 0

Fig. 5. Swap mutation.

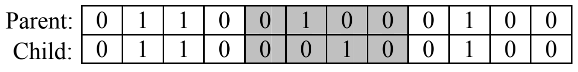
Parent: 0 1 1 0 0 1 0 0 0 1 0 0
Child: 0 1 1 0 0 0 1 0 0 1 0 0

Fig. 6. Inversion operator.

3.8. Local Improver

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

3.9. Population Replacement

Chromosomes for the next generation are selected from the enlarged population. Once the offspring are generated by the GA operators (crossover, mutation and the neighborhood search procedure), they are added to the current population called the enlarged population. Then 60 percent of the new population is filled by the best fitted chromosomes of the enlarged population. The remaining chromosomes are selected randomly from the remainder individuals in the enlarged population.

3.10. Termination Criteria

In our implementation, the PT-HGA stops if either a pre-determined number of generations (*max_gen)* are executed or the number of non-improving generations reaches *max_nonimprove*.

4. UPPER BOUNDS ON k_i VALUES

In order to represent all possible and feasible values of each k_i multiplier, an upper bound is determined for each one. In our PT-HGA, an upper bound on k_i is derived through determining an upper bound on the objective value of each product *i*. Following steps describe how an upper bound for each k_i (k_i^{UB}) can be computed:

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 the 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* as follows: $TC_{cc}^{i} = TC_{cc} - \sum_{u=1,u\neq i}^{n} TC_{cc}^{u}$. We would have: $TC^{i}_{BP} \le TC^{i}_{cc}$ (see Yao and Elmaghraby, 2001), where TC^{i}_{BP} is the 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 the following objective function:

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

Obviously, to obtain the optimal solution, the starting times have to be determined such 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:

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

Finally for a given value of F, we can derive an upper bound on v_i, denoted by v_i^{UB} by using the following equations:

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

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

$$\underset{i=1,\dots,n}{maximum}\left\{\sum_{j=1}^{m}s_{ij}+\sum_{j=1}^{m}\frac{d_iF}{p_{ij}}\right\}\le F$$
$$\Rightarrow F\ge\underset{i=1,\dots,n}{maximum}\left\{\frac{\sum_{j=1}^{m}s_{ij}}{1-\sum_{j=1}^{m}d_i/p_{ij}}\right\}.$$

5. 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 the products for all H basic periods are first calculated. The following procedure can be used for this purpose.

```
for $k=1,\dots,H$
  for each $i\in\sigma_k$
    for $j=1,\dots,m$
      $process1=\sum_{u\in\sigma_{kk'j}}k_ud_u/p_{uj}$, product $u$ is before $i$ at basic period $k$ on machine $M_{k'j}$.
      $process2=\sum_{u\in\sigma_{kk',j-1}}k_ud_u/p_{u,j-1}+k_id_i/p_{i,j-1}$, product $u$ is before $i$ at basic period $k$ on machine $M_{k'j-1}$
      $fin=max\{process1,process2\}+k_id_i/p_{ij}$
    end
    $ft_{ik}=fin$
  end
  $ft_k=max_i\{ft_{ik}\}$
end
```

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

For converting an infeasible solution to a feasible one, the following iterative repair procedure is proposed based on the 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 the largest process time ($max_i\ \{\sum_{j=1,\dots,m}k_id_i/p_{ij}\}$; $i\in\sigma_{k1}$), 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 multipliers. If this solution is feasible, stop, otherwise go to *step1*. If $v_i=0$; select the product with the next largest process time and go to *step3*.

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

6. COMPUTATIONAL EXPERIMENT

To verify efficiency of the proposed algorithm in terms of the solution quality and the required computation time, some numerical experiments are conducted. The experimental tests are implemented on a PC with an Intel Pentium IV 1800 MHz CPU. PT-HGA was coded on MATLAB 6.5. Moreover, LINGO 6.0 optimization software was used to solve the mixed zero-one non-linear models.

6.1. Parameter Setting and Data Set

The parameters of the PT-HGA were tuned empirically using some initial tests to these values: 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 were randomly generated from the following uniform distributions:
$d_i\sim U(100,1000)$, $p_{ij}\sim U(5000,15000)$,
$s_{ij}\sim U(0.01,0.025)$, $h_{i1}\sim U(1,10)$,
$A\sim U(10000,20000)$.

As stated earlier, the h_{ij} values must be non-decreasing for each j. So, after random generation of h_{i1} for each product i, the rest of 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 is computed using the following equation: $sc_{ij}=15000\times s_{ij}+1000\times U(0,1)$.

6.2. Performance Evaluation

To verify efficiency of proposed solution method, eight different problem sizes were considered. For each problem size, 20 problem instances were generated at random. The problem instances were divided in two sets: problem instances with 4 and 5 products, and 2 and 3 stages (small-sized problems), and problem instances with 5 and 10 products and 5 and 10 stages (medium and large-sized problems). For small size problems, the solutions of PT-HGA were compared with those of the LINGO software. For medium and large-sized problems, an index λ is calculated as λ= (TC-LB)/LB for comparisons, where TC is the total cost of a problem instance obtained by the algorithm and LB is the associated lower bound which is described subsequently.

6.3. A Lower Bound

The lower bound LB on the objective function is the minimum value of the following equation:

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

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

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

The minimum value of the above equation is obtined when b_{ij}'s are determined such that $(b_{ij}-b_{i,j-1})$ 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 late as possible, b_{im} can be substituted with $F-d_i.k_iF/p_{im}$. Therefore, a good lower bound can de computed as follows:

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

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

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

6.4. The results

The computational results for small problems are given in Table 1. Moreover, the results of two kinds of comparisons for medium and large sized problems are given in Tables 2 and 3.

The following results could be summarized based on the observations derived from Tables 1 to 3:

1. As could be seen in Table 1, in small size problems, *PT-HGA* outperforms LINGO in 67 out of 80 runs. It seems that the mixed zero-one and nonlinear nature of the proposed mathematical model prohibits LINGO to obtain good results. Moreover, the quality of the solutions obtained by the *PT-HGA* is, in average, 4.3 percent better than those of LINGO. In brief, the Table 1 indicates the superiority of the proposed algorithm versus LINGO with respect to both CPU time and solution quality.

Table 1 Results of small-sized test problems

Problem size (n×m)	Number of times the PT-HGA outperforms LINGO's solution	Superiority of PT-HGA's solution over LINGO's solution (%)	Average CPU time for LINGO (sec)	Average CPU time for PT-HGA (sec)
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

2. In Table 2, it could be observed that the average performance ratio of the PT-HGA degrades as the problem size increases. This performance reduction could be either due to the increased difference between the lower bound and the corresponding optimal cost, or due to the reduced performance of the proposed algorithm in large solution spaces.

3. Table 3 reports the average of cost difference between the solutions obtained by PT-HGA and the solutions of the common cycle approach. These results reveal the average improvement of 7.85 percent in solutions of the *PT-HGA* over the common cycle's solutions. These results indicate merit of applying the basic period policy instead of common cycle approach in the problem.

Table 2. Performance of PT-HGA in medium and large-sized test problems (Comparison with LB)

Problem size (n×m)	The average performance ratio (%)	Average CPU time (sec)
5×5	8.01	270.45
5×10	18	825.59
10×5	6.04	988.86
10×10	15.29	1869.1

Table 3. Performance of PT-HGA in medium and large-sized test problems (Comparison with the common cycle approach)

Problem size (n×m)	The average improvement in PT-HGA's solution compare to common cycle approach (%)
5×5	9.76
5×10	6.49
10×5	8.35
10×10	6.82

7. CONCLUSION

In this paper, the basic period approach was applied to solve the economic production and delivery lot-sizing and scheduling problem in flexible flow lines over a finite planning horizon. To do so, a new mixed zero–one nonlinear model was developed to solve the problem to optimality. Providing an optimal solution is cumbersome and not practical for the medium and large-sized problems. So, we have developed an efficient Hybrid Genetic Algorithm called PT-HGA. Performance of the *PT-HGA* was compared with LINGO software in small-sized problems. Also, for medium and large-sized problems a lower bound was employed to compare performance of the developed algorithm. Computational results are very promising and indicate the superiority of *PT-HGA* over the common cycle approach with respect to the solution quality.

8. ACKNOWLEDGMENT

S. Afshin Mansouri is supported in part by EPSRC under grant EP/D050863/1.

9. REFERENCES

[1] Bomberger, E. E., A dynamic programming approach to the lot size scheduling problem, *Management Science*, 12 (1966) 778-784.

[2] Cheng, R., Gen, M., Parallel machine scheduling problems using memetic algorithms, *Computers and Industrial Engineering*, 33 (1997) 761–764.

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

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

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

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

[7] Jensen M. T., Khouja M., An optimal polynomial time algorithm for the common cycle economic lot and delivery scheduling problem, *European Journal of Operational Research*, 156 (2) (2004) 305–311.

[8] Ouenniche J., Bertrand, J. W. M., The finite horizon economic lot sizing problem in job shops: the multiple cycle approach, *International Journal of Production economics*, 74 (2001) 49-61.

[9] Ouenniche J., Boctor F. F., Sequencing, lot sizing and scheduling of several components in job shops: The common cycle approach, *International Journal of Production Research*, 36(4) (1998) 1125–1140.

[10] Ouenniche J., Boctor F. F., Martel A., The impact of sequencing decisions on multi-item lot sizing and scheduling in flow shops, *International Journal of Production Research*, 37(10) (1999) 2253–2270.

[11] Ouenniche J., Boctor F. F., The multi-product, economic lot-sizing problem in flow shops: the powers-of-two heuristic, *Computers and operations research*, 28 (2001a) 1165-1182.

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

[13] Ouenniche J., Boctor F. F., The G-group heuristic to solve the multi-product, sequencing, lot-sizing and scheduling problem in flow shops, *International journal of production research*, 39(1) (2001c) 89-98.

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

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