International Journal of Industrial Engineering & Production Research (2017)

September 2017, Volume 28, Number 3
pp. 251- 266

DOI: 10.22068/ijiepr.28.3.251 http://IJIEPR.iust.ac.ir/

Hybrid Artificial Immune System and Simulated Annealing Algorithms for Solving Hybrid JIT Flow Shop with Parallel Batches and Machine Eligibility

Javad Rezaeian* & Masoud Shafipour

Javad Rezaeian, *Mazandaran University of Science and Technology*
Masoud Shafipour, *Mazandaran University of Science and Technology*

KEYWORDS
hybrid flow shop,
parallel batching,
artificial immune system,
machine eligibility,
Earliness and Tardiness.

ABSTRACT

This research deals with a hybrid flow shop scheduling problem with parallel batching, machine eligibility, unrelated parallel machines, and different release dates to minimize the total sum of the weighted earliness and tardiness (ET) penalties. In parallel batching situation, it is supposed that a number of machines in some stages are able to perform a certain number of jobs simultaneously. Firstly, with respect to the proposed problem, a mixed integer linear programming model is developed. Then, since the problem is NP-hard, for solving large-sized problems, a hybrid meta-heuristic algorithm, which combines artificial immune system and simulated annealing, is proposed. The performance of hybrid algorithm is tested by some numerical experiments, and the results show its superiority to the other two algorithms.

1. Introduction

Sequencing and scheduling is a form of decision-making that plays a vital role in manufacturing and service industries. In the current competitive environment, effective sequencing and scheduling has become a necessity for survival in the marketplace.

Moreover, during the past two decades, issues of sequencing and scheduling in batch production systems have been taken into great consideration. Generally, mainspring of production in batches can be searched to avoid startup costs, handling costs or machine's ability to produce a batch. In general, scheduling models in a batch production, depending on batching, is classified into two types. The first type contains a number of jobs within the same batch that are processed one after another, called serial batches (s-batching problem), while the second type includes a group of jobs going through a machine and are processed simultaneously, called parallel batches (p-batching problem). Applications of scheduling models in a batch processing can be found in various industries containing foundry industry (Mathirajan et al., 2004), manufacturing furniture and home furnishings (Yaghubian et al., 1999), iron and steel industry (Oulamara 2007), aluminum foundry industry (Gravel et al., 2000), footwear industry (Fanti et al., 1996), and aircraft industry (Zee et al., 1997).

The hybrid flow shop (HFS or flexible flow shop (Pinedo 2008)) scheduling problem is a mixture of two particular types of scheduling problems: the parallel machine scheduling and the flow shop scheduling. Indeed, the HFS is composed of a set of two or more production centers (or

* Corresponding author: *Javad Rezaeian*
Email: j_rezaeian@ustmb.ac.ir
Received 10 April 2017; revised 21 June 2017; accepted 10 September 2017

stages) with at least one center having two or more parallel machines.

Because solving most of the scheduling problems is very tough (Graham et al., 1979), the proposed scheduling problems are only evaluated by a single criterion (e.g., makespan, total earliness, and so on) (T'kindt and Billaut 2005). However, many of the scheduling studies have displayed that most of industrial problems commonly include simultaneous incommensurable criteria, which can occasionally be inconsistent (Zitzler and Thiele 1999). Thus, for reflecting real-world status sufficiently, a bi-objective HFS scheduling problem, which consists of the earliness and tardiness (ET) penalties, is represented. The intended objective indicates just-in-time (JIT) production concept (Portmann and Mouloua 2007). Actually, JIT criterion comes from the make-to-order philosophy in management and production theory: an item should be delivered exactly when the customer requires it. Thus, both early and tardy deliveries of a task according to its due date is penalized (M'Hallah 2007).

In this study, we investigate a hybrid flow shop-scheduling problem with parallel batches, machine eligibility, unrelated parallel machines, release date, and just-in-time objective function. Since the hybrid flow shop scheduling is NP-hard even with two centers (or stages) and makespan objective function, both when preemption is allowed (Hoogeveen et al., 1996) and is not admissible (Gupta 1988), and considering the augmenting number of centers, the complexity of the problem increases; therefore, the complexity of the problem in this research is also NP-hard.

Khalouli et al. (2010) considered the application of an ant colony optimization (ACO) to HFS scheduling problem with identical parallel machines, without preemption and JIT objective function. Bertel and Billaut (2004) investigated a multi-processor flow shop scheduling problem with recirculation constraint and minimized the weighted number of tardy jobs' criteria. Firstly, authors proposed an integer linear programming formulation, and then a lower bound, a greedy algorithm, and a genetic algorithm were deemed as approximate approaches. In the literature, parallel batching in HFS environment has rarely been studied. First, Amin-Naseri and Beheshti-Nia (2009) regarded hybrid flow shop scheduling with parallel batching and completion time criteria. They explored not only three heuristic algorithms, but also a three-dimensional genetic algorithm (3DGA); after solving several test problems, they represented that 3DGA is better than the other heuristics. Later, Costa et al. (2014) presented a new genetic algorithm for solving HFS problem. Their problem assumptions are parallel batching, machine eligibility and incompatible job families with makespan criterion. At first, they proposed a mixed integer linear programming model for their problem. Then, they demonstrated the superiority of their proposed algorithm compared to some algorithms studied in the previous studies. Potts and Kovalyov (2000) reviewed the literature on scheduling with serial batching in three environments: single-machine, parallel-machines, and shop problems.

Allaoui and Artiba (2004) addressed a two-stage hybrid flow shop-scheduling problem with availability constraints. They considered makespan criterion and, after proposing the branch and bound model, calculated the worst-case performances of three heuristics: list scheduling algorithm, LPT heuristic, and H-heuristic. Ruiz and Maroto (2006) addressed a genetic algorithm for hybrid flow shop with sequence-dependent setup times and machine eligibility. They established four new crossover operators for the genetic algorithm. Ruiz et al. (2008) provided a mixed integer-programming model and some heuristics for hybrid flexible flow shop scheduling problems with sequence-dependent setup time, machine eligibility constraints, and minimization of the makespan criterion. Tadayon and Salmasi (2013) proposed a particle swarm optimization (PSO) for solving bi-objective function flexible flow shop with machine eligibility assumption. Their problem criteria are minimization of the average flow time and maximization of the lateness.

Jungwattanakit et al. (2008) investigated flexible flow shop problem with unrelated parallel machines and setup times. The objective functions of their problem were both minimization of makespan and the number of tardy jobs. They proposed a binary mixed integer programming model and also investigated three metaheuristic algorithms, including simulated annealing (SA), tabu search (TS), and genetic algorithms. Eventually, they concluded that SA algorithm is better than TS and genetic algorithms. Naderi et al. (2009) proposed a combinatorial simulated annealing for solving hybrid flow shop scheduling problem to minimize both total completion time and total tardiness criteria. They assumed sequence-dependent setup and transportation times for the problem. Sawik (2002) explored a mixed integer

programming procedure for minimization of makespan in flexible flow lines with finite intermediate buffers. The finite intermediate buffers between the stages will results in a problem with machine blocking, where a completed job has to remain on the past machine (i.e., blocking) (Pinedo 2008). Jin et al. (2006) considered multi-stage hybrid flow shop scheduling problem with minimizing completion time. They proposed a simulated annealing algorithm and a variable-depth search for solving the problem under their considerations. A batch-scheduling problem for flexible flow lines with minimizing setup costs and the mean flow time objective functions were regarded by Quadt and Kuhn (2007). They developed two-nested genetic algorithm to solve their problem. Choong et al. (2011) investigated the hybrid flow shop problem with minimizing makespan criterion. They proposed two hybrid heuristic algorithms in which particle swarm optimization (PSO) was merged both with simulated annealing (SA) and tabu search (TS), respectively. Ziaeifar et al. (2012) investigated hybrid flow shop scheduling problem considering processor assignment and minimizing completion time and cost of assigning a number of processors to stages as two objective functions. At first, they developed a new mathematical model, and then established a new heuristic algorithm based on genetic algorithm for gaining the best sequence of jobs and processor assignment.

Mirsanei et al. (2011) provided a novel simulated annealing (NSA) algorithm for solving hybrid flow shop scheduling with sequence-dependent setup times and minimizing the makespan. They compared the result of NSA with those of random key genetic algorithm (RKGA) and immune algorithm (IA). The results demonstrated that NSA is better than the two other algorithms. Engin and Doyen (2004) and Chung and Liao (2013) extended an effective artificial immune system (AIS) algorithm for solving HFS scheduling problem with minimization of the completion time.

The combination of algorithms, such as ant colony, simulated annealing, and variable neighborhood search (hybrid metaheuristic algorithm (HMH)), were adopted for the hybrid flow shop scheduling with sequence-dependent setup times constraint by Behnamian et al. (2012). Yaurima et al. (2009) considered a genetic algorithm to minimize the total completion time in hybrid flow shop. Their problem constraints are unrelated machines, sequence-dependent setup time, availability constraints, and limited buffers.

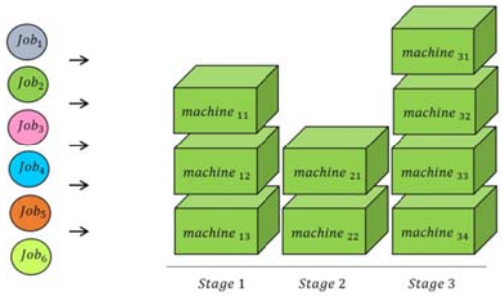

Fig. 1. Flexible Flow Shop Scheduling Problem with three stage and six jobs

Several research studies on hybrid flow shop scheduling problems have been reviewed and classified. Initially, Linn and Zhang (1999) addressed a survey on hybrid flow shop scheduling problem. Later, Kis and Pesch (2005) considered a review on exact solution methods for flexible flow shop in which preemption is not allowed. Afterwards, Ribas et al. (2010) proposed a comprehensive review in hybrid flow shop. They classified studies into two distinct points of view: the first one is based on the HFS characteristics and production limitations; the second is according to the solution approach. Ruiz and Vazquez-Rodriguez (2010) considered a literature review on exact, heuristic and metaheuristic approaches in hybrid flow shop environment.

Also, Wang (2011) presented a literature review on HFS scheduling problem and analyzed some common solution approaches including exact algorithms and approximation algorithms.

2. Problem Definition

This paper deals with a hybrid flow shop scheduling problem. Parallel batching, machine eligibility, unrelated parallel machines, release date are assumed for JIT scheduling. It is assumed that there are several workstations with unrelated parallel machines (in other words, machines in each stage are not identical). Each job in each stage should be processed by a particular machine; in fact, every machine cannot process every job.

A few machines in some stages have the ability to process a certain number of tasks simultaneously. In such a condition, completion time of each job of a batch is equal to maximum completion time of jobs belonging to that batch. It is also supposed that the amount of early and tardy penalties of each job is different and depends on the importance of each job.

Moreover, when a job starts to be processed, until its completion, it must be kept on the machine. In other words, interruption in the processing of a job is not allowed. Besides, sequence-dependent setup times are supposed to be zero and machine breakdowns are not allowed.

3. Mathematical Model

The problem considered in this research can be proposed formally as a mixed integer linear programing model. The indices, sets, parameters, variables, and mathematical model are as follows:

Indices	
j, k	Indices of jobs
m	Index of machines
i	Index of stages
Parameters/ sets	
N	Number of jobs
I	Number of stages
M_i	Number of machines at stage "*i*"
p_{jmi}	Processing time of job "*j*" on machine "*m*" at stage "*i*"
V_{ji}	Set of eligible machines to process job "*j*" at stage "*i*"
B_{mi}	Maximum batch size on machine "*m*" at stage "*i*"
r_j	Release date of job "*j*"
d_j	Due date of job "*j*"
We_j	the penalty weight per unit of time when job "*j*" is produced early
Wt_j	the penalty weight per unit of time when job "*j*" is produced late
Q	A large number
variables	
X_{jmi}	1,if job "*j*" is assigned to machine "*m*" at stage "*i*"; 0,otherwise
Y_{jki}	1,if job "*j*" precedes the job "*k*" on the same machine at stage "*i*";0,otherwise
U_{jki}	1,if jobs "*j*" and "*k*" are assigned to the same batch on a given machine at stage "*i*"; 0,otherwise
C_{ji}	Completion time of job "*j*" at stage "*i*"
E_j	Earliness of job "*j*"
T_j	Tardiness of job "*j*"

$$Min\ Z = \sum_{j=1}^{n} We_j E_j + Wt_j T_j \tag{1}$$

$s.t.:$

$$\sum_{m=1}^{M_i (m \in V_{ji})} X_{jmi} = 1 \qquad j = 1,2,\dots,N \ ; \ i = 1,2,\dots,I \tag{2}$$

$$C_{j0} = r_j \qquad j = 1,2,\dots,N \tag{3}$$

$$C_{ji} \geq C_{j(i-1)} + \sum_{m=1}^{M_i} X_{jmi}.P_{jmi} \qquad j=1,2,\dots,N \ ; \ i=1,2,\dots,I \tag{4}$$

$$C_{ji} + \sum_{m=1}^{M_i} X_{kmi}.P_{kmi} \leq C_{ki} + Q.(1-Y_{jki}) \qquad j,k = 1,2,\dots,N \quad j \neq k \ ; \ i = 1,2,\dots,I \tag{5}$$

$$\begin{cases} C_{ji} \le C_{ki} + Q.(1 - U_{jki}) \\ C_{ki} \le C_{ji} + Q.(1 - U_{jki}) \end{cases} \quad \begin{matrix} j = 1,2,\dots,N \ ; \\ k = j+1, j+2, \dots, N \ ; \\ i = 1,2,\dots,I \end{matrix} \qquad (6)$$

$$1 - Q.(2 - X_{jmi} - X_{kmi}) \le Y_{jki} + Y_{kji} + U_{jki} \quad \begin{matrix} j = 1,2,\dots,N \ ; \ k \\ = j+1, j \\ +2, \dots, N \ ; \\ i = 1,2,\dots,I \ ; \ m \\ = 1,2,\dots,M_i \end{matrix} \qquad (7)$$

$$\sum_{m=1}^{M_i} B_{mi}.X_{jmi} - 1 \ge \sum_{k=1}^{j-1} U_{kji} + \sum_{k=j+1}^{N} U_{jki} \quad \begin{matrix} j = 1,2,\dots,N \ ; \ i \\ = 1,2,\dots,I \end{matrix} \qquad (8)$$

$$C_{jI} - d_j = T_j - E_j \qquad j = 1,2,\dots,N \qquad (9)$$

$$X_{jmi} \in \{0,1\} \qquad \begin{matrix} j = 1,2,\dots,N \ ; \ i \\ = 1,2,\dots,I \ ; \\ m = 1,2,\dots,M_i \end{matrix} \qquad (10)$$

$$Y_{jki} \in \{0,1\} \qquad \begin{matrix} j,k = 1,2,\dots,N \quad j \ne k \ ; \\ i = 1,2,\dots,I \end{matrix} \qquad (11)$$

$$U_{jki} \in \{0,1\} \qquad \begin{matrix} j = 1,2,\dots,N \ ; \ k \\ = j+1, j \\ +2, \dots, N \ ; \\ i = 1,2,\dots,I \end{matrix} \qquad (12)$$

$$C_{jI}, T_j, E_j \ge 0 \qquad j = 1,2,\dots,N \qquad (13)$$

As presented in Eq. 1, the objective function is a weighted sum of earliness and tardiness times of each job. Constraint (2) ensured that each job is only assigned to one machine per stage with regard to those eligible ones.

Constraint (3) explains that the completion time of a given job at stage zero is equal to its release date. Constraint (4) describes the relationship between completion times of each job for two consecutive stages. Constraint (5) guarantees that if job *j* precedes job *k* in the same machine at stage *i*, job *j* must be completed before the start of job *k*. Through constraint (6), it is assured that if jobs *j* and *k* are assigned to the same batch to be processed by a determined machine at stage *i*, their completion times must be equal.

Constraint (7) explains that if jobs *j* and *k* are assigned to the same machine at stage *i*, one of the following states must occur: job *k* precedes job j, job j precedes job k, and k and j are processed within the same batch simultaneously. Constraint (8) ensures that the number of jobs assigned to a certain machine at stage *i* must not exceed its maximum batch size.

Constraint (9) computes the earliness and tardiness of each job. Constraints (10), (11), and (12) describe the binary variables; finally, constraint (13) defines positive variables.

4. Meta-Heuristic Algorithm

4-1. AIS background

The natural immune system is a very complex system with several functional components. It employs a multi-level defense against pathogens through non-specific (innate) and specific (acquired) immune mechanisms. Actually, the main work of the immune system (antibody) is to identify all molecules (or cells) within the body and classify those cells as self or non-self. The non-self cells are further categorized in order to stimulate an appropriate type of defensive mechanism. The immune system learns through evolution to distinguish between foreign antigens (e.g., fungi, viruses, bacteria, etc.) and the body's own cells or molecules.

The AIS, such as genetic algorithm (GA) and particle swarm optimization (PSO), starts working with a population of antibodies that each of them shows a point of solution space. Each antibody of the initial population consist of an affinity value with regard to its solution quality. In fact, an antibody with the better solution quality means that its antibody has higher affinity value. A generic of the AIS runs as follows: first,

according to a pre-determined affinity function (antigen), the number of the clones that will be proliferated from each selected antibody is computed. Those antibodies with higher affinity values have a more number of clones. Afterward, clones are mutated and the new generated antibodies are evaluated, and eventually, the worst antibodies of the initial population would be replaced by better mutated antibodies. As long as one of the stopping criteria is met, this procedure is repeated.

4-2. Proposed hybrid AIS-SA algorithm

Many research studies have shown that combined methods for scheduling problems can enhance the quality of the solution (Choong et al., 2011; Behnamian et al., 2012; Huang and Liao, 2006; Zhang et al., 2008; Dai et al., 2013). Meanwhile, a number of researchers demonstrated that if the AIS is merged with another algorithm, its performance is significantly increased (Naderi et al., 2009; Zhang and Wu, 2010).

For this reason, we propose a hybrid algorithm(AIS-SA) based on the characteristics of AISs and the annealing procedure of SA algorithms in order to both overcome the defects of each algorithm and augment their exploration and exploitation capabilities. The regarded AIS-SA algorithm utilizes a diversification technique from AISs to search for new and unknown regions in the solution space and can escape the local optimum by assigning a probability of selection to poorer solution in the annealing procedure of SA algorithms.

The details of the proposed algorithm are explained respectively in the following sections.

4-3. Antibody representation and initial population

Generally, the hybrid flow shop scheduling problem requires a particular representation, which individually runs the job sequencing and the assigning jobs to machines in each stage (Sherali et al., 1990; Rajendran and Chaundhuri, 1992). Since, for just-in-time objective functions, jobs must be completed on time, not earlier or later than their due dates; otherwise, a penalty will be imposed. Thus, in this study, two vector strings for each station have been represented. In this type of encoding, each station contains two vector strings where the first vector represents sequence of jobs and the second vector displays assignment of jobs to machines in each stage. Assignment of jobs to machines at each stage is done according to the machine eligibility constraints. Example 1 illustrates the construction of an antibody in this method.

Example 1: Consider a HFS scheduling problem with 7 jobs and 3 stages. There exist three machines in the first stage, two machines in the second, and finally four machines in the third one. Table 1 shows that the machines on which each job can be processed in each stage.

Tab. 1. Processing capabilities of machines in each stage

	$i=1$	$i=2$	$i=3$
$j=1$	[1,2]	[2]	[3,4]
$j=2$	[1,3]	[1,2]	[1,2,4]
$j=3$	[1,2,3]	[1,2]	[1,2,3]
$j=4$	[2]	[2]	[1,2,3,4]
$j=5$	[2,3]	[1,2]	[1,2,3]
$j=6$	[1,2,3]	[1]	[2,3,4]
$j=7$	[3]	[1,2]	[1,4]

With regard to Table 1, a feasible solution could be as in Figure 2.

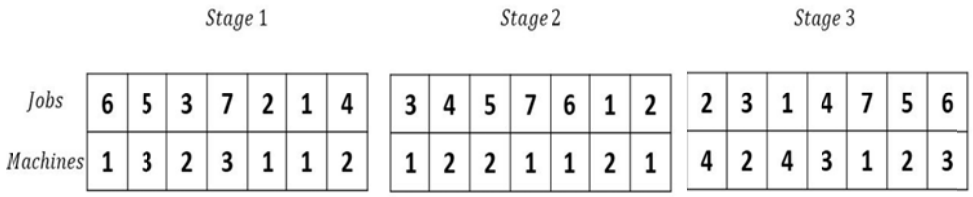

Fig. 2. Antibody structure

With regard to Table 1, a feasible solution could be as in Figure 2.

Since the artificial immune systems have been satisfactorily used for a wide range of theoretical problems and real-world applications (Hart and Timmis 2008), immune-inspired algorithms establish a new way to make use of the particular characteristics of specific problems to improve the solution quality.

For example, in stage 2, jobs 3, 7, 6, and 2 must be performed sequentially on machine 1; similarly, jobs 4, 5, and 1 must be performed on machine 2. Meanwhile, according to this type of encoding, the initial population of antibodies is generated randomly.

4-4. Antigen and calculation of affinity value

In fact, antigen is the objective function of the problem that must be optimized; as mentioned before, it is the weighted sum of early and tardy jobs. Each antibody has an ET penalty value that must be used for the calculation of affinity value. The smaller ET penalty values and higher affinity values are optimal; therefore, the affinity value for each antibody is determined using Eq. 14:

$$\text{Affinity (z)} = \frac{1}{\text{ET penalty(z)}} \quad (14)$$

According to the above equation, it is evident that the antibody with lower ET penalty value (higher affinity value) is better than the antibody with the higher one.

4-5. Selecting and cloning

In the search process of AIS, *Nbest* antibodies with the highest affinity values are selected from *PopAb* (Population of Antibodies) for cloning. According to the paper by Lin and Ying (2013), the number of clone for each *Nbest* selected antibodies is equal to$(Nbest - r + 1)$, where r is ranking number of the selected antibodies in *PopAb*. In this way, the antibody with higher affinity value has a more number of clones. Furthermore, this phase not only will increase the average affinity value, but also give to the later steps more opportunity to move further toward the best antibody.

4-6. Mutation operators

In this research, three mutation operators are applied to change the sequence and assignment of the antibodies, *shift*, *swap*, and *new mutation*. Note that for this AIS, the mutation operators are performed in two phases. In the first phase, one or more stage is selected randomly, and then, in the second phase, one of the following mutation operators is executed:

Shift: The shift mutation operator after choosing a gene (column) randomly shifts it to a random position, which is placed in the right side of the current position (Kleeman and Lamont 2007). This mutation is shown in Figure 3.

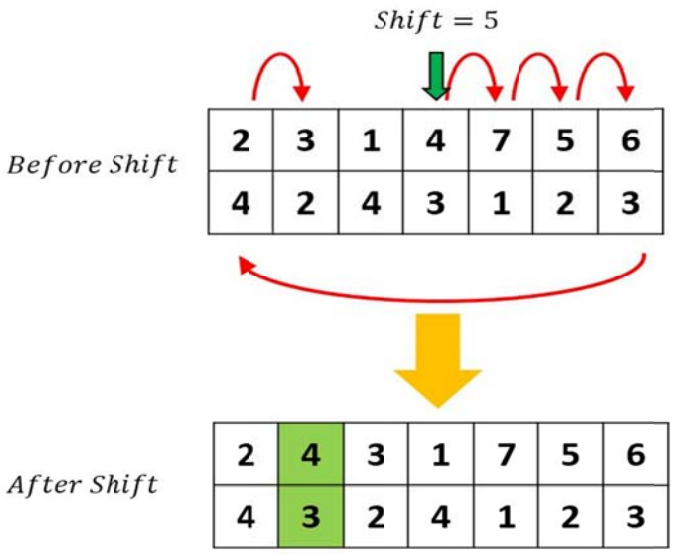

Fig. 3. Shift mutation operator in the proposed algorithm

Swap: The swap mutation is carried out by randomly selecting a pair of jobs (column) for exchanging them. Figure 4 displays an instance of this mutation.

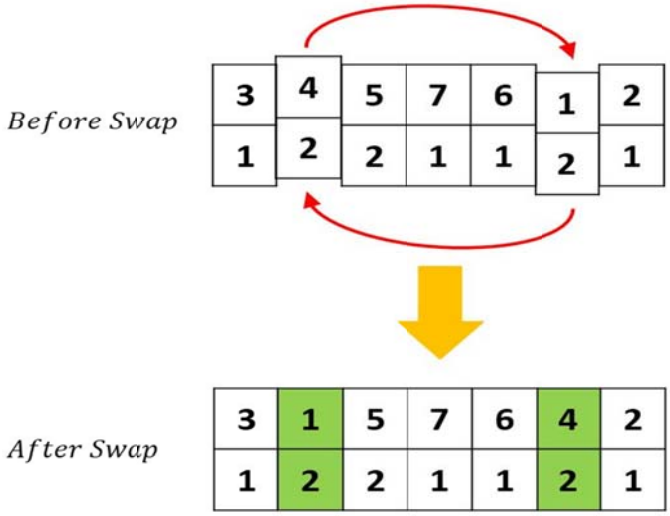

Fig. 4. Swap mutation operator in the proposed algorithm

New Mutation: The two previous operators only change the sequence of jobs and do not modify the assignment of jobs in machines. For this reason, we present a new mutation for the problem under consideration. Using this mutation operator, after selecting one or more genes on the first vector, a number of jobs in these genes will be extracted. Then, according to Table 1, a new machine is selected and replaced by the previous machine. For example, at stage 3, gen number 2 with value job 3 is selected randomly; then, according to Table 1, machines 1, 2, and 3 have the ability to process this job in stage 3. Finally, between machines 1 and 3, machine 1 is selected randomly (see Figure 5).

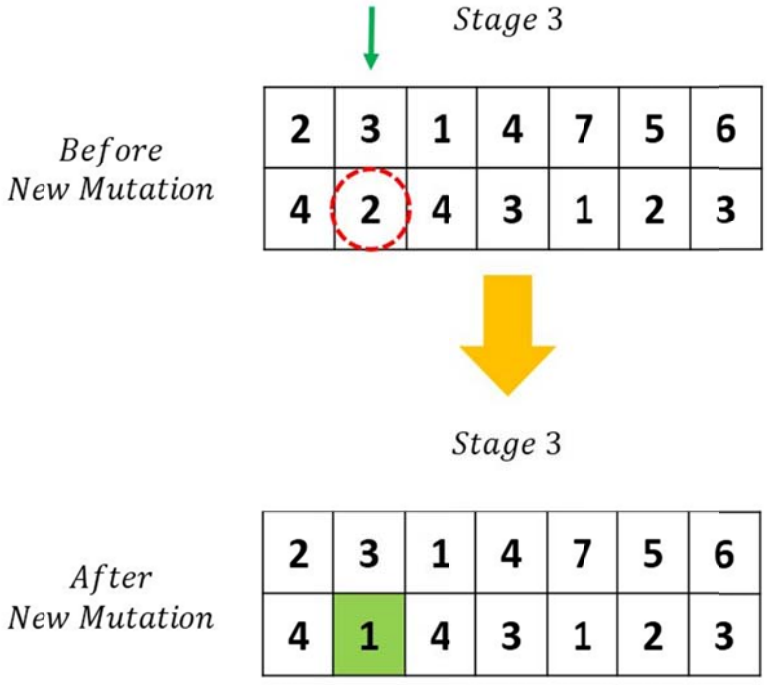

Fig. 5. New mutation operator in proposed algorithm

4-7. Updating population of antibody

After selecting *Nbest* of better mutant antibodies and *Nbest* of worse antibodies in original population, to combine the AIS with the annealing procedure of SA, if the affinity value of the mutant antibody is not worse than corresponding worst selected antibody, the original one is replaced by the mutant antibody;

otherwise, it is replaced under a certain probability. Let Δf explain the difference of ET penalty between the mutant antibody and the original one. The probability of accepting the mutant antibody is $e^{-\Delta f/T}$, where T defines the current temperature of the system. This is done by creating a random number $k\epsilon[0,1]$ and replacing the original antibody with the mutant antibody if $k < e^{-\Delta f/T}$. The generated new population of antibodies are sorted according to their affinity values and be prepared for cloning in the next generation.

The hybrid proposed algorithm includes the following steps:

Step 1: Parameters setting of AIS and SA: set the number of initial population of antibody ($PopAb$), the number of iterations (Nit), number of antibody with higher affinity values for cloning procedure ($Nbest$), cloning rate (Cr), initial temperature (T_0), cooling rate (α).

Step 2: Initial population generation and calculated affinity value of each antibody: create a population of *PopAb* antibodies and evaluate the objective function (affinity) for each antibody.

Step 3: Check the stopping criterion. If it is true, go to step 12; otherwise, go to step 4.

Step 4: Select the number of *Nbest* of better antibodies for proliferation (cloning) phase.

Step 5: Proliferate each selected antibodies in the previous step according to cloning rate.

Step 6: Mutate all antibodies of clone population using the swap, insertion, and new mutation operators and then evaluate them.

Step 7: Select *Nbest* of better of mutant antibodies and *Nbest* of worse antibodies in original population.

Step 8: For all *Nbest* of better of mutant antibodies, doing step 9, if all better antibodies have been considered, go to step 11.

Step 9: If the affinity value of mutant antibody is better than corresponding worst selected antibody, replace mutant antibody by corresponding worst selected antibody and go to step 8; otherwise, go to step 10.

Step 10: Generate a random number "k" ($k\epsilon[0,1]$) , if $p = \exp[-\frac{\Delta f}{T}] > k$, replace mutant antibody by corresponding worst selected antibody, and go to step 8; otherwise, keep the worst antibody selected in its location and go to step 8.

Step 11: Due to the rate of cooling, reduce the temperature of the system and go to step 3.

Step 12: Present the best antibody and stop.

5. Computational Experiments

The performance of the proposed AIS-SA algorithm for solving the problem under consideration will be explained in this section and evaluated by running some test problems. Data generation, parameters tuning, and the comparison of computational results of the proposed AIS-SA algorithm with some other algorithms are described as follows.

5-1. Data generation

Since the scheduling problem, examined in this study, has not been regarded in the previous studies, to evaluate the performance of the proposed hybrid algorithm in various situations, the combination of well-known benchmark problems in the literature is used. Amin-Naseri and Beheshti-Nia (2009) and Costa et al. (2014) represent a set of benchmark problems for hybrid flow shop scheduling problem with parallel batching constraint. The presented benchmark problems by them are randomly generated according to the following five factors: number of jobs N , number of stages I , number of machines in each stage M_i, processing time p_{jmi}, maximum batch size of each machine per stage B_{mi}. Three levels of data will be generated for each parameter shown in Table 2. For example, the high level for machine consists of a uniform random number between 1 and 5 per stage.

Tab. 2. Different parameters for benchmark problems

Parameters	*Low*	*Medium*	*High*
Number of jobs	10	50	100
Number of stages	3	10	20
Number of machines in stage "i"	$U[1,3]$	$U[1,5]$	$U[1,10]$
Maximum batch size on machine "m" at stage "i"		$U[1,3]$	$U[1,5]$

Processing time of job "j"*on machine* "*m*" *at stage* "*i*"	$U(5,12)$
Release date of job "*j*"	$U(0,2)$
the penalty weight per unit of time when job "j"*is produced early or late*	$U(2,5)$

There are 54 kinds of problems ($3 \times 3 \times 3 \times 2 \times 1 \times 1 \times 1 = 54$). In this study, for each combination of seven factors, five test problems were generated randomly. Then, the total numbers of 270 test problems were considered. In addition, Khalouli et al. (2010) considered to calculate the due date (d_j) of each job (j) in flexible flow shop environment with JIT criterion and identical parallel machines. Therefore, equation (15) was proposed for unrelated parallel machine state.

$$d_j = \sum_{j=1}^{N} (p'_{ji}) \times (1 + C) \quad \forall\, i = 1,2, \dots, I \tag{15}$$

where $C \epsilon [0\ 1]$ and p'_{ji} obtained as following:

$$p'_{ji} = \max(p_{jmi}) \qquad \forall\, m = 1, \dots, M_i \tag{16}$$

In fact, p'_{ji} is the maximum processing time of each job per stage.

5-2. Parameters tuning

Parameters tuning may impress the quality of the solutions (Lin and Ying 2013). So, to select the best factor combination; numbers of instances have been chosen randomly from the benchmark problem set for primary experiment. The following combinations of factors were tested on the selected instances: $Ab = 40, 50, 60, 70, 80$; $Cr = 0.1, 0.2, 0.3, 0.4, 0.5$; $Nit = 100, 200, 300$; $\alpha = 0.93, 0.95, 0.97$; and $T_0 = 70, 90, 130, 160$.

It is observed that when $PopAb$ or Cr is increased, a higher quality of solutions is obtained; but, computational time is raised significantly. In addition, by evaluating the effect of Nit, α, and T_0, it is observed that when their values are too high, more computational time is required to achieve good solutions; but, when Nit, α and T_0 values are too low due to the probability decrease of accepting worse solutions, the proposed algorithm converges prematurely.

Based on the primary testing, the following factors' values have the best performance within an acceptable computational time: $PopAb = 60$, $Cr = 0.2$ $Nit = 200$, $\alpha = 0.97$ and $T_0 = 90$. Hence, in this study, these values are applied to the experiments.

6. Computational Results and Discussion

To evaluate the performance of the proposed hybrid algorithm, the computational results of its implementation are compared with those obtained by implementing those algorithms which form the structure of the proposed hybrid algorithm, i.e., simulated annulling (SA) and artificial immune system (AIS).

All three algorithms are coded in MATLAB language and implemented on a PC with an Intel Core 2 Duo 2.20GHz CPU with 2GB of RAM. Relative Percentage Deviation (RPD) is obtained according to the following expression for each class of problems:

$$RPD = 100 \times \frac{Sol_{avg} - Sol_{best}}{Sol_{best}} \tag{17}$$

where Sol_{best} is the best achieved solution for each combination and Sol_{avg} is the average of solutions obtained from running algorithms in 5 repetition. RPD results are presented in Table 3. Bold numbers indicate the best RPD values in the class.

The obtained results demonstrate the superiority of AIS-SA in solving test problems of the HFS problem. The proposed AIS-SA algorithm achieves the lowest RPD value in 23 out of 36 classes of test problems, whereas AIS and SA only achieve the lowest RPD values in 3 and 6 classes of test problems, respectively. Moreover, to assess the computational efficiency of the methods under study, the required CPU time by each algorithm has been measured. The results are presented in Table 4 and, for brevity, they are classified according to the number of jobs, which is clearly the most effective factor influencing the computational time (Costa et al., 2014).

Although the results show that SA is faster than the other algorithms, it seems to be a less effective approach to solving the problem under investigation, as shown in Table 4.

Tab. 3. Comparison of RPD values for running the three algorithms: AIS−SA, AIS, and SA

$No.\ of\ combination$	N	I	M_i	B_{mi}	AIS − SA	AIS	SA
1	10	3	$U[1,3]$	$U[1,3]$	0.51	11.47	4.26
2	10	3	$U[1,3]$	$U[1,5]$	0.56	8.46	5.72
3	10	3	$U[1,5]$	$U[1,3]$	0.67	13.25	8.06
4	10	3	$U[1,5]$	$U[1,5]$	2.29	19.85	4.72
5	10	3	$U[1,10]$	$U[1,3]$	2.24	13.14	2.21
6	10	3	$U[1,10]$	$U[1,5]$	4.64	3.25	8.63
7	10	10	$U[1,3]$	$U[1,3]$	1.28	3.65	7.05
8	10	10	$U[1,3]$	$U[1,5]$	1.53	5.27	6.77
9	10	10	$U[1,5]$	$U[1,3]$	0.02	19.26	8.53
10	10	10	$U[1,5]$	$U[1,5]$	0.78	3.61	3.98
11	10	10	$U[1,10]$	$U[1,3]$	2.42	10.13	6.95
12	10	10	$U[1,10]$	$U[1,5]$	0.19	13.26	4.56
13	10	20	$U[1,3]$	$U[1,3]$	1.12	17.97	7.41
14	10	20	$U[1,3]$	$U[1,5]$	0.28	12.82	4.86
15	10	20	$U[1,5]$	$U[1,3]$	1.88	8.16	6.81
16	10	20	$U[1,5]$	$U[1,5]$	2.87	1.51	5.65
17	10	20	$U[1,10]$	$U[1,3]$	2.93	3.73	1.56
18	10	20	$U[1,10]$	$U[1,5]$	2.45	32.27	6.39
19	50	3	$U[1,3]$	$U[1,3]$	1.76	17.89	3.89
20	50	3	$U[1,3]$	$U[1,5]$	2.87	27.15	5.65
21	50	3	$U[1,5]$	$U[1,3]$	1.88	11.80	3.35
22	50	3	$U[1,5]$	$U[1,5]$	3.46	39.09	2.22
23	50	3	$U[1,10]$	$U[1,3]$	0.01	22.07	4.44
24	50	3	$U[1,10]$	$U[1,5]$	1.79	1.23	8.21
25	50	10	$U[1,3]$	$U[1,3]$	4.84	3.54	4.09
26	50	10	$U[1,3]$	$U[1,5]$	0.58	8.19	4.54
27	50	10	$U[1,5]$	$U[1,3]$	2.64	15.34	6.29
28	50	10	$U[1,5]$	$U[1,5]$	0.93	11.27	7.55
29	50	10	$U[1,10]$	$U[1,3]$	2.08	1.35	7.91
30	50	10	$U[1,10]$	$U[1,5]$	0.36	19.41	6.66
31	50	20	$U[1,3]$	$U[1,3]$	0.68	11.54	3.96
32	50	20	$U[1,3]$	$U[1,5]$	2.33	21.64	2.03

33	50	20	$U[1,5]$	$U[1,3]$	2.90	15.74	8.44
34	50	20	$U[1,5]$	$U[1,5]$	0.54	3.34	4.52
35	50	20	$U[1,10]$	$U[1,3]$	2.73	16.71	5.95
36	50	20	$U[1,10]$	$U[1,5]$	0.30	9.34	4.81
37	100	3	$U[1,3]$	$U[1,3]$	0.06	14.96	3.37
38	100	3	$U[1,3]$	$U[1,5]$	0.72	12.48	5.97
39	100	3	$U[1,5]$	$U[1,3]$	0.62	8.45	6.71
40	100	3	$U[1,5]$	$U[1,5]$	2.80	14.17	3.53
41	100	3	$U[1,10]$	$U[1,3]$	2.48	13.06	6.81
42	100	3	$U[1,10]$	$U[1,5]$	0.38	9.47	2.61
43	100	10	$U[1,3]$	$U[1,3]$	1.42	9.28	2.92
44	100	10	$U[1,3]$	$U[1,5]$	0.98	13.64	3.78
45	100	10	$U[1,5]$	$U[1,3]$	0.27	9.25	5.79
46	100	10	$U[1,5]$	$U[1,5]$	0.69	8.88	6.04
47	100	10	$U[1,10]$	$U[1,3]$	1.87	8.46	6.20
48	100	10	$U[1,10]$	$U[1,5]$	3.86	13.29	2.79
49	100	20	$U[1,3]$	$U[1,3]$	1.52	10.09	4.52
50	100	20	$U[1,3]$	$U[1,5]$	0.25	8.89	5.67
51	100	20	$U[1,5]$	$U[1,3]$	1.52	12.28	4.32
52	100	20	$U[1,5]$	$U[1,5]$	1.48	11.13	3.98
53	100	20	$U[1,10]$	$U[1,3]$	2.86	13.49	5.98
54	100	20	$U[1,10]$	$U[1,5]$	0.96	10.31	2.03
Average					1.59	12.20	5.21

Tab. 4. Average CPU time (in seconds)for test problems

N	AIS-SA	AIS	SA
10	33.91	35.84	29.05
50	153.62	144.31	127.82
100	479.48	467.19	402.01
Average	222.34	215.78	186.29

7. Conclusions

In this study, a hybrid flow shop scheduling problem with just-in-time objective function was investigated. It was supposed that a number of machines in some stations have the ability to perform a certain number of jobs simultaneously (parallel batching). In addition, each job in each station should be processed by a predetermined particular machine. A mixed integer linear programing model was developed for this problem. Since the HFS scheduling problem even with two stages and makespan criterion is NP-hard (Gupta 1988), the problem under consideration is also NP-hard.

Based on the just-in-time criterion, jobs must be delivered on time; otherwise, a penalty would be imposed on a manufacturing environment. Then, the two vector strings for each stage were proposed, not considered in previous works (Figure 2). Basic artificial immune system was combined with the annealing procedure of SA algorithms to overcome the deficiencies of each algorithm separately and increase their exploitation and exploration capabilities in the reasonable computation time. A hybrid artificial immune system (AIS-SA) algorithm was employed for solving a combination of well-known benchmark problems in the literature that were randomly generated (Table 2). To evaluate the performance of the AIS-SA, the results of its implementation were compared with those obtained by implementing the algorithms that were contained in the structure of the proposed hybrid algorithm (i.e., AIS and SA). The results display that AIS-SA outperforms the two other algorithms (Tables 3 & 4).

References

[1] Allaoui, H., and A. Artiba, "*Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints.*" Computers and Industrial Engineering Vol. 47, No. 4, (2004), PP. 431–450.

[2] Amin-Naseri, M. R., and M. A. Beheshti-Nia, "*Hybrid flow shop scheduling with parallel batching.*" Int. J. Production Economics, (2009), PP. 185–196.

[3] Babayan, A., and D. He.,"*Solving the n-job 3-stage flexible flow shop scheduling problem using an agent-based approach.*" International Journal of Production Research Vol. 42, No. 4, (2004), PP. 777–799.

[4] Behnamian, J., S. M. Ghomi, and M. Zandieh., "*Hybrid flow shop scheduling with sequence-dependent setup times by hybridizing max–min ant system, simulated annealing and variable neighbourhood search.*" Expert Systems . (2012), PP. 613-627.

[5] Bertel, S., and J.-C. Billaut., "*A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation.*" European Journal of Operational Research, (2004), PP. 651–662.

[6] Choong, F., S. Phon-Amnuaisuk, and M. Alias., "*Metaheuristic methods in hybrid flow shop scheduling problem.*" Expert Systems with Applications, (2011), PP. 10787–10793.

[7] Chunga, T.-P., and C.-J. Liao., "*An immunoglobulin-based artificial immune system for solving the hybrid flow shop problem.*" Applied Soft Computing, (2013), PP. 3729–3736.

[8] Costa, A., F. A. Cappadonna, and S. Fichera., "*A novel genetic algorithm for the hybrid flow shop scheduling with parallel batching and eligibility constraints.*" International Journal of Advanced Manufacturing Technology, (2014), PP. 781-796.

[9] Dai, M., Dunbing Tang, Adriana Giret, Miguel A. Salido, and W.D. Li., "*Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm.*" Robotics and Computer-Integrated Manufacturing, (2013), PP. 418–429.

[10] Engin, O., and A. Doyen., "*A new approach to solve hybrid flow shop scheduling problems by artificial immune system.*" Future Generation Computer Systems, Vol. 20, No. 6, (2004), PP. 1083–1095.

[11] Fanti, M.P., B. Maione, G. Piscitelli, and B. Turchiano.,"*Heuristic scheduling of jobs on a multi-product batch processing machine.*" International Journal of Production Research, Vol. 34, No. 8, (1996), PP. 2163–2186.

[12] Graham, R.L., E.L. Lawler, and A.H.G. RinnooyKan., "*Optimization and approximation in deterministic sequencing and scheduling: a survey.*" Annals of Discrete Mathematics, Vol. 5, (1979), PP. 287–326.

[13] Gravel, M., W. Price, and C. Gagne., "*Scheduling jobs in an aluminum foundry using a genetic algorithm.*" International Journal of Production Research, Vol. 38, No. 13, (2000). PP. 3031–3041.

[14] Gupta, J.N.D., "*Two stage hybrid flow shop scheduling problem.*" Journal of Operational Research Society, Vol. 39, No. 4, (1988), PP. 359–364.

[15] Hart, E., and J. Timmis., "*Application areas of AIS: the past, the present and the future*." Applied Soft Computing, Vol. 8, No. 1, (2008), PP.191–201.

[16] Hoogeveen, J., J. Lenstra, and B. Veltman., "*Preemptive scheduling in a two-stage multiprocessor flow shop is strongly NP-hard.*" European Journal of Operational Research, Vol. 89, (1996), PP. 172–175.

[17] Huang, K.L., and C.J. Liao., "*Ant colony optimization combined with tabu search for the job shop scheduling problem.*" Computers and Operations Research, Vol. 35, No. 10, (2006), PP. 30–46.

[18] Jin, Z.H., Z. Yang, and T. Ito., "*Metaheuristic algorithms for the multistage hybrid flow shop scheduling problem.*" International Journal of Production Economics, Vol. 100, No. 2, (2006), PP. 322–334.

[19] Jungwattanakit, J., M. Reodecha, P. Chaovalitwongse, and F. Werner., "*Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria.*" International Journal of Advanced Manufacturing Technology, Vol. 37, No. 3–4, (2008), PP. 354–370.

[20] Khalouli, S., F. Ghedjati, and A. Hamzaoui., "*A meta-heuristic approach to solve a JIT scheduling problem in hybrid flow shop.*" Engineering Applications of Artificial Intelligence, (2010), PP. 765–771.

[21] Kis, T., and E. Pesch., "*A review of exact solution methods for the non-preemptive multiprocessor flow shop problem.*" European Journal of Operational Research, Vol. 164, No. 3, (2005), PP. 592–608.

[22] Kleeman, M.P., and G.B. Lamont., "*Scheduling of Flow-Shop, Job-Shop, and Combined Scheduling Problems using MOEAs with Fixed and Variable Length Chromosomes.*" Studies in Computational Intelligence (SCI),Vol. 49, (2007), PP. 49–99.

[23] Lin, S.W., and K.C. Ying., "*Minimizing makespan in a blocking flow shop using a revised artificial immune system algorithm.*" Omega, Vol. 41, (2013), PP. 383–389.

[24] Linn, R., and W. Zhang., "*Hybrid flow shop scheduling: a survey.*" Computers and Industrial Engineering, Vol. 37, No. (1–2), (1999), PP. 57–61.

[25] M'Hallah, R., "*Minimizing total earliness and tardiness on a single machine using a hybrid heuristic.*" Computers and Operations Research, Vol. 34, (2007), PP. 3126–3142.

[26] Mathirajan, M., A.I. Sivakumar, and V. Chandru., "*Scheduling algorithms for heterogeneous batch processors with incompatible job families.*" Journal of Intelligent Manufacturing, Vol. 15, (2004), PP. 787–803.

[27] Mirsanei, H.S., M. Zandieh, M.J. Moayed, and M.R. Khabbazi., "*A simulated annealing approach to hybrid flow shop scheduling with sequence-dependent setup times.*" J Intell Manuf, Vol. 22, (2011), PP. 965–978.

[28] Naderi, B., M. Zandieh, A. Khaleghi Ghoshe Balagh, and Roshanaei V., "*An improved simulated annealing for hybrid flow shops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness.*" Expert Systems with Applications, Vol. 36, No. 6, (2009), PP. 9625–9633.

[29] Oulamara, A., "*Makespan minimization in a no-wait flow shop problem with two batching machines.*" Computers and Operations Research, Vol. 34, No. 4, (2007), PP. 1033–1050.

[30] Pinedo, M. L., *Scheduling: Theory, Algorithms, and Systems.* New York: Springer Science, (2008).

[31] Portmann, M.C., and Z. Mouloua., "*A window time negotiation approach at the scheduling level inside supply chains.*" Proceedings of the 3th Multi- disciplinary International Scheduling Conference:

Theory and Application (MISTA 2007). Paris. (2007), PP. 410–417.

[32] Potts, C. N., and M. Y. Kovalyov., "*Scheduling with batching: A review.*" European Journal of Operational Research, (2000), PP. 228-249.

[33] Quadt, D., and D. Kuhn., "*A taxonomy of flexible flow line scheduling procedures.*" European Journal of Operational Research, Vol. 178, No. 3, (2007), PP. 686–698.

[34] Rajendran, C., and D. Chaundhuri., "*A multi-stage parallel processor flow shop problem with minimum flow time.*" Eur J Oper Res, Vol. 57, (1992), PP. 111–122.

[35] Ribas, I., R. Leisten, and J. M.Framinan., "*Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective.*" Computers & Operations Research, (2010), PP. 1439–1454.

[36] Ruiz, R., and C. Maroto., "*A genetic algorithm for hybrid flow shops with sequence dependent setup times and machine eligibility.*" European Journal of Operational Research, Vol. 169, No. 3, (2006), PP. 781–800.

[37] Ruiz, R., and J.A. Vázquez-Rodríguez., "*The hybrid flow shop scheduling problem.*" European Journal of Operational Research,Vol. 205, (2010), PP. 1–18.

[38] Ruiz, R., F.S. Serifoglu, and T. Urlings., "*Modeling realistic hybrid flexible flow shop scheduling problems.*" Computers and Operations Research, Vol. 35, No. 4, (2008), PP. 1151–1175.

[39] Sawik, T.J., "*An exact approach for batch scheduling in flexible flow lines with limited intermediate buffers.*" Mathematical and Computer Modeling, Vol. 36, No. 4–5, (2002), PP. 461–471.

[40] Sherali, HD, SC Sarin, and MS. Kodialam., "*Models and algorithms for a two-stage production process.*" Prod Plann Contr, Vol. 1, No. 1, (1990), PP. 27–39.

[41] T'kindt, V., and J.C. Billaut., *Multicriteria Scheduling: Theory, Models and Algorithms.* Berlin: Springer, (2005).

[42] Tadayon, B., and N. Salmasi., "*A two-criteria objective function flexible flow shop scheduling problem with machine eligibility constraint.*" International Journal of Advance Manufacturing Technology, (2013), PP. 1001–1015.

[43] Wang, W.,"*Review on Hybrid Flow Shop Scheduling.*" International Conference of Information Technology, Computer Engineering and Management Sciences. IEEE, (2011).

[44] Yaghubian, A. R., T. Hodgson, J. Joines, T. Culberth, and J. Huang., "*Dry kiln scheduling in furniture production.*" IIE Transactions, Vol. 31, (1999), PP. 733–738.

[45] Yaurima, V., L. Burtseva, and A. Tchernykh., "*Hybrid flow shop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers.*" Computers and Industrial Engineering, Vol. 56, No. 4, (2009), PP. 1452–1463.

[46] Zee DJ, van der, A. Van Harten, and P. C. Schuur., "*Dynamic job assignment heuristics for multi-server batch operations– A cost based approach.*" International Journal of Production Research, Vol. 35, No. 11, (1997), PP. 3063–3093.

[47] Zhang, C.Y., P. Li, Y. Rao, and Guan Z., "*A very fast TS/SA algorithm for the job shop scheduling problem.*" Computers and Operations Research, Vol. 35, No. 2, (2008), PP. 82–94.

[48] Zhang, R., and C. Wu., "*A hybrid immune simulated annealing algorithm for the job shop scheduling problem.*" Applied Soft Computing, (2010), PP. 79–89.

[49] Ziaeifar, A., R. Tavakkoli-Moghaddam, and K. Pichka., "*Solving a new mathematical model for a hybrid flow shop scheduling problem with a processor assignment by a genetic algorithm.*" International Journal of Advanced Manufacturing Technology, Vol. 61, (2012), PP. 339–349.

[50] Zitzler, E., and L. Thiele., "*Multiobjective evolutionary algorithms: a comparative case study and strength Pareto approach.*" IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, (1999), PP. 257–271.

Follow This Article at The Following Site

Rezaeian J, Shafipour M. Hybrid artificial immune system and simulated annealing algorithms for solving hybrid JIT flow shop with parallel batches and machine eligibility . IJIEPR. 2017; 28 (3) :251-266
DOI: 10.22068/ijiepr.28.3.251
URL: http://ijiepr.iust.ac.ir/article-1-734-en.html