Journal of Optimization in Industrial Engineering
Vol.12, Issue 2, Summer & Autumn 2019, 79-92
DOI: 10.22094/JOIE.2017.499.11

Research Article

The Preemptive Just-in-Time Scheduling Problem in a Flow Shop Scheduling System

Javad Rezaeian [a,*], Sadegh Hosseini-Kia [a], Iraj Mahdavi [b]

[a] *Department of industrial engineering, Mazandaran University of Science and Technology, Babol, Iran.*

Received 09July 2015; Revised 16April 2017; Accepted 19October 2017

Abstract

Flow shop scheduling problem has a wide application in the manufacturing and has attracted much attention in academic fields. From other point, on time delivery of products and services is a major necessity of companies' todays; early and tardy delivery times will result additional cost such as holding or penalty costs. In this paper, just-in-time (JIT) flow shop scheduling problem with preemption and machine idle time assumptions is considered in which objective function is minimizing the sum of weighted earliness and tardiness. A new non-linear mathematical model is formulated for this problem and due to high complexity of the problem meta-heuristic approaches have been applied to solve the problem for finding optimal solution. The parameters of algorithms are set by Taguchi method. Each parameter is tested in three levels. By implementation of many problems with different sizes these levels are determined .Genetic algorithm, imperialist competitive algorithm and hybrid of these algorithms are applied to solve the problem and the performance of the proposed algorithms are evaluated by many test problems. The Computational results indicate the superiority of the performance of hybrid approach than GA and ICA in finding the best solution in reasonable computational time.

Keywords: JIT scheduling; Flow shop; Preemption; Idle time

1. Introduction

In many industries, machines are arranged within flow shop systems based on production sequence, in which several machines are assigned to process jobs in series and all jobs have a same operation sequence. Each machine can only process one job at a time and processing of a job must be completed on the current machine before processing of the job is started on the succeeding machine (Mehravaran and Logendran 2012).Rezaeian et al (2013) presented a hybrid algorithm of genetic algorithm (GA) and Imperialist competitive algorithm (ICA)to minimizethe total weighted of makespan and maximum tardiness for a proposed hybrid flow shop scheduling problem,. Fu et al (2012) studied flow shop scheduling problem with incompatible job families and limited buffer to minimize the mean completion time of all jobs. Nikjo and Rezaeian (2014) considereda flow line manufacturing cell problem and solved it by a modified simulated annealing algorithm. Permutation flow shop scheduling with family setups has been regarded by Schaller (2012) in which the objective function is minimizing total tardiness. The problem of hybrid flow shop scheduling has been regarded by Najafi et al (2012) and a mathematical model and an immune system algorithm are presented. A mixed-integer linear programming model for $F_m|prmp, S_{ijk}|\sum U_j$ has been presented by Varmazyar and Salmasi (2012). Several meta-heuristic algorithms based on tabu search and imperialist competitive algorithms have been developed to solve the problem. Experimental result showed that the hybrid of these two algorithms produce better solutions than the other proposed algorithms for large size problems.

Timely delivery of products in many industries is one of the important goals. Producing a good before or after the delivery time imposed on manufacturers a cost. This cost can be considered as the opportunity cost of the money invested in inventory, storage and insurance costs and deterioration, in case of earliness and customer dissatisfaction, contract penalties, loss of sale and loss of reputation in case of tardiness (Liao and Cheng 2007).

In the scope of JIT scheduling, many articles have been published. Most studies in this issue build on minimizing of total earliness and tardiness or total weighted earliness and tardiness (see, e.g., Hendel and Sourd 2006; Esteve et al 2006; Baker and Scudder 1990). Finke et all (2007) studied Non-preemptive flow shop scheduling problem with unequal due dates with respect to finding a permutation schedule that minimizes sum of earliness and tardiness. To solve this problem a tabu search meta-heuristic combined with an LP evaluation function is applied. Also, permutation flow shop scheduling has been regarded by Schaller and Valente (2013) with the objective of minimizing total earliness and tardiness and a genetic algorithm is proposed to solve the problem. The permutation flow shop scheduling problem with earliness and tardiness penalties and common due date for jobs has been considered by Chandra et al (2009). They divided the problem into three cases: (i) the due date is such that all jobs are necessarily tardy; (ii) the due date is unrestricted; and (iii) the due date is between the two. A comprehensive approach for solving the problem over the entire range of due dates has been presented. Sun et al

*Corresponding author Email address: j.rezaeian@ustmb.ac.ir

(2012) proposed a model of identical parallel conveyor belt flow shop using JIT with unstable information based on the real situation of motorcycle assembly lines and an artificial fish swarm algorithm based on polar co-ordinate coding has been developed to solve the problem. The problem of minimizing makespan and sum of the earliness and tardiness of jobs in hybrid flow shop system with fuzzy tasks' operation times, due dates and sequence-dependent setup times has been investigated by Behnamian and FatemiGhomi (2014) and a bi-level algorithm extended to solve the problem. In a similar research Tadayoni Rad et al (2015) considered the same objective function in a two-stage assembly flow shop environment and theε -constraint method has been used to optimize the problem. Huynh Tuong and Soukhal (2010) studied minimizing of total weighted earliness–tardiness and due date cost in single machine and parallel machine systems with a common due date. Ventura and Radhakrishnan (2003) studied single machine scheduling with varying processing times and distinct due dates in JIT production environments and presented a binary linear integer mathematical model for the problem. Also, JIT one-machine scheduling with sequence-dependent setup times studied by Sourd (2005) and a branch-and-bound algorithm is presented for problem.

A different approach has been regarded in the JIT scheduling problems. Lann and Mosheiov (1996) considered the number of early-tardy jobs. The objective is maximizing the weighted number of jobs which are completed exactly on their delivery times and explained several applications of this problem. Shabtay (2012) investigated weighted number of JIT jobs problem in flow shop system. For this problem, four different scenarios have been regarded and for each one of the scenarios, an algorithm has been presented. Also Shabtay et al (2012) gave an analysis for two-machine flow-shop problem with two criteria including weighted number of JIT jobs and total resource consumption cost. They proved that the problem is NP-hard even for constant processing times and presented a polynomial time algorithm. a pseudo-polynomial dynamic programming algorithm have been introduced by Gerstl et al (2014) for the problem of maximizing the weighted number of Just-in-Time jobs on a proportionate flow shop which is faster than the algorithms presented before.

JIT scheduling by preemption consideration has not been attended in flow shop systems. If work process in a job interrupted before its completion time on machine referred as preemption. Recently Khorasanian and Moslehi (2017) proposed two mathematical models for a preemption flow shop scheduling problem by two machines which the first machine is multi-task and can be blocked. In order to solve the large sized instances a variable neighbourhood search algorithm (VNS) and a new variant of it, namely, dynamic VNS (DVNS), have been extended. DVNS has a better performance in compare to VNS based on computational results. AfsharNadjafi and Shadrokh (2010) minimized weighted earliness, tardiness and preemption penalties for scheduling projects. The problem of preemptive project scheduling studied by Hassanpour et al (2017) which the resources are constrained in multi-mode case. They indicated that presented multi-objective imperialist algorithm achieves to solutions with higher quality. Most papers in the field of preemptive JIT problems are presented for single machine system. Khorshidian et al (2011) minimized the total weighted earliness and tardiness of a single machine problem with the allowance of preemption and idle time for machine. They presented a genetic algorithm for this problem. Hendel et al (2009) studied JIT scheduling with preemption in single machine. They proposeda different computation of earliness related to the start time of the jobs. In other words, $T_j = \max(0, C_j - d_j^c)$ where $C_j$is completion time of job j and d_j^c is due date, $E_j = max(0, d_j^s - S_j)$ where S_j is start time of job j and $d_j^s = d_j^c - p_j$. Bulbul et al (2007) considered earliness/tardiness scheduling problem with preemption on a single machine. Also, Runge and Sourd (2009) presented a new model for preemptive scheduling on single machine. A local search algorithm was presented for the problem.

In the case of preemption allowance, the amount of work done for a job will be considered after preemption (Khorasanian and Moslehi (2017)). They presented by at least a flexible machine when preemption is allowed the solution space will be enlarged. Preemtion can be considered in manufacturing operations generally (Ebadi and Moslehi (2012)). A practical sample of preemption is presented Khorasanian and Moslehi (2017)for recovery of a patient when the bed is busy and the recovery will be continued after availability of bed .

In this study, JIT flow shop scheduling problem is developed by consideration preemption and machine idle time. Based on the general assumptions, buffers without capacity limitations are assumed between machines. Although, intermediate buffers may not be stayed because of technological requirements or process characteristics in some cases (Khorasanian and Moslehi (2017)). Furthermore there are no sequence dependent setup time and release date. A set of jobs $G = \{J_1, J_2, \dots, J_n\}$ has been considered. The objective function is minimizing $\sum_{j=1}^{n}(\alpha_j E_j + \beta_j T_j)$ which E_j and T_j are earliness and tardiness of job J_j. JIT preemptive One-machine scheduling is NP-hard (Hendel et al 2009). If earliness penalty (α_j) be equal to zero for any job then objective function would be modified to $\sum_{j=1}^{n}\beta_j T_j$. Lenstra et al (1977) proved that $1||\sum W_j T_j$ is strongly NP-hard.

In this study minimization problem of total weighted earliness and tardiness of jobs has been considered. This problem is equivalent to $F_m|prmp|\sum(\alpha_j E_j + \beta_j)$. Due to the complexity of this problem, meta-heuristic methods have been used to access an optimal solution in acceptable computational time.

In the rest of the study, in sections 2 and 3 parameters, variables and the mathematical model for the problem is explained. Proposed GA, ICA and hybrid algorithms have been described in section 4. Computational results are evaluated in section 5. Finally, in section 6 the conclusion and future research directions are presented.

2. Problem Definition

In this section, we express the problem in a mathematical model with linear and non-linear constraints so that preemption and machine idle time are allowable. The objective function deals with minimization earliness and tardiness of jobs.

The model variables are defined as follows: a set of machines $\{M_1, M_2, \dots, M_m\}$ and a set of independent jobs $\{J_1, J_2, \dots, J_n\}$ have been regarded. Each job J_j must be process on all machines and each machine can process only one job at the same time. As long as the processing of a job on current machine has not been completed, its process cannot start on next machine. Each job J_j has a due date d_j and processing time p_{ij} where i is related to the number of machines. Earliness and tardiness of job J_j are defined as $E_j = max(0, d_j - C_{mj})$ and $T_j = \max(0, C_{mj} - d_j)$, respectively where C_{mj} is the completion time of job J_j on last machine. All machines are steadily available and machine destruction does not happen. All jobs are available to process at zero time on first machine. α_j and β_j are earliness and tardiness penalties respectively.

Parameters:

i	1,2,…,m index for machines
j	1,2,…,n index for jobs
k_i	Index for time windows on machine i
H_i	Index for number of time windows on machine i
p_{ij}	Processing time of job J_j on machine i
d_j	Due date of job J_j

Variables:

C'_{ij}	Auxiliary variable to compute completion time of job J_j on machine i
C_{ij}	Completion time of job J_j on machine i
ST_i	Start time of machine i
S_{ij}	Start time of job J_j on machine i

$$X_{ik_ij} = \begin{cases} 1 & \text{if job } J_j \text{ on machine } i \text{ is placed in the position } k_i \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_{ij} = \begin{cases} 0 & \text{if } X_{i1j} = 1 \\ \text{M} & \text{otherwise} \end{cases}$$

$$S^{i}_{ik_ij} = \begin{cases} k_i + ST_i - 1 & \text{if } X_{ik_ij} = 1 \\ \text{M} & \text{otherwise} \end{cases}$$

3. The Mathematical Model

We formulated the model as follows:

$$MinZ = \sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j) \quad (1)$$

Subject To:

$$\sum_{k_i=1}^{H_i} X_{ik_ij} = p_{ij} i = 1, \dots, mj = 1, \dots, n \quad (2)$$

$$\sum_{j=1}^{n} X_{ik_ij} \le 1 \qquad i = 1, \dots, mk_i = 1, \dots, H_i \quad (3)$$

$$C'_{ij} = \max_{k_i=1}^{H_i} \left(k_i * X_{ik_ij}\right) i = 1, \dots, mj = 1, \dots, n \quad (4)$$

$$ST_i = \min_{j=1}^{n} \left(\delta_{ij} + C_{i-1,j}\right) i = 2, \dots m \quad (5)$$

$$C_{ij} = C'_{ij} + ST_i i = 2, \dots, mj = 1, \dots, n (6)$$

$$S_{ij} = \min_{k_i=1}^{H_i} S^{'}_{ik_ij}\, i = 2, \dots, mj = 1, \dots, n \quad (7)$$

$$C_{i-1,j} \le S_{ij} i = 2, \dots, mj = 1, \dots, n \quad (8)$$

$$T_j = max\left(0, C_{mj} - d_j\right) j = 1, \dots, n \quad (9)$$

$$E_j = max\left(0, d_j - C_{mj}\right) j = 1, \dots, n \quad (10)$$

The objective function (1) focused on minimizing total weighted earliness and tardiness. Constraint (2) ensures that each job is divided into p_{ij} period on each machine. We have H_i distinct period on each machine which each period is the smallest unit of time that a preemptionon jobs can happen. and H_i is length of schedule on machine i. constraint (3) ensures that in each period, only one part of one job can be processed or the machine does not process any job, in other words we have machine idle time. Constraints (4), (5) and (6) compute the completion time of each job J_j on machine i. constraint (7) calculates start time of each job J_j on each machine and constraint (8) guarantees that no jobs start on next machine before its completion on previous machine. Constraints (9) and (10) are related to the tardiness and earliness of jobs.

For verification of the proposed model, the mathematical model presented by Mehravaran and Logendran (2012) are used. Both models have been coded in Lingo software and the evaluation is done by test problems. To set same circumstances in both models, the parameters in designed test problems that do not exist in the other one (like sequence dependent setup times in proposed model by Mehravaran and Logendran (2012)) have been set equal to 0 and a penalty has been added to the objective function in our proposed model in order to prevent preemption. The results of running the codes have shown the validity of our model.

4. Proposed Algorithms

Here, GA, ICA and a hybrid of these algorithms are designed for solving the problem.

4.1. GA

The concept of genetic algorithm first used by Holland (1975) and popularized by Goldberg (1989). The main steps of this algorithm are as follows:

4-1-1. Chromosome encoding

The first step in the adoption and implementation of the genetic algorithms is mapping solution characteristics in the form of a chromosome. In this study, two-dimensional

chromosome is designed related to the number of machines and periods. To avoid a large increase in the search space, a dominant set is presented where the optimal solution exists in. A typical chromosome of a problem by two machines and two jobs is shown in Figure 1. The number of gens that allocated to each job on every machine is equal to the processing time of job j on machine i. Number of gens on each machine that are not allocated is considered as idle time. The number in each gen is related to job J_j and 0 is related to idle time.

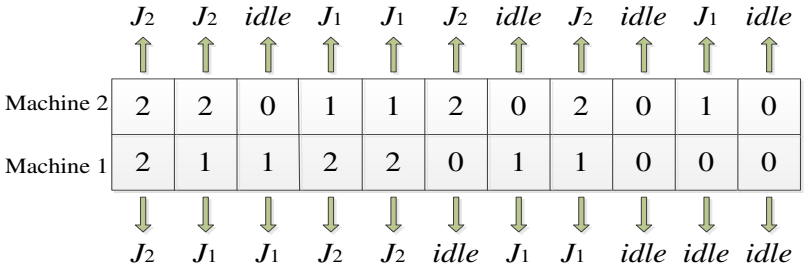

Fig 1. Decoding the chromosome

4.1.2. Initial population

Producing an initial set of solutions is the second step of the GA implementation. First, a sequence of jobs are randomly generated on a machine. Then, for next machine, jobs are randomly arranged according to their completion times on last machine. Number of genes of each machine must be greater than or equal to the sum of processing time of jobs on the machine, because of the allowance of idle time.

4.1.3. Fitness evaluation

The aim of the fitness evaluation is calculating competence of each solution in the population with regard to the objective function. By calculating the completion time of jobs, earliness and tardiness are calculated and the cost function of each chromosome is calculated with the following formula.

$$f = \sum_{j=1}^{n}(\alpha_j E_j + \beta_j T_j) \quad (11)$$

By calculating the objective function of each chromosome, a penalty added for violation of the constraints. If a solution does not satisfy the precedence constraint or the number of gens allocated to a job is not equal to the processing time of the job then a penalty is added to the objective function and is called adjusted objective function. This function is calculated by relation (12) where f shows the value of objective function, j_1 is first job on machine i, P'_{ij} is number of gens that are equal to j on machine i, γ and δ are penalties for violation of constraints.

$$Adjustedobjectivefunction(F_i) = f + \gamma\left(\sum_{i=1}^{m}\sum_{j=1}^{n}\left|\frac{P'_{ij}}{P_{ij}} - 1\right|\right) + \mu\left(\sum_{i=2}^{m}\sum_{j=1}^{n} max\left(\frac{C'_{i-1,j} - C'_{i-1,j1}}{S_{ij} - ST_i} - 1{,}0\right)\right) \quad (12)$$

Finally, the fitness function will be calculated by relation (13).

$$fitnessfunction = \exp\left(-\rho * (F_i / worstF_i)\right) \quad (13)$$

The parameter ρ is selection pressure and adjusted so that, the sum of selection probability for the first half of the population with better objective function equals 80 percent.

4.1.4. Selection strategies

There are many methods for selecting the population and each has its own advantage and disadvantage. Goldberg (1989) proposed roulette wheel method to select parents. In this study, the roulette wheel procedure as the most popular method is applied for selection.

4.1.5. Genetic operators

In the proposed GA, two types of crossover operator are used: single point crossover and double point crossover. Crossover operator performed on chromosomes that have been selected by the roulette wheel method. In single point crossover, one machine between [1,m] will be selected randomly and the gens of parents after the selected machine will be replaced. As shown in Figure 2, single point crossover may produce infeasible chromosome. For example, in sequence of offspring A and B, job 1 and job 2 doesn't satisfy its precedence constraints respectively (in offspring A, $C_{11} > S_{21}$ and in offspring B, $C_{12} > S_{22}$).

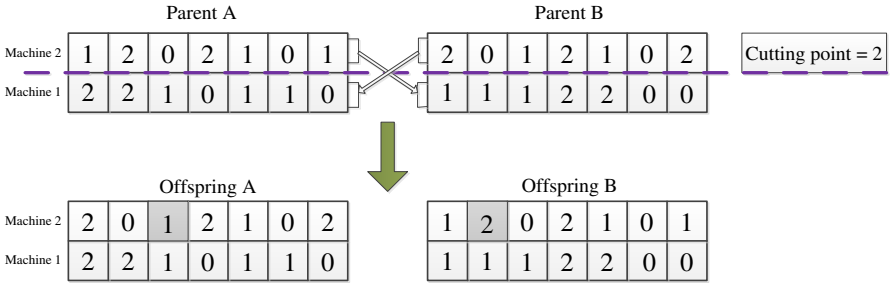

Fig 2. Single point crossover

Therefore, the chromosomes should be modified. The values of gen 3 on machine 2 in offspring A would be equal to 0 and first gen with 0 value after completion time of job 1 would be equal to 1. As it is illustrated in Figure 3, the relation $C_{i-1,j} \leq S_{ij}$ is confirmed for all jobs. In double point cross over operator, two machines are randomly selected between [1,m] and the genes of parents chromosomes between the two machines will be replaced.

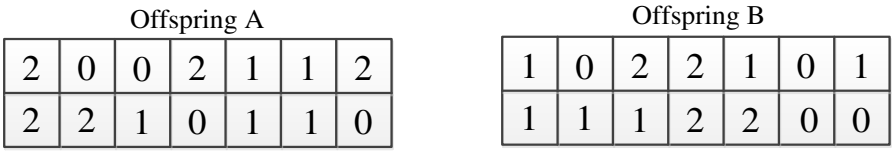

Fig. 3. Offspring A and B after modification

In this algorithm, swap mutation operator was also applied. In this operator, a chromosome and a machine will be selected randomly, then two random jobs between [1,n] will be selected and the last gen with these value will be replaced with each other. It may also produce infeasible chromosomes. To solve this problem, modification will be done as well as crossover operator. The swap mutation shown in Figures 4(a) and 4(b). In this figures selected machine is 2 and selected jobs are 1 and 2.

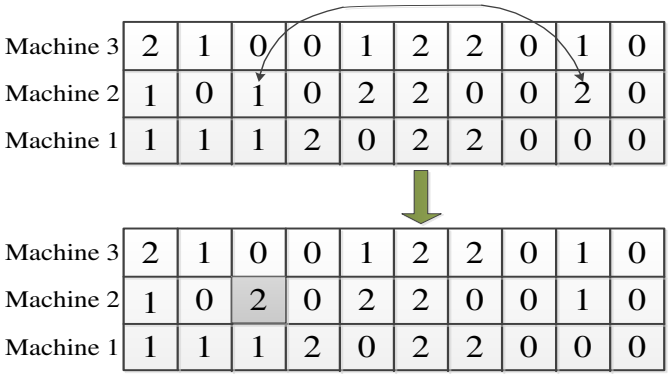

Fig. 4.(a). Swap mutation before modification

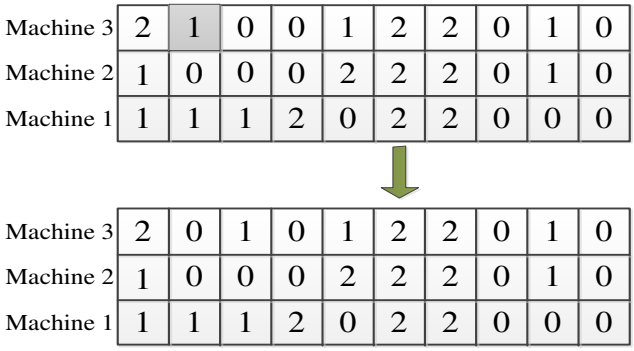

Fig. 4.(b). Swap mutation after modification

4.2. ICA

This method introduced by Atashpaz-gargari and Lucas (2007) for the first time and simulated the social-political process of imperialistic competition. Imperialist countries try to dominate another countries by legislate directly or through indirect methods, such as control of market and raw materials. Imperialism was a political control over other countries in order to use their resources or to prevent the dominance of other imperialist countries. Imperialist countries have intense competition to take possession of colonial countries. This competition led to the development of colonial countries in terms of political, economic and military. Imperialists competes for increasing the number of colonies and expansion of their sovereignty. Similarly to what mentioned above, ICA is simulated and such as many meta-heuristic algorithms is population based. Each individual constitutes a country or an imperialist. The competition of empires is the foundation of ICA. The weak empires will be dropped and the strong will be extend their colonies. In this manner the algorithm will be converged to the strongest empire and colony (Atashpaz-gargari and Lucas 2007). Main steps of the algorithm are as follow.

4.2.1. Initial empires

Countries in ICA have a similar role like chromosomes in GA. Also, initial countries are generated like initial chromosomes in GA. After producing this population, N_{imp} countries with better cost functions have been considered as empires and remaining countries (N_{col}) have been considered as colonies. According to the imperialist powers, the colonies will be distributed between imperialists which the normalized cost of each empire is calculated with following relation. $c_n$is the cost of nth empire and C_n is the normalized cost.

$$C_n = max\{c_i\} - c_n \tag{14}$$

And the normalized power of each empire is achieved by relation (15).

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right| \tag{15}$$

The number of colonies of each imperialist is estimated by relation (16).

$$NC_n = round\{p_n . (N_{col})\} \tag{16}$$

4.2.2. Assimilation policy

One of the most critical steps of the ICA is the movement of colonies to imperialist. For simulation of this policy in proposed algorithm crossover and mutation operators of GA have been used. At first two countries inside each empire will be selected randomly as parents that one of them will be selected from imperialists and the other one will be selected randomly among its colonies. Then crossover operation will be performed. Now, among two produced offspring and selected colony, best of them will be replaced with the selected colony. This process graphically is shown in Figure 5. The values of parameters for 2 jobs and 2 machines are shown in Table 1. Cost function of Parent *A* (imperialist), parent *B* (selected colony), offspring *A* and offspring *B* are equal to 10, 20, 5 and 7 respectively. So, offspring *A* with minimum cost function will be replaced with the selected colony. Then on this new colony mutation operator will be performed. If the produced colony is improved to a better position, mutated colony will be replaced.
By implementation of this policy, colonies may reach to a better position than the empire. In this mode of operation, the position of the colony and imperialist will be exchanged and the algorithm continues with the new empire. Now, the new imperialist began to impose assimilation policy on its colonies.

Table 1
The value of the parameters

Job J_j	P_{1j}	P_{2j}	d_j	α_j	β_j
J_1	2	3	11	2	3
J_2	3	2	12	2	2

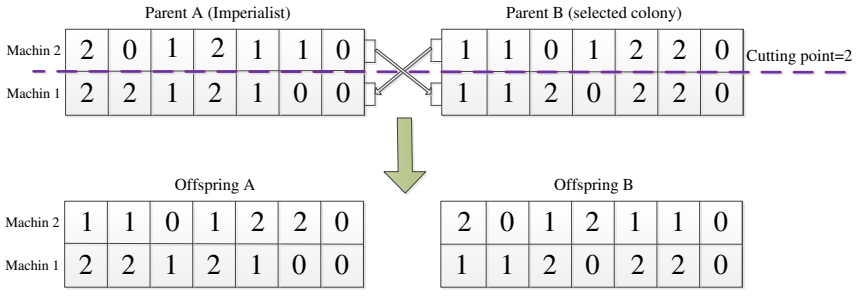

Fig. 5. Movement of colonies to imperialist

4.2.3.imperialistic competition

Total power of an empire is equal to sum of the imperialist power and a percentage of the power of its colonies and will be calculated by following formula where TC_n is total cost of nth empire and ξ is a number between 0 and 1.

$$TC_n = \text{Cost (imperialist}_n) + \xi \text{ mean } \{\text{Cost (colonies of empire}_n)\} \quad (17)$$

All empires try to take possession of colonies of other empires and control them (Atashpaz-gargari and Lucas 2007). This competition is modelled by picking one of the weakest colonies of the weakest empires and to take possession of this colony, a competition among all empires was created. First, the normalized total cost of each empire will be calculated.

$$NTC_n = \max_i(TC_i) - TC_n \quad (18)$$

Where NTC_n is the normalized total cost of nth empire. The possession probability of each empire will be calculated as follow.

$$p_n = \left| \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \right| \quad (19)$$

Based on the presented method by AtashpazGargari and Lucas (2007), vector P will be formed as follow.

$$P = [p_1, p_2, \ldots, p_{N_{imp}}] \quad (20)$$

A random vector R with the same size as P will be generated. This vector is an array of random numbers with uniform distribution on the interval [0,1].

$$R = [r_1, r_2, \ldots, r_{N_{imp}}] \quad (21)$$

The vector D will be produced as follow.

$$D = P - R = \left[D_1, D_2, D_3, \ldots, D_{N_{imp}}\right] = [P_{p_1} - r_1, P_{p_2} - r_2, \ldots, P_{pN_{imp}} - r_{N_{imp}}] \quad (22)$$

Referring to vector D, mentioned colonies will be assigned to an empire that its index on the vector D is maximized.

4.2.4. Convergence

Different conditions can be considered for the elimination of an empire. In the proposed algorithm, an empire will be eliminated when it lost its colonies. Stopping criteria is the end of all iterations of the algorithm. Then sequence of strongest empire among all remaining empires will be selected as the optimal sequence.

4.3. Hybrid Algorithm

Both of the two mentioned algorithms have their own advantages. Computational results show that the number of iterations in ICA is much more than of GA at the same time, but GA has more precise in solution. In the other word, faster convergence is the main advantage of ICA, on the other hand GA has a better global optimal achievement. The convergence of ICA and GA for a problem with 20 jobs and 3 machines are shown in Figure 6. As it shown, ICA has converged faster, but GA is produced better solutions than ICA. Here, a new hybrid algorithm based on GA and ICA is developed to use the advantage of both algorithms.

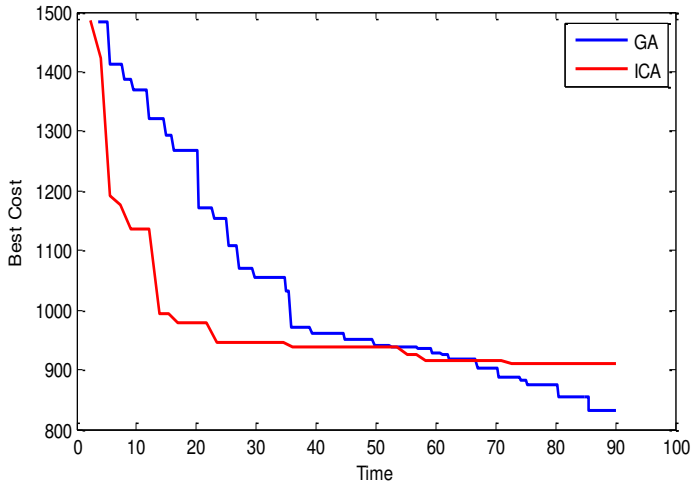

Fig. 6. The convergence in ICA and GA

The main tasks of the proposed hybrid algorithm can be divided into two main parts. First part is to create an efficient and proper population and the second part is improving this population in order to achieve an optimal solution. Since, ICA is faster than GA, it does a lot of search on the different sequence of the problem in order to find a proper and good population. Thus, ICA can be used as an efficient tool for creating an initial good population for GA. The precision of GA for this case can be a useful tool that improves initial population created by the ICA to achieve an optimal or near-optimal sequence. As shown in Figure 7, the hybrid algorithm has both advantages of ICA and GA.

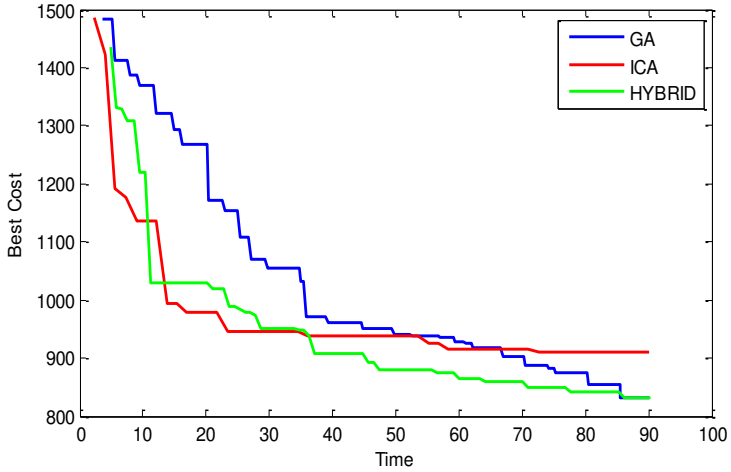

Fig. 7. The convergence in ICA, GA and hybrid algorithm

5. Computational Experiments and Analysis

5.1. Data generation

In this section, the performances of three mentioned algorithms have been investigated by many test problems in different sizes which the parameters follow a special distribution function. Processing times are generated from discrete uniform distribution [1,9], earliness and tardiness penalties are generated from discrete uniform distribution [1,4] and to produce deadlines, the following formula introduced by zegordi at al (1995) have been used where $\bar{d}$ is the mean of due dates and parameter τ is the earliness/tardiness function.

$$\bar{d} = (1-\tau) * \sum_{i=1}^{m} \sum_{j=1}^{n} P_{ij} \qquad (23)$$

Since, sum of processing times will be a large number and most of jobs may have earliness, the relation proposed by Koulamas (1998 a,b) has been used to produce $\bar{d}$, where M is maximum completion time achieved by Johnson's order.

$$\bar{d} = (1-\tau)M \qquad (24)$$

After calculating $\bar{d}$, due dates have been produced by relation (25) where parameter R is range of due dates.

$$\mathrm{d} = \left[\bar{\mathrm{d}} - \frac{\mathrm{R}}{2}\mathrm{M}\,, \bar{\mathrm{d}} + \frac{\mathrm{R}}{2}\mathrm{M}\right] \qquad (25)$$

Three different types of due dates produced considering $\tau = 0.2\,, R = 0.6$ ، $\tau = 0.35\,, R = 0.8$and $\tau = 0.5\,, R = 0.8$.

5.2. Parameters setting

In order to study the interplay between parameters and obtain the optimal combination of these parameters, a set of experiments by Taguchi method have been designed. Each parameter is tested in three levels. By implementation of many problems with different sizes these levels are determined. These factors and their levels in the hybrid algorithms are shown in Table 2 where Pop_{GA} is the number of initial population of GA, *iteration* is the number of iteration in GA, P_c shows the crossover rate, P_m shows the mutation rate, Pop_{ICA} represent the number of initial population of ICA, *Decade* is the number of iteration in ICA and P_{ICA} is the percent of the colonies role in determining the total power an empire.

Table 2
Factors and their levels

Factors	Levels
$\boldsymbol{Pop_{GA}}$	400, 600, 700
Iteration	60, 70, 80
$\boldsymbol{P_c}$	0.6, 0.7, 0.8
$\boldsymbol{P_m}$	0.3, 0.4, 0.5
$\boldsymbol{Pop_{ICA}}$	200, 300, 400
Decade	250, 300, 350
$\boldsymbol{P_{ICA}}$	0.05, 0.01, 0.15

The generated data have been analyzed by Minitab 14. The summary of the results are shown in Table 3 and parameters of the hybrid algorithm are set according to these values.

Table 3
Setting values of parameters

Factors	***Iteration***	P_m	P_c	Pop_{GA}	***Decade***	P_{ICA}	Pop_{ICA}
Level	80	0.5	0.6	600	350	0.1	400

5.3. Experimental results

The mentioned algorithms are coded in MATLAB7.11 and many test problems considered to evaluate the performance of proposed algorithms. The data are generated randomly and the algorithms are run by a computer with core i3 CPU (2.53 GHz / 3M cache) with 3GB RAM. Each test problem has been run 10 times and the best, worst and average of solutions are shown in Table 4. As it shown, for the small size of problems, solutions by three proposed algorithms are the same and the averages of solutions in GA for 4 test problems are better and in other test problems, the hybrid algorithm produced better solutions than the other algorithms.

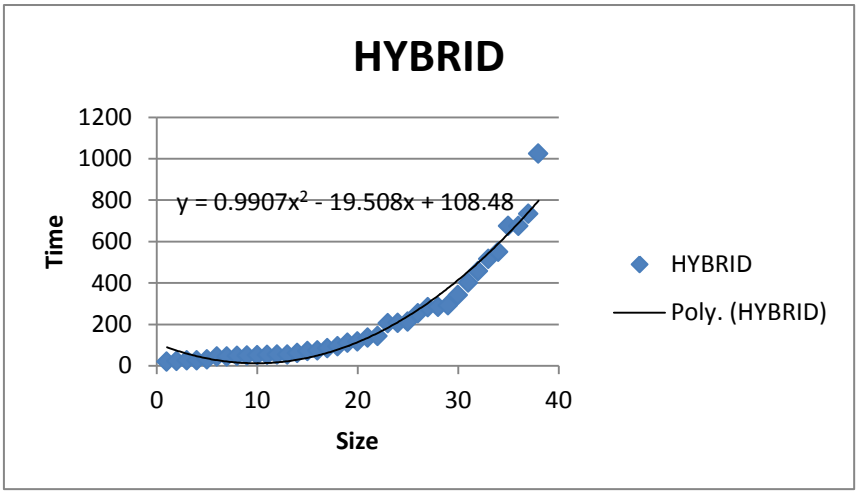

Fig. 8. Polynomial approximation of the hybrid algorithm

Table 4
Computational results of implementation of ICA, GA and Hybrid

Problem size(S=Small, M=Medium, L=Large)	**Number of jobs**	**Number of machines**	**Worst**			**Average**			**Best**		
			GA	**ICA**	**HYB**	**GA**	**ICA**	**HYB**	**GA**	**ICA**	**HYB**
S	**2 jobs**	2 machines	6	6	6	6	6	6	6	6	6
S		3 machines	9	9	9	9	9	9	9	9	9
S	**3 jobs**	2 machines	9	9	9	9	9	9	9	9	9
S		3 machines	10	10	10	10	10	10	10	10	10
S		4 machines	11	11	11	11	11	11	11	11	11
S		5 machines	28	30	28	27	27.6	26.9	26	26	26
S	**4 jobs**	2 machines	2	2	2	2	2	2	2	2	2
S		3 machines	10	10	10	7	7.6	6.9	6	6	6
S		4 machines	35	35	34	33	34.1	32.6	32	33	32
S		5 machines	43	43	42	41.6	42	41.3	41	41	41
S	**5 jobs**	2 machines	25	25	24	24.3	24.5	20	24	24	18
S		3 machines	26	27	26	22.5	24.2	22.2	21	23	21
S		4 machines	15	18	14	14.3	15.6	13	14	14	12
M		5 machines	103	107	101	98.3	101	96.3	93	94	92

S	**8 jobs**	2 machines	28	31	28	23.6	27.2	23.5	21	21	18
M		3 machines	82	105	69	67.2	74.1	61.7	56	62	55
M		4 machines	82	86	74	72.7	74.8	68.3	61	66	63
M		5 machines	147	151	133	112.6	130.2	110.6	90	112	90
M	**15 jobs**	2 machines	163	179	148	135.6	146.1	127.3	107	113	107
M		3 machines	203	259	180	152.6	190.1	149.2	114	154	121
M		4 machines	364	396	352	302.2	353.5	294.7	272	303	240
M		5 machines	907	932	842	658.5	693	642.4	438	463	427
M	**20 jobs**	2 machines	402	563	396	361.1	448.2	359.9	281	313	302
M		3 machines	738	829	706	653.7	758.5	629.2	525	684	518
M		4 machines	631	840	705	524.3	684.8	592.5	478	560	512
L		5 machines	890	1013	885	692	821	648	457	674	539
M	**25 jobs**	2 machines	572	596	535	426.1	511.5	425.2	338	454	316
M		3 machines	983	1243	962	833	1093.2	811.6	686	866	721
L		4 machines	1950	2663	2176	1743.4	2329.6	1683.3	1526	1857	1566
M	**30 jobs**	2 machines	902	1314	874	758.5	1029.6	734.2	627	827	620
L		3 machines	1097	968	1008	822	844.1	777	641	695	619
L		4 machines	2714	3535	2875	2474.9	2957.8	2612.2	2157	2316	2298
M	**35 jobs**	2 machines	1438	1807	1370	1293	1656.6	1260.5	1105	1415	1103
L		3 machines	2696	3090	2544	2432.9	2836.6	2421.9	2198	2277	2260
L		4 machines	3191	3515	3328	2849.3	3415.5	2945.1	2617	3251	2596
L	**40 jobs**	2 machines	2680	2906	2583	2317.8	2734.1	2400.2	2204	2568	2194
L		3 machines	4482	4295	4370	3945.9	3982.4	3832	3273	3687	3426
L		4 machines	4958	6015	5000	4589.1	5797	4582.4	4218	5195	4103

Also, Table 5 shows the best solutions of each algorithm with its computational time and optimal solutions, achieved by Lingo 9. The results show that ICA is faster than other algorithms but the quality of solutions are lower than others, and GA produced good solutions in more computational time. But the hybrid algorithm produced best quality of solutions with reasonable computational time.

Table 5
The compare of computational time and quality of solutions in proposed algorithms and Lingo

Problem size (S=Small, M=Medium, L=Large)	Number of Jobs	Number of Machines	Global Solver (Lingo)		GA		ICA		HYBRID	
			Optimal Solution	Computational time*	Best Solution	Computational time*	Best Solution	Computational time*	Best Solution	Computational time*
S	2 jobs	2 machines	6	0:00:56	6	0:00:24	6	0:00:15	6	0:00:19
S		3 machines	9	0:19:15	9	0:00:32	9	0:00:21	9	0:00:26
S	3 jobs	2 machines	9	0:06:19	9	0:00:27	9	0:00:17	9	0:00:22
S		3 machines	10	8:54:52	10	0:00:33	10	0:00:26	10	0:00:30
S		4 machines	---	10:00:00	11	0:00:59	11	0:00:38	11	0:00:44
S		5 machines	---	10:00:00	26	0:01:02	26	0:00:42	26	0:00:51
S	4 jobs	2 machines	---	10:00:00	2	0:00:28	2	0:00:23	2	0:00:25
S		3 machines	---	10:00:00	6	0:01:01	6	0:00:38	6	0:00:47
S		4 machines	---	10:00:00	32	0:01:12	33	0:00:41	32	0:00:53
S		5 machines	---	10:00:00	41	0:01:18	41	0:00:47	41	0:01:01
S	5 jobs	2 machines	---	10:00:00	24	0:00:50	24	0:00:38	18	0:00:44
S		3 machines	---	10:00:00	21	0:00:54	23	0:00:45	21	0:00:48
S		4 machines	---	10:00:00	14	0:01:03	14	0:00:42	12	0:00:54
M		5 machines	---	10:00:00	93	0:01:28	94	0:00:55	92	0:01:14
S		2 machines	---	10:00:00	21	0:01:07	21	0:00:44	18	0:00:52
M		3 machines	---	10:00:00	56	0:01:31	62	0:01:04	55	0:01:11

M	8 jobs	4 machines	---	10:00:00	61	0:02:04	66	0:01:12	63	0:01:33
M		5 machines	---	10:00:00	90	0:02:24	112	0:01:28	90	0:01:57
M	15 jobs	2 machines	---	10:00:00	107	0:01:38	113	0:01:13	107	0:01:24
M		3 machines	---	10:00:00	114	0:02:59	154	0:01:46	121	0:02:23
M		4 machines	---	10:00:00	272	0:04:32	303	0:02:38	240	0:03:32
M		5 machines	---	10:00:00	438	0:05:58	463	0:03:36	427	0:04:52
M	20 jobs	2 machines	---	10:00:00	281	0:03:02	313	0:01:35	302	0:01:50
M		3 machines	---	10:00:00	525	0:04:16	684	0:02:35	518	0:03:26
M		4 machines	---	10:00:00	478	0:05:57	560	0:03:21	512	0:04:43
L		5 machines	---	10:00:00	457	0:08:18	674	0:05:06	539	0:06:41
M	25 jobs	2 machines	---	10:00:00	338	0:03:48	454	0:01:54	316	0:02:16
M		3 machines	---	10:00:00	686	0:05:54	866	0:03:31	721	0:04:42
L		4 machines	---	10:00:00	1526	0:10:48	1857	0:06:26	1566	0:08:35
M	30 jobs	2 machines	---	10:00:00	627	0:04:59	827	0:02:32	620	0:03:15
L		3 machines	---	10:00:00	641	0:09:28	695	0:05:50	619	0:07:37
L		4 machines	---	10:00:00	2157	0:13:59	2316	0:07:10	2298	0:11:15
M	35 jobs	2 machines	---	10:00:00	1105	0:05:11	1415	0:03:06	1103	0:04:13
L		3 machines	---	10:00:00	2198	0:11:12	2277	0:06:50	2260	0:09:10
L		4 machines	---	10:00:00	2617	0:14:47	3251	0:09:00	2596	0:12:13
L	40 jobs	2 machines	---	10:00:00	2204	0:06:59	2568	0:04:09	2194	0:05:41
L		3 machines	---	10:00:00	3273	0:13:42	3687	0:08:28	3426	0:11:15
L		4 machines	---	10:00:00	4218	0:20:26	5195	0:12:20	4103	0:17:00

A polynomial approximation function of the hybrid algorithm is shown in Figure 8.

5.3.1. Experimental evaluation

To evaluate the mentioned algorithms, relative percentage deviation (RPD) as the most common performance measure is used according to relation (26).

$$RPD = \frac{sol_{avg} - sol_{min}}{sol_{min}} \quad (26)$$

Where sol_{avg} is the average value of the objective function of each algorithm and sol_{min} is the minimum value obtained by three proposed algorithms for each instance. The RPD values for each algorithm are calculated in Table 6.

Table 6
The RPD values of ICA, GA and hybrid algorithm

Number of Jobs	Number of Machines	GA	ICA	HYBRID	Number of Jobs	Number of Machines	GA	ICA	HYBRID
2 jobs	2 machines	0.00	0.00	0.00	15 jobs	3 machines	0.34	0.67	0.31
	3 machines	0.00	0.00	0.00		4 machines	0.26	0.47	0.23
3 jobs	2 machines	0.00	0.00	0.00		5 machines	0.54	0.62	0.50
	3 machines	0.00	0.00	0.00	20 jobs	2 machines	0.29	0.60	0.28
	4 machines	0.00	0.00	0.00		3 machines	0.26	0.46	0.21
	5 machines	0.04	0.06	0.03		4 machines	0.10	0.43	0.24
4 jobs	2 machines	0.00	0.00	0.00		5 machines	0.51	0.80	0.42
	3 machines	0.17	0.27	0.15	25 jobs	2 machines	0.35	0.62	0.35
	4 machines	0.03	0.07	0.02		3 machines	0.21	0.59	0.18
	5 machines	0.01	0.02	0.01		4 machines	0.14	0.53	0.10
5 jobs	2 machines	0.35	0.36	0.11	30 jobs	2 machines	0.22	0.66	0.18
	3 machines	0.07	0.15	0.06		3 machines	0.33	0.36	0.26
	4 machines	0.19	0.30	0.08		4 machines	0.15	0.37	0.21
	5 machines	0.07	0.10	0.05	35 jobs	2 machines	0.17	0.50	0.14
8 jobs	2 machines	0.31	0.51	0.31		3 machines	0.11	0.29	0.10
	3 machines	0.22	0.35	0.12		4 machines	0.10	0.32	0.13
	4 machines	0.19	0.23	0.12	40 jobs	2 machines	0.06	0.25	0.09
	5 machines	0.25	0.45	0.23		3 machines	0.21	0.22	0.17
15 jobs	2 machines	0.27	0.37	0.19		4 machines	0.12	0.41	0.12

This table shows that hybrid algorithm has better results than other algorithms. The least significant differences (LSD) intervals at the 95% confidence level for three proposed algorithms are shown in Figure 9. It shows that there is significant difference between three mentioned algorithms and the hybrid algorithm produced better solutions than ICA and GA.

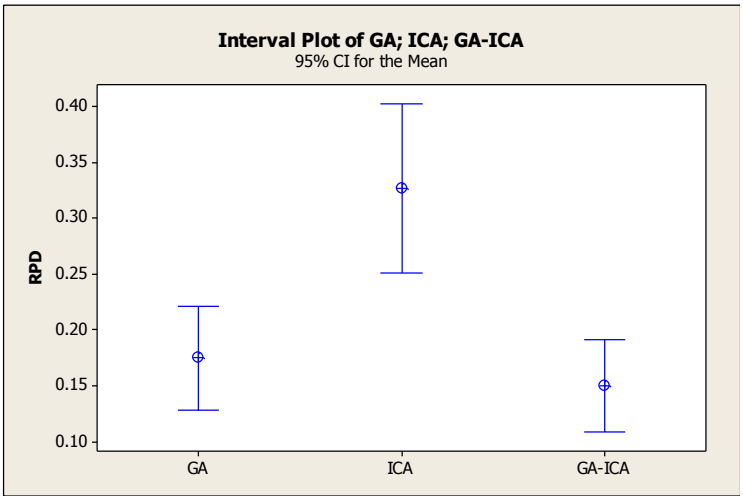

Fig. 9. LSD intervals for the type of algorithms

6. Conclusions

In this study, a preemptive flow shop scheduling with JIT approach was investigated where machine idle time is allowed. To assume this case a new non-linear mathematical model presented for the problem. Three meta-heuristic algorithms (GA, ICA and hybrid of GA and ICA)was proposed to solve the problem. To verify the effectiveness of the search algorithms, 38 test problems with three structures in small, medium and large sizes were produced and solved. The experiments analysis showed that the produced solutions by hybrid algorithm have a better quality than ICA and GA and their computational time is reasonable. Also, considering some assumptions like sequence dependent setup time can be applied for future researches.

References

Afsharnadjafi, B., & Shadrokh, S. (2010). The preemptive resource-constrained project scheduling problem subject to due dates and preemption penalties: An integer programming approach. *Journal of Optimization in Industrial Engineering*, Volume 1(Issue 1), 35-39.

Atashpaz-Gargari, E., & Lucas, C. (2007, 25-28 Sept. 2007). Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. Paper presented at the 2007 IEEE Congress on Evolutionary Computation.

Baker, K. R., & Scudder, G. D. (1990).Sequencing with Earliness and Tardiness Penalties: *A Review. Operations Research*, 38(1), 22-36. doi:10.1287/opre.38.1.22

Behnamian, J., & FatemiGhomi, S. M. T. (2014).Multi-objective fuzzy multiprocessor flowshop scheduling.*Applied Soft Computing*, 21, 139-148.doi:http://dx.doi.org/10.1016/j.asoc.2014.03.031

Chandra, P., Mehta, P., & Tirupati, D. (2009).Permutation flow shop scheduling with earliness and tardiness penalties.*International Journal of Production Research*, 47(20), 5591-5610. doi:10.1080/00207540802124301

Ebadi, A., & Moslehi, G. (2012). Mathematical models for preemptive shop scheduling problems. *Computers & Operations Research*, 39(7), 1605-1614. doi:http://dx.doi.org/10.1016/j.cor.2011.09.013

Esteve, B., Aubijoux, C., Chartier, A., & T'kindt, V. (2006).A recovering beam search algorithm for the single machine Just-in-Time scheduling problem.*European Journal of Operational Research*, 172(3), 798-813. doi:http://dx.doi.org/10.1016/j.ejor.2004.11.014

International Journal of Production Research, 45(21), 4899-4915. doi:10.1080/00207540600871228

Fu, Q., Sivakumar, A. I., & Li, K. (2012).Optimisation of flow-shop scheduling with batch processor and limited buffer.*International Journal of Production Research,* 50(8), 2267-2285. doi:10.1080/00207543.2011.565813

Gerstl, E., Mor, B., & Mosheiov, G. (2015). A note: Maximizing the weighted number of just-in-time jobs on a proportionate flow shop. *Information Processing Letters*, 115(2), 159-162.doi:http://dx.doi.org/10.1016/j.ipl.2014.09.004

Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning: Addison-Wesley, Reading, MA.

hasanpour, j., ghodoosi, m., & Hosseini, z. s. (2016). Optimizing a bi-objective preemptive multi-mode resource constrained project scheduling problem: NSGA-II and MOICA algorithms. *Journal of*

Optimization in Industrial Engineering, 10(21), 79-92.

Hendel, Y., Runge, N., & Sourd, F. (2009).The one-machine just-in-time scheduling problem with preemption. Discrete Optimization, 6(1), 10-22. doi:http://dx.doi.org/10.1016/j.disopt.2008.08.001

Hendel, Y., & Sourd, F. (2006).Efficient neighborhood search for the one-machine earliness–tardiness scheduling problem.*European Journal of Operational Research*, 173(1), 108-119. doi:http://dx.doi.org/10.1016/j.ejor.2004.11.022

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems: University of Michigan Press.

Huynh Tuong, N., & Soukhal, A. (2010).Due dates assignment and JIT scheduling with equal-size jobs.*European Journal of Operational Research*, 205(2), 280-289. doi:http://dx.doi.org/10.1016/j.ejor.2010.01.016

Khorasanian, D., & Moslehi, G. (2017). Two-machine flow shop scheduling problem with blocking, multi-task flexibility of the first machine, and preemption. *Computers & Operations Research*, 79, 94-108. doi:http://dx.doi.org/10.1016/j.cor.2016.09.023

Khorshidian, H., Javadian, N., Zandieh, M., Rezaeian, J., & Rahmani, K. (2011).A genetic algorithm for JIT single machine scheduling with preemption and machine idle time.*Expert Systems with Applications*, 38(7), 7911-7918.doi:http://dx.doi.org/10.1016/j.eswa.2010.10.066

Koulamas, C. (1998a). On the complexity of two-machine flowshop problems with due date related objectives. *European Journal of Operational Research*, 106(1), 95-100. doi:http://dx.doi.org/10.1016/S0377-2217(98)00323-3

Koulamas, C. (1998b). A guaranteed accuracy shifting bottleneck algorithm for the two-machine flowshop total tardiness problem.*Computers & Operations Research,* 25(2), 83-89. doi:http://dx.doi.org/10.1016/S0305-0548(97)00028-2

Lann, A., & Mosheiov, G. (1996).Single machine scheduling to minimize the number of early and tardy jobs.*Computers & Operations Research*, 23(8), 769-781. doi:http://dx.doi.org/10.1016/0305-0548(95)00078-X

Lenstra, J. K., RinnooyKan, A. H. G., &Brucker, P. (1977).Complexity of Machine Scheduling Problems. In E. L. J. B. H. K. P.L. Hammer & G. L. Nemhauser (Eds.), *Annals of Discrete Mathematics* (Vol. Volume 1, pp. 343-362): Elsevier.

Liao, C.-J., & Cheng, C.-C. (2007). A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Computers & Industrial Engineering*, 52(4), 404-413. doi:http://dx.doi.org/10.1016/j.cie.2007.01.004

Mehravaran, Y., & Logendran, R. (2012).Non-permutation flowshop scheduling in a supply chain with sequence-dependent setup times. *International Journal of Production Economics*, 135(2), 953-963. doi:http://dx.doi.org/10.1016/j.ijpe.2011.11.011

Najafi, E., Naderi, B., Sadeghi, H., & Yazdani, M. (2012).A Mathematical Model and a Solution Method for Hybrid Flow Shop Scheduling. *Journal of Optimization in Industrial Engineering*, 5(10), 65-72.

Nikjo, B., & Rezaeian, J. (2014).Meta heuristic for Minimizing Makespan in a Flow-line Manufacturing Cell with Sequence Dependent Family Setup Times.*Journal of Optimization in Industrial Engineering*, 7(16), 21-29.

Rezaeian, J., Seidgar, H., & Kiani, M. (2013). Scheduling of a flexible flow shop with multiprocessor task by a hybrid approach based on genetic and imperialist competitive algorithms. *Journal of Optimization in Industrial Engineering*, 6(13), 1-11.

Runge, N., & Sourd, F. (2009).A new model for the preemptive earliness–tardiness scheduling problem.*Computers & Operations Research*, 36(7), 2242-2249.doi:http://dx.doi.org/10.1016/j.cor.2008.08.018

Schaller, J. (2012). Scheduling a permutation flow shop with family setups to minimise total tardiness.*International Journal of Production Research*, 50(8), 2204-2217. doi:10.1080/00207543.2011.575094

Schaller, J., & Valente, J. M. S. (2013). A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness. *International Journal of Production Research,* 51(3), 772-779. doi:10.1080/00207543.2012.663945

Operational Research, 216(3), 521-532. doi:http://dx.doi.org/10.1016/j.ejor.2011.07.053

Shabtay, D., Bensoussan, Y., &Kaspi, M. (2012).A bicriteria approach to maximize the weighted number of just-in-time jobs and to minimize the total resource consumption cost in a two-machine flow-shop scheduling system. *International Journal of Production Economics*, 136(1), 67-74. doi:http://dx.doi.org/10.1016/j.ijpe.2011.09.011

Sourd, F. (2005).Punctuality and idleness in just-in-time scheduling.*European Journal of Operational Research*, 167(3), 739-751. doi:http://dx.doi.org/10.1016/j.ejor.2004.07.018

Sun, D.-h., Song, X.-x., Zhao, M., & Zheng, L.-J.(2012). Research on a JIT scheduling problem in parallel motorcycle assembly lines considering actual situations.*International Journal of Production Research,* 50(18), 4923-4936. doi:10.1080/00207543.2011.616232

Tadayoni Rad, S., Gholami, S., Shafaei, R., &Seidgar, H. (2015). Bi-objective Optimization for Just in Time Scheduling: Application to the Two-Stage Assembly Flow Shop Problem. *Journal of Quality Engineering and Production Optimization*, 1(1), 21-32. doi:10.22070/jqepo.2015.186

Varmazyar, M., & Salmasi, N. (2012).Sequence-dependent flow shop scheduling problem minimising

the number of tardy jobs.*International Journal of Production Research,* 50(20), 5843-5858. doi:10.1080/00207543.2011.632385

Ventura, J. A., & Radhakrishnan, S. (2003). Single machine scheduling with symmetric earliness and tardiness penalties. *European Journal of Operational Research,* 144(3), 598-612. doi:http://dx.doi.org/10.1016/S0377-2217(02)00163-7

Zegordi, S. H., Itoh, K., & Enkawa, T. (1995). A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem.*International Journal of Production Research*, 33(5), 1449-1466. doi:10.1080/00207549508930220

This article can be cited: Rezaeian, J., Hosseini-Kia, S. & Mahdavi, I. (2019)
The Preemptive Just-In-Time Scheduling Problem In a Flow Shop Scheduling System.
Journal of Optimization in Industrial Engineering. 12 (2), 79-92.

http://www.qjie.ir/article_538173.html
DOI: 10.22094/JOIE.2017.499.11