Industrial Engineering & Management
ISSN: 2169-0316

Rahmanidoust et al., Ind Eng Manage 2017, 6:3
DOI: 10.4172/2169-0316.1000225

Research Article Open Access

An Effective Harmony Search Algorithm for Solving a No-Wait Hybrid Flow Shop Scheduling Problem with Machine Availability Constraint

Mohammad Rahmanidoust[1]*, Jianguo Zheng[1] and Meysam Rabiee[2]

[1]Glorious Sun School of Business and Management, Donghua University, PR China
[2]Department of Industrial Engineering, Bu-Ali Sina University, Hamedan, Iran

Abstract

This research investigates a no-wait hybrid flow shop scheduling problem. Minimizing the mean tardiness is considered as the objective to develop the optimal scheduling algorithm. Characteristics of our considered problem leads to the complexity of problem. First, no-wait operations. Second, setup time of each job is separated from its processing time and depends upon its preceding job. Third, all of jobs aren't available at the first of scheduling. In other word, each job has individual ready time. Finally, machines are not continuously available due to the preventive maintenance. An effective harmony search algorithm is used to tackle the mentioned problem. A series of computational experiments is conducted by comparing our algorithm with previous meta-heuristic algorithms like population based simulated annealing (PBSA), Adopted imperialist competitive algorithm (ICA) and hybridization of PBSA and ICA (ICA+PBSA). To achieve reliable results, Taguchi approach is used to define robust parameters' values for our proposed algorithm. The computational results with random test problems suggest that our proposed harmony search outperforms the three foregoing algorithms.

Keywords: Scheduling; No-wait; Flow shop; Harmony search; Taguchi

Notations

n: The number of jobs to be scheduled (j=1, 2,.., n)

m^i: The number of parallel machines at stage i

p_{iu}^j : Processing time for job j at stage i (i=1, 2,.., s) on u^{th} machine

d_j: Due date for job j

$S_{k,j}^i$: Sequence-dependent setup time from job k to job j at stage i

π: Permutation of the given jobs (π={ π_1, π_2,..., π_n})

MA_{ui}: Machine availability time for uth machine at stage i

C_j: Completion time of job j

Tj: Tardiness for job j (Tj=max(0, C_j - dj))

$\overline{T}$: Mean tardiness $(\overline{T} = (\sum_{j=1}^{n} T_j)/n)$

NP-Hard: Non-deterministic polynomial-time Hard.

Introduction

Production scheduling is one of the prominent decision-making process in the operation level of each manufacture or service companies. It can be defined as sequencing of a number of jobs on one or several machines aiming to optimally utilizing the resources while meeting the customer's demands in an efficient manner. Such a frequently occurring scheduling problem is difficult to solve due its complex nature. In recent years, researchers have focus in solving new challenges of machine scheduling problems [1-7].

One of the very noticeable process to make decision is the planning for production in service companies. This is actually a classification of jobs for one or more machines to best use of the capabilities in operation and reach the customer's satisfaction at the same time. This planning problem that is often occurring, is interacted to solve since the environment is so [8].

One of the most applicable problems in scheduling area in both theory and practice is flexible flow shop (FFS), or a hybrid flow shop (HFS), or a flow shop with multiple processors (FSMP). A typical FFS problem can be defined as follows: there are N jobs passing through a K stage flow line with one or more parallel machines at each stage. No-wait flow shop and flexible flow shop scheduling problem has been studied by many researchers [9-19]. For a literature review in this area the readers are referred to those conducted by Richard and Zhang [20], Ruiz et al. [21], and Ribas et al. [22],

Classical flexible flow shop scheduling problems assume that there is unlimited intermediate storage available to work in process (WIP) jobs between two adjacent stages. In a particular case of the FFL, there is no longer any need for intermediate storage or blocking between stages. The operations of all jobs have to be processed from start to finish without interruptions either on or between stages. i.e., if necessary, the start of a job on a given machine must be delayed so that the completion of the operation coincides with the beginning of the operation on the following machine. These conditions are quite common in several industries. In some industries, due to the temperature or other characteristics of the material it is required that each operation follow the previous one immediately. Such situations appear in the chemical processing [23], food processing [24], concrete ware production [25], pharmaceutical processing [26] and production of steel, plastics, and aluminum products [27]. For instance, in the steel-making and continuous casting processes of iron and steel manufacturing enterprises, a no-wait scheduling can reduce the energy loss of high-temperature molten steel and plays an important role in

***Corresponding author:** Mohammad Rahmanidoust, Glorious Sun School of Business and Management, Donghua University, PR China, Tel: +989121366482; E-mail: rahmanidoust@hotmail.com

Received June 13, 2017; **Accepted** June 14, 2017; **Published** August 11, 2017

Citation: Rahmanidoust M, Zheng J, Rabiee M (2017) An Effective Harmony Search Algorithm for Solving a No-Wait Hybrid Flow Shop Scheduling Problem with Machine Availability Constraint. Ind Eng Manage 6: 225. doi:10.4172/2169-0316.1000225

realizing the advanced production style of HCR/DHCR [28]. For a further study in no wait manufacturing the readers are recommended to review the paper present by Hall and Sriskandarajah [26].

In the literature, the no-wait flow shop scheduling problem has attracted many studies since 1964 [29-44]. No-wait flow shop scheduling problem is a typical scheduling problem with strong engineering background.

Gilmore and Gomory [29] presented an algorithm that can solve a restricted version of the travelling salesman problem (TSP). As the no-wait two-machine flow shop problem of mean completion time (MCT) minimization can be formulated as the restricted TSP, the Gilmore and Gomory algorithm can be used to solve the problem in polynomial time. Gupta et al. [45] showed that the problem can be reduced to the Gilmore–Gomory TSP and can be solved in polynomial time. Cheng et al. [30] considered the problem of one-operator two-machine flow-shop scheduling with setup and dismounting times to minimize MCT. The no-wait two-machine flow-shop scheduling problem with separate sequence-independent setup times was addressed by Aldowaisan and Allahverdi [46] with the objective of minimizing total completion time. They developed optimal solutions for certain cases, established a local dominance relation for the general case, and proposed a simple but effective heuristic. The same problem was studied by Aldowaisan [47] where a global dominance relation along with a heuristic and a branch-and-bound method was provided.

The algorithm presented by Gilmore and Gomory [29], can answer to the salesman travelling question. When the no-wait two-machine flow shop problem of mean completion time (MCT) is shown as salesman travelling question, the abovementioned algorithm can be answered in several steps. Also to decrease MCT, Cheng et al. uses special system and descending time. The entire accomplishment time is the goal of Aldowaisan and Allahverdi. They applied discrete sequence-independent arrangement times to come up with the no-wait two-machine flow-shop scheduling problem. They made the best answer for some issues, for the general cases they based local supremacy relation, and finally suggested a modest and operative empirical method. Another study by Aldowaisan was proposed to answer the same issue that a global dominance relation accompanied with an empirical and branch-and-bound method was created.

The two-machine no-wait flow-shop separate setup time problem with the objective of minimizing MCT is also addressed in the literature. Sidney et al. [48] consider the same problem but where the setup on the second machine consists of two parts. Nagano and Araújo [42] addressed the problem of scheduling jobs in a no-wait flow-shop with sequence-dependent setup times with the objective of minimizing the makespan and the total flow time. They presented two new constructive heuristics to obtain good approximate solutions for the problem in a short CPU time, named GAPH and QUARTS. Samarghandi and ElMekkawy [44] developed a mathematical model of the problem and the problem was reduced to a permutation problem. A straightforward algorithm for calculating the makespan of the permutation of jobs was developed. A particle swarm optimization (PSO) was applied on the encoded sequences for exploration of the solution space. Computational results on the available test problems revealed the efficiency of the PSO in finding good-quality solutions.

In real manufacturing systems, many assumptions could be considered but in most of the studies in this area, investigators tried to solve the problems with a few practical assumptions. In contrast to the existence of many research results on the no-wait flexible flow shop scheduling, there have been few attempts to study scheduling problems that involve sequence dependent setup time, minimizing tardiness, unrelated parallel machine and machine availability in no-wait flexible flow shop simultaneously. Some of close researches to our studied problem have discussed as follows.

Liu et al. [49] presented a heuristic algorithm named Least Deviation (LD) algorithm for two-stage no-wait hybrid flow shop scheduling with a single machine in either stage. The performance measure used in this study is makespan. The results showed that LD algorithm outperforms the others in most practical cases. In addition the proposed algorithms showed low computational complexity and easy to implement, thus it is favourable application value.

Xie et al. [50] proposed a new heuristic algorithm known as Minimum Deviation Algorithm (MDA) to minimize makespan in a two stage flexible flow shop with no waiting time. Experimental results of the study showed that MDA outperforms partition method, partition method with LPT, Johnson's and modified Johnson's algorithms. Huang et al. [51] considered a no-wait two stage flexible flow shop with setup times and with minimum total completion time performance measure. They proposed an integer programming model and Ant Colony Optimization heuristic approach. The results revealed that the efficiency of the proposed algorithm is superior to those solved by integer programming while having satisfactory solutions. Jolai et al. [52] introduced no-wait flexible flow line scheduling problem with time windows and job rejection to maximizing profit. This is an extension of production and delivery scheduling problem with time windows. They also presented a mixed integer-linear programming model and genetic algorithm procedures to solve their model efficiently. Comparison of the results obtained by GA with LINGO solutions and Tabu search showed that the proposed GA obtains better solutions in a very low computational time in comparison with the solutions obtained from LINGO optimization software. Jolai et al. [53] introduced a novel hybrid meta-heuristic algorithm to solve a no-wait flexible flow shop scheduling problem with sequence-dependent setup times to minimize the maximum completion time. They proposed three novel meta-heuristic algorithms, namely Population Based Simulated Annealing (PBSA), Adapted Imperialist Competitive Algorithm (AICA) and hybridization of adapted imperialist competitive algorithm and population based simulated annealing (AICA+PBSA) to solve the addressed problem. The computational evaluations of their study manifestly support the high performance of our proposed novel hybrid algorithm against other algorithms which were applied in literature for related production scheduling problems. Rabiee et al. [54] addressed the problem of no-wait two stage flexible flow shop scheduling problem with respect to unrelated parallel machines, sequence-dependent setup times, probable reworks and different ready times to actualize the problem. They proposed an intelligent hybrid meta-heuristic which was based on imperialist competitive algorithm (ICA), simulated annealing (SA), variable neighbourhood search (VNS) and genetic algorithm (GA) for solving the mentioned problem. The results of their study revealed the relative superiority of proposed algorithm. Ramezani et al. [39] dealt with a no-wait scheduling problem considering anticipatory sequence-dependent setup times on the flexible flow shop environment with uniform parallel machines to minimize maximum completion time of jobs. Since this problem was known to be NP-hard, they introduced a novel approach to tackle the problem. They proposed a hybrid meta-heuristic which involved invasive weed optimization, variable neighbourhood search and simulated annealing to tackle of the problem. The experimental results of their research revealed the superiority of the performance of the hybrid meta-heuristic in

comparison with original ones singularly. Asefi et al. [55] proposed a hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flow shop scheduling problem. They compared proposed algorithm with NSGA-II and SPEA-II and revealed that their algorithm can achieve reliable and better results. Khalili and Naderi [56] proposed a novel bi-objective imperialist competitive algorithm to solve a no-wait flexible flow shop scheduling problem with sequence dependent setup times.

To the best of our knowledge, the mean tardiness minimization in no-wait flexible flow shop scheduling problem with sequence dependent setup times, unrelated parallel machine, ready time and machine availability constraint has not been studied yet. Considering the importance of the no-wait flexible flow shop problem and the fact that the no-wait flexible flow shop problem with mentioned constraints and assumptions has not been given much heed by researchers, here we have offered a new effective harmony search algorithm for solving it.

The outline of the paper is as follows: problem definition is presented in section 2. Section 3 explains the proposed harmony search to solve the considered problem. Computational experiments as well as parameter tuning are provided in section 4. Finally, section 5 is devoted to conclusion remarks and future researches.

Problem Definition

In this section, first the notations which are used in this research are defined then the assumptions of the studied problem are elaborated.

Assumptions

The problem under study here involves processing of a set of n jobs ($j=\{j_1, j_2,..., j_n\}$). It is needed that these jobs are processed in k consecutive stages. The number of parallel machines in the k_{th} stage is M_k and the processing time of the j_{th} job in the u_{th} machine at i_{th} stage is p_{iu}^{j}. The sequence dependent setup time between j_{th} and L_{th} job in j_{th} stage is shown as. Our problem there is finding the sequence of jobs with minimum average delay ($\overline{T}$). Here are some assumptions of the problem:

- All the data used here for the study of the problem are known deterministically.
- Once a job began on a machine, it must proceed to completion without interruption. That is, once a job is commenced on the first machine, it must proceed through all machines without any occlusion or interruption.
- Each stage has at least one machine, and there is at least one stage which has more than one machine.
- A machine can process only one job at a time.
- Travel times between stages are negligible.
- Each job is assigned to every machine one at a time and no machines are twice occupied by the same job.
- To processing of each job some of machines are available due to machine eligibility an. There is no breakdown or s machine availability constraint.
- The release time of all jobs are different, meaning that each job can be processed after release time and it can't be processed before its ready time.

Setup times depend on sequencing of jobs which means setup times are sequence dependent and the length of time required to do the setup depends on the prior and current jobs and the machine which is to do the processing in the mentioned stage ($S_{jl}^{\ i}$).

Machine in all stages are non-identical which means speed of each machine relatively is different. In this case, processors work in parallel and speed of processing time of job i at stage t uniformly differs depends on relative speed of $v_{i,j}^{t}$ (see eqn. (1)).

$$p_{i,j}^{t}=\frac{p_i^t}{v_{i,j}^{t}}\ \ i=1,..,n\ j=1,\ldots,mt\ t=1,\ldots.g \tag{1}$$

The problem of no wait flexible flow shop is shown by *FFS* ($QM^{(1)},..., QM^{(m)}$) / no-wait, SDST, $r_{j,}\ M_j$/**T** and formally defined in the following. The problem is processing of n jobs $\{J_1, J_2, J_3,\ldots, J_n\}$ on a series of stages {1, 2,..., *t*,…, *k*}which at each stage there are *mt* machines. Scheduling of the addressed problem comprises three sub-problems: At first, the problem is finding a sequence which minimizes the average tardiness. Second issue which has to be taken into account is machine assignment. Thirdly, minimum starting times must be determined in a way that all of the no-wait constraints are satisfied. The no-wait constraint requires that the starting time of job J*j* at stage *t* be equal to the completion time of job J*j* at stage *t*-1 for each *i* and *t*.

$$C_{\pi_{[1]}}^{1}=\min_{1\le i\le m^t}\left\{p_{\pi_{[1],i}}^{1}+S_{\pi_{[0],\pi_{[1],i}}}^{1}\right\} \tag{2}$$

$$C_{\pi_{[1]}}^{1}=C_{\pi_{[1]}}^{t-1}+\min_{1\le j\le m^t}\left\{p_{\pi_{[1],j}}^{t}+S_{\pi_{[10],\pi_{[1]^j}}}^{t}\right\} \tag{3}$$

$$C_{\pi_{[1]}}^{1}=C_{\pi_{[J]}}^{1}+F_{\pi_{[J]}}\sum_{l=2}^{t}\min_{1\le j\le m^l}\left\{p_{\pi_{[J],j}}^{l}+S_{\pi_{[J-1],\pi_{[J]^j}}}^{l}\right\} \tag{4}$$

Where,

$$F_{\pi_{[i]}}=\max\left\{0,\min_{2\le i\le m^t}\left(C_{\pi_{[J-1]}}^{t}-\left(C_{\pi_{[J-1]}}^{1}+\sum_{l=1}^{t-1}\min_{1\le i\le m^l}\left\{p_{\pi_{[J],i}}^{l}+S_{\pi_{[J-1],\pi_{[J]^i}}}^{l}\right\}\right)\right)\right\}\ j=2,\ldots,n \tag{5}$$

Regarding above mentioned relations, completion time of a job is equal to completion time of that job at the final stage:

$$C_{\pi_{[J]}}=C_{\pi_{[J]}}^{k} \tag{6}$$

And as mentioned earlier, the makespan of the scheduling corresponding to the given sequence of jobs is calculated as follows:

$$T=\max\left\{0,\left(C_{\pi_{[J]}}^{g}-d_j\right)\right\} \tag{7}$$

$$\overline{T}=\frac{\sum_{j=1}^{n}T_j}{N} \tag{8}$$

Herein, the goal of the problem of no-wait SDST flexible flow shop with different ready time and machine availability is to find an optimal sequence which can minimize the mean tardiness of scheduling.

In this formula, the objective is to find out the best order that can highly reduce the lateness of planning for the problem of no-wait SDST flexible flow shop with variety of ready time and machine readiness.

No-wait two-stage flexible flow shop problems are NP-hard in the strong sense. So, the no-wait k-stage flexible flow shop problem is NP-hard, too. Hence, all exact approaches for even simple problems will most likely have running times that increase exponentially with the problem size. In this paper a novel advanced meta-heuristic algorithm (effective harmony search) is suggested for solving the problem described above. This problem considered in this study is schematically depicted in Figure 1. Also, the framework of this algorithm is elucidated in the next section.

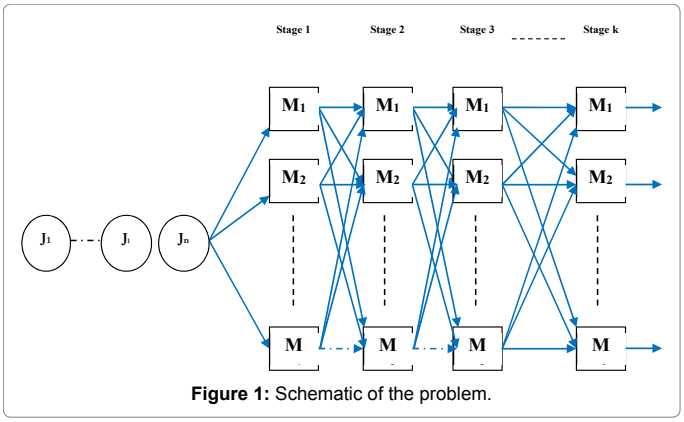

Figure 1: Schematic of the problem.

No-wait two-stage flexible flow shop problems are NP-hard in the strong sense. Hence the same is for the no-wait k-stage flexible flow shop problem, i, e; the NP-hard too. So, any possible precise methods for even simple problems probably have process times that rapidly grow as the size of problem grows. This study attempts to suggest a new advanced meta-heuristic algorithm (effective harmony search) to solve the abovementioned problem.

Hybrid Harmony Search (HHS)

Recently, the meta-heuristics have become quite popular over the other approximate, exact or heuristic methods for solving complex combinatorial optimization problems such as job shop, flow shop scheduling problem and too many other hard problems [57-60]. In this paper, a new algorithm called 'hybrid algorithm' (HA)" is proposed to solve the described problem.

Harmony search is a music-based metaheuristic optimization algorithm. It was inspired by the observation that the aim of music is to search for a perfect state of harmony. The effort to find the harmony in music is analogous to find the optimality in an optimization process. A musician always intends to produce a piece of music with perfect harmony. On the other hand, an optimal solution to an optimization problem should be the best solution available to the problem under the given objectives and limited by constraints. Both processes intend to produce the best or optimum. In order to explain the Harmony Search in more detail, let us first idealize the improvisation process by a skilled musician. When a musician is improvising, he or she has three possible choices: (1) playing any famous tune exactly from his or her memory; (2) playing something similar to the aforementioned tune (thus adjusting the pitch slightly); or (3) composing new or random notes.

Geem et al. [61] formalized these three options into quantitative optimization process, and the three corresponding components become: usage of harmony memory, pitch adjusting, and randomization.

Searching for harmony which is a subject in music, is an algorithm for metaheuristic optimization. The criteria were recognized when the amazing state of music harmony was searched as a goal. In the process of harmony search in music, the corresponding note is being attempted to find. The composer aims to find it when the best result is recognized. So the objective is to find optimum or the best solution for our subjected optimization problem and this is surely the best possible solution considering the goals, limits and capabilities. So what really matters in both areas is, "the best" or "optimum". To elaborate the process of Harmony search, let's simulate what an expert composer does. When he is improvising, he goes through 3 different choices. One is, copying what he has in mind from a famous tune. Two is, simulate a tune that is similar to that and three is, producing a tune by chance. These are what Geem et al. [61] refers to as usage of harmony memory, pitch adjusting, and randomization.

The steps in the procedure of harmony search are as follows [62]:

Step 1. Initialize the problem and algorithm parameters.

Step 2. Initialize the harmony memory.

Step 3. Improvise a new harmony.

Step 4. Update the harmony memory.

Step 5. Check the stopping criterion.

These steps are described in the next subsections.

Algorithm's parameters

Before defining the steps of proposed hybrid harmony search the parameters of this algorithm should be introduced. The HS algorithm parameters are also specified in this step. These are the harmony memory size (HMS), or the number of solution vectors in the harmony memory; harmony memory considering rate (HMCR); pitch adjusting rate (PAR); and the number of improvisations (NI), or stopping criterion. The harmony memory (HM) is a memory location where all the solution vectors (sets of decision variables) are stored. This HM is similar to the genetic pool in the GA [63]. Here, HMCR and PAR are parameters that are used to improve the solution vector.

As the first step, the factors of suggested hybrid harmony search are elaborated to define the details of it. Also, the HS algorithm parameters are stated here. They're referred as Harmony memory size (HMS) or the quantity of possible solution path for that harmony memory. Harmony memory considering rate (HMCR); pitch adjusting rate (PAR); and the number of improvisations (NI), or stopping criterion. All solution paths or collection of choices are stored in Harmony Memory. HM corresponds to what we already mentioned as genetic pool in the GA. Both HMCR and PAR are the factors that matter to develop the solution path bank.

Harmony memory initialization and evaluation

The HM matrix is filled with as many randomly generated solution vectors as the HMS. This matrix has N columns where N is the total number of decision variables and HMS rows which are selected in the first step. This initial memory is created by assigning random values that lie inside the lower and upper bounds of the decision variable to each decision parameter of each vector of the memory as shown in eqn. (10). An initial population of harmony vectors are randomly generated and stored in a harmony memory (HM). Then a new candidate harmony is generated from all of the solutions in HM by using a memory consideration, a pitch adjustment, and a random selection.

The HM matrix is accumulated with maximum number of randomly made solution paths. It has N columns where N is the sum of choices and HMS rows that are selected in the first step. This primarily memory is shaped via allocating random values that are from upper and lower boundaries of choices to every single decision factor and parts of memory as shown in eqn. (10). A pool from harmony paths are randomly created and stored in HM. Using a memory status, an adjustment in pitch and a random choice, another harmony variable is created among all the possible choices in HM.

$$x_{i,j}^{0} = x_j^{\min} + r_j \times \left(x_j^{\max} - x_j^{\min} \right) \tag{9}$$

Where $x_j^{\min}$ and $x_j^{\max}$ are the lower and upper bound of the j_{th} decision parameter respectively, and $r_j \in [0,1]$ is an uniformly distributed random number generated anew for each value of j. Pseudo code of memory initialization can be shown in Figure 2. Thus, HM form can be shown as in Figure 3. Candidate solution vectors in HM shown in Figure 3 are then analyzed and their objective function values are calculated.

$(f(x_i), 1,2,..., HMS.$

It should be noted that to use HS algorithm for solving the mentioned scheduling problem, it is necessary to convert it to a job permutation for evaluating the objective value. Let each index of the dimensions of the vector represent a typical job from J={1, 2,..., n}, and the n indexes denote n different jobs. Thereafter, the largest position value (LPV) rule is employed to obtain a job permutation p={p(1),p(2),..., p(n)} by ordering the jobs in their non-increasing position value of Xi. A simple example is illustrated in Figure 4.

Improvise a new harmony

In this step, a New Harmony vector $x_i' = (x_{i,1}', x_{i,2}', \ldots x_{i,n}')$ is generated based on three rules. They are memory consideration, pitch adjustment, and random selection. The value of a design variable can be selected from the values stored in HM with a probability of harmony memory considering rate (HMCR). It can be further adjusted by moving to a neighbor value of a selected value from the HM with a probability of pitch adjusting rate (PAR). Or, it can be selected randomly from the set of all candidate values without considering the stored values in HM, with the probability of (1-HMCR).

```
for i = 1 to HMS
    for j = 1 to decision variables
        x^0_{i,j} = x^min_j + r_j × (x^max_j − x^min_j)
    end
end
```

Figure 2: Memory initialization.

$$HM = \begin{bmatrix} x_1^1 & x_2^1 & \ldots & x_n^1 & | & f(x^1) \\ x_1^2 & x_2^2 & \ldots & x_n^2 & | & f(x^2) \\ \ldots. & \ldots & \ldots & \ldots & | & \ldots \\ x_1^{HMS} & x_2^{HMS} & \ldots & x_n^{HMS} & | & f(x^{HMS}) \end{bmatrix}$$

Figure 3: HM form.

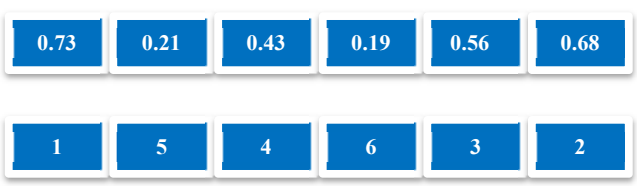

Figure 4: An example for the LPV rule.

Memory consideration: The usage of harmony memory (HM) is important because it ensures that good harmonies are considered as elements of new solution vectors. In order to use this memory effectively, the HS algorithm adopts a parameter HMCR ∈ [0,1], called harmony memory considering (or accepting) rate. If this rate is too low, only few elite harmonies are selected and it may converge too slowly. If this rate is extremely high (near 1), the pitches in the harmony memory are mostly used, and other ones are not explored well, leading not into good solutions. Therefore, based on researchers suggestion HMCR is considered between 0.7 and 0.95. We consider a linear relation for HMCR so that it has bigger value at first iteration and lower value at the end of iterations.

Using HM drastically matters since it guarantees the best harmonies are purified for components of the selected solution. To meritoriously apply the memory, HS process implements a factor HMCR ∈ [0,1], that is known as rate of harmony memory accepting. In case the best and few harmonies are selected, this rate is too low and the convergence is very slow. On the other hand, when the rate is tremendously high, it means the pitches in harmony memory are almost fully working, the other ones are not working well and hence they're not led to proper solution. As a result, the investigators suggest 0.7 and 0.95 as the range of HMCR. Also since the relation of it is linear, at the first iteration, it has higher value and it incrementally decrease till the end of iterations that has low value.

Memory consideration: The usage of harmony memory (HM) is important because it ensures that good harmonies are considered as elements of new solution vectors. In order to use this memory effectively, the HS algorithm adopts a parameter HMCR ∈ [0,1], called harmony memory considering (or accepting) rate.

If this rate is too low, only few elite harmonies are selected and it may converge too slowly. If this rate is extremely high (near 1), the pitches in the harmony memory are mostly used, and other ones are not explored well, leading not into good solutions. Therefore, based on researchers suggestion HMCR is considered between 0.7 and 0.95. We consider a linear relation for HMCR so that it has bigger value at first iteration and lower value at the end of iterations.

Using HM drastically matters since it guarantees the best harmonies are purified for components of the selected solution. To meritoriously apply the memory, HS process implements a factor HMCR ∈ [0,1], that is known as rate of harmony memory accepting. In case the best and few harmonies are selected, this rate is too low and the convergence is very slow. On the other hand, when the rate is tremendously high, it means the pitches in harmony memory are almost fully working, the other ones are not working well and hence they're not led to proper solution. As a result, the investigators suggest 0.7 and 0.95 as the range of HMCR. Aslo since the relation of it is linear, at the first iteration, it has higher value and it incrementally decrease till the end of iterations that has low value.

Memory consideration: The usage of harmony memory (HM) is important because it ensures that good harmonies are considered as elements of new solution vectors. In order to use this memory effectively, the HS algorithm adopts a parameter HMCR ∈ [0,1], called harmony memory considering (or accepting) rate.

If this rate is too low, only few elite harmonies are selected and it may converge too slowly. If this rate is extremely high (near 1), the pitches in the harmony memory are mostly used, and other ones are not explored well, leading not into good solutions. Therefore, based on researchers suggestion HMCR is considered between 0.7 and 0.95. We

consider a linear relation for HMCR so that it has bigger value at first iteration and lower value at the end of iterations.

Using HM drastically matters since it guarantees the best harmonies are purified for components of the selected solution. To meritoriously apply the memory, HS process implements a factor HMCR ∈ [0,1], that is known as rate of harmony memory accepting. In case the best and few harmonies are selected, this rate is too low and the convergence is very slow. On the other hand, when the rate is tremendously high, it means the pitches in harmony memory are almost fully working, the other ones are not working well and hence they're not led to proper solution. As a result, the investigators suggest 0.7 and 0.95 as the range of HMCR. Aslo since the relation of it is linear, at the first iteration, it has higher value and it incrementally decrease till the end of iterations that has low value.

Memory consideration: The usage of harmony memory (HM) is important because it ensures that good harmonies are considered as elements of new solution vectors. In order to use this memory effectively, the HS algorithm adopts a parameter HMCR ∈ [0,1], called harmony memory considering (or accepting) rate.

If this rate is too low, only few elite harmonies are selected and it may converge too slowly. If this rate is extremely high (near 1), the pitches in the harmony memory are mostly used, and other ones are not explored well, leading not into good solutions. Therefore, based on researchers suggestion HMCR is considered between 0.7 and 0.95. We consider a linear relation for HMCR so that it has bigger value at first iteration and lower value at the end of iterations.

Using HM drastically matters since it guarantees the best harmonies are purified for components of the selected solution. To meritoriously apply the memory, HS process implements a factor HMCR ∈ [0,1], that is known as rate of harmony memory accepting. In case the best and few harmonies are selected, this rate is too low and the convergence is very slow. On the other hand, when the rate is tremendously high, it means the pitches in harmony memory are almost fully working, the other ones are not working well and hence they're not led to proper solution. As a result, the investigators suggest 0.7 and 0.95 as the range of HMCR. Aslo since the relation of it is linear, at the first iteration, it has higher value and it incrementally decrease till the end of iterations that has low value.

$$HMCR(t) = HMCR_{\min} + \frac{HMCR_{\max} - HMCR_{\min}}{NI} \times t \quad (10)$$

Where HMCR(t) is harmony memory consideration rate for iteration t, $HMCR_{min}$ and $HMCR_{max}$ are the minimum and maximum value for *HMCR*, respectively and finally *NI* is number of iterations.

Pitch adjustment: The second component is the pitch adjustment which has parameters such as pitch bandwidth (bw) and pitch adjusting rate (PAR). As the pitch adjustment in music means changing the frequency, it means generating a slightly different value in the HS algorithm. In theory, the pitch can be adjusted linearly or nonlinearly, but in practice, linear adjustment is used. This operation uses the PAR parameter, which is the rate of pitch adjustment and r which is a random number between 0 and 1; and bw is an arbitrary distance bandwidth. Pitch adjustment is similar to the mutation operator in genetic algorithms. We can assign a pitch-adjusting rate (PAR) to control the degree of the adjustment (Figure 5). A low pitch adjusting rate with a narrow bandwidth can slow down the convergence of HS because of the limitation in the exploration of only a small subspace of the whole search space. On the other hand, a very high pitch-adjusting rate with a wide bandwidth may cause the solution to scatter around some potential optima as in a random search. It has examined that PAR=0.1~0.5 is the best value to obtain better results. We also used another modification for pitch adjustment rate and consider a linear relation so that it has bigger value at first iteration and lower value at the end of iterations (See eqn. (11)).

$$\begin{aligned} &If \quad rand \prec PAR \\ &\quad x'_{:,j} = x'_{:,j} \pm r \times bw \\ &else \\ &x'_{:,j} = x'_{:,j} \\ &endif \end{aligned}$$

Figure 5: Pitch adjustment.

Pitch adjustment is the second part. Its factors are Pitch bandwidth (bw) and pitch adjusting rate (PAR). In music, this means frequency change. Here, it means adjusting the value in HS algorithm. Both adjustment, linear and non-linear are possible in theory but in fact only linear adjustment is applied that uses the PAR factor, which is the rate of pitch adjustment and r which is a random number between 0 and 1; and bw is an arbitrary distance bandwidth. Pitch adjustment performs like mutation operator in genetic algorithms it means it's possible to allocate a PAR to regulate the degree of adjustment (Figure 5). Due to restriction in searching subspace, a low PAR and narrow bw decelerate the convergence of HS. Vice versa, in random exploration, a high PAR and wide bw leads to the key to scatter around some potential optima.. It has examined that PAR=0.1~0.5 is the best value to reach the best results. We also used another modification for PAR. The relation is linear, i.e. at the first iteration, the value is higher and at the ending iterations, the value becomes lower and lower (See eqn. (11)).

$$PAR(t) = PAR_{\min} + \frac{PAR_{\max} - PAR_{\min}}{NI} \times t \quad (11)$$

Random selection: The third component is the randomization, which is to increase the diversity of the solutions. Although the pitch adjustment has a similar role, it is limited to certain area and thus corresponds to a local search. The use of randomization can drive the system further to explore various diverse solutions so as to attain the global optimality.

Harmony memory update

If the newly generated harmony vector gives a better function value than the worst one, the new harmony vector is included in the HM and the worst harmony is excluded.

Affinity function

Affinity function was used for avoiding from premature convergence and increasing the diversification. Affinity function allows us the generated solutions with high diversity. We consider a parameter called percentage of affinity, which is denoted PAF for defining the percentage of good sorted solutions, which remained at each iteration, and then the remained capacity of the population is selected from unique solutions existing among the present solutions. If unique solutions were not enough for filling the remained capacity of population, we have to use the repetitive solutions.

In order to prevent from untimely convergence and also to increase the diversification, affinity function was applied. This function provides

us the capability to create diverse solutions. Affinity percentage is a parameter that is meant PAF to show the percentage of good sorted solutions. The solutions that we have at each iteration and hence, the capacity of the quantity of infrequent solutions that are among the percentage. If infrequent solutions were not sufficient to fill the free capacity of total quantity, we must apply repetitive solutions.

Termination criterion

The process of harmony search is stopped if the number of repetition ends. The highest number of repetition is shown by MaxIt. Our proposed algorithm in pseudo code is shown in Figure 6.

Computational Experiments

Problem design

In this study we examined the effectiveness of the proposed approaches for a 15 test problems. The problem data can be characterized by three factors in terms of the number of jobs, number of machines, processing time, sequences dependent setup times, Ready time and machine availability time. Table 1 shows the random generated problems in detail.

1) Calculate mean processing time of each job on all s stages.

$$p_j = \text{round}\left(\sum_{i=1}^{s} \frac{\sum_{u=1}^{no.eligiblemachine} p_{iu}^{j}}{no.eligiblemachine_i}\right), \forall i \in N \quad (12)$$

2) Calculate average setup times for all possible subsequent jobs and sum it for all s stages.

$$s_j = \text{round}\left(\sum_{i=1}^{s} \frac{\sum_{u=1}^{no.eligiblemachine} s_{k,j,u}^{i}}{(n-1)\times no.eligiblemachine_i}\right), \forall i \in N \quad (13)$$

Determine a due date for each job with following formula.

$$d_j = p_j + s_j + \text{round}\left(\alpha \times \text{U}[0, \frac{\sum_{j=1}^{n}(p_j + s_j)}{\sum_{i=1}^{s} mi}]\right) \quad (14)$$

```
begin
  Define harmony memory considering rate (HMCR)
  Define pitch adjustment rate (PAR)
  Define Maximum Iteration (MaxIt)
  Define a percent for affinity function (P_AF)
 Generate Harmony Memory with random harmonies
while (t < MaxIt)
  while (i ≤ number of devision variables)
    if (rand < HMCR(t)), Choose a value from HM for the variable i
      if (rand < PAR(t)), Adjust the value by adding certain amount
      endif
    else Choose a random value
     end if
   end while
   Accept the new harmony if better
   Apply affinity function
  end while
  Find the currrent best solution
 end
```

Figure 6: Pseudo code for harmony search.

Factors	Levels
Number of job	8,16,20,24,30
Number of stages	2,3,4
Number of Machines at each stage	U(1,4)
Processing times	U(1,100)
Sequence dependent setup times	U(5,20)
Ready time	U(1,100)
Machine availability time	U(500,1000)

Table 1: Problem parameters and their levels.

Parameter tuning

It is known that the great choice of parameters has striking impact on performance of algorithms. Furthermore, the suitable design parameter values highly depend on the type of problems. Most of researches where were conducted by using evolutionary algorithms generally have been fixed parameter values after some preliminary experiment or have been fixed with reference to values of the previous similar literature. Main motive of this behaviour related to large number of parameters and their levels; because a comprehensive calibration requires to time and resource-consuming. Calibration plays a prominent role in improvement of performance of algorithm and in some cases it is a compulsory step in the developing the algorithms. For the purpose of calibration of algorithms some methods were used in literature. However, the most frequently used and exhaustive approach is a full factorial experiment [64,65]. This methodology usually is utilized when number of factor and their levels also CPU time of algorithm is small or moderate. Using of this approach gets very difficult for algorithms with numerous factors and levels and high CPU time. To diminish the number of required tests, fractional factorial experiment (FFE) was developed [66]. FFEs permit only a portion of the total possible combinations to estimate the main effect of factors and some of their interactions [67].

We already knew the algorithms performance is highly impacted by choice of parameters. Besides, the problem type is a key factor to the suitable design parameter values. Usually, the researches that are run using evolutionary algorithm are fixed the values after sequences of experiment or by going back to values of previous identical experiences. The key reason to conduct as such, is depended to the quantity of parameters as well as their level. Because abroad correction needs time as well as consumption of many resources. Correction or calibration is a main factor to increase the performance level of algorithm and sometimes is mandatory phase. In order to do the calibration, some methodologies were discussed in papers. However, the most frequently used and exhaustive approach is a full factorial experiment. This approach is generally applied if quantity of factors, their levels, CPU time of algorithm are low or near average. To diminish the number of required tests, fractional factorial experiment (FFE) was developed. FFEs permit only a portion of the total possible combinations to estimate the main effect of factors and some of their interactions.

A family of matrices decreasing the number of experiments is established by Ross [68]. Taguchi developed a family of FFE matrices which eventually reduce the number of experiments, but still provide sufficient information. In Taguchi method, the orthogonal arrays are used to study a large number of decision variables with a small number of experiments.

The experimental design proposed by Taguchi involves using orthogonal arrays to organize the parameters affecting the process and the levels at which they should be varying. Instead of having to test all possible combinations like the factorial design, the Taguchi method tests pairs of combinations. This makes collection of the necessary data to determine which factors have most significant effects on product quality with a minimum amount of experiment, thus saving time and resources. An advantage of the Taguchi method is that it emphasizes a mean performance characteristic value close to the target value rather than a value within certain specification limits, thus improving the final quality. Additionally, Taguchi method for experimental design is straightforward and easy to apply for many engineering situations. This makes it a powerful simple tool yet. It can be used to various research projects or to identify problems in a manufacturing process. Also, it has wide range of application in analysis of many different parameters without a prohibitively high amount of experimentation.

In Taguchi approach, variables are classified in two groups: controllable and noise factors (uncontrollable). Noise factors are those over which we have no direct control. Since elimination of the noise factors is impractical and often impossible, the Taguchi method seeks to minimize the effect of noises and determine optimal levels of important controllable factors based on the concept of robustness [69].

Besides determining the optimal levels, Taguchi identifies the relative significance of individual factors in terms of their main effects on the objective function. Taguchi has created a transformation of the repetition data to another value which is the measure of variation. The transformation is the signal-to-noise (S/N) ratio which explains why this type of parameter design is called robust design. Here, the term "signal" denotes the desirable value (mean response variable) and "noise" denotes the undesirable value (standard deviation); so the S/N ratio indicates the amount of variation presents in the response variable. The aim is to maximize the signal-to-noise ratio [70-83].

Taguchi method classifies the variable into two sets. Controllable and uncontrollable (or sometimes known as noise factors). Noise factors are those over which we have no direct control. Since elimination of the noise factors is impractical and often impossible, the Taguchi method seeks to minimize the effect of noises and determine optimal levels of important controllable factors based on the concept of robustness. Not only Taguchi does determine the importance ratio of every single factor in terms of their impact on the aiming function, but also it identifies its ideal level too. What Taguchi introduced, is a sort of modification in the iterative data to an additional value that is called measure of variation. The transformation is the signal-to-noise (S/N) ratio which explains why this type of parameter design is called robust design. By "signal" it infers, those values that are critical or anticipated. Dislike the "signal", by "noise" it infers those values that are not wanted or anticipated (Standard Deviation). Hence, this is a percentage of variation in response variable. This is what we need to try to increase [84-97].

Taguchi categorizes objective functions into three groups: the smaller-the-better type, the larger-the-better type, and nominal-is-best type. Since almost all objective functions in scheduling are categorized in the smaller-the-better type, its corresponding S/N ratio is:

$$S/N \text{ ratio} = -\log_{10}\left(-\sum(objective\ function)\right) \quad (15)$$

Before calibration of HS, the algorithm is subjected to some preliminary tests to obtain the proper parameter levels to be tested in the fine-tuning process. Some quick experiments showed that In order to achieve to more accurate and stable results for our proposed algorithm, we considered eleven parameters for tuning. These parameters are HMCR, PAR, MaxIt, nPop, P_{AF} which they are shown with their level in Table 2. The considered orthogonal array with eleven factors and three levels in Taguchi method is L27. The orthogonal array L27 is presented in Table 3.

By calculating all of experimental results in Taguchi method, the average S/N ratio and average of maximum completion time were obtained for both considered scales. Figures 6 displays the average S/N ratio obtained at each level. As illustrated in Figure 7, optimal levels are A(3), B(1), C(3), D(3), E(1), F(1). Furthermore, computed results in terms of Objective Function in Taguchi experimental analysis confirmed the achieved optimal levels using S/N ratio (Figure 8). The ranking for importance of harmony search's parameters showed in Figure 9.

Results

In this section the results of tested experiments for all algorithms are presented and the performance of the proposed algorithms is compared to each other in terms of the performance metrics. All algorithms were coded using MATLAB 2013a and run on personal computer with a 2.66 GHz CPU and 4 GB main memory.

	A	B	C	D	E	F
Level		HMS	nPop	HMCR	PAR	PAF
1	100	5	25	0.75	0.1	0.40
2	150	10	40	0.85	0.3	0.50
3	200	15	80	0.95	0.5	0.60

Table 2: Parameters and their levels.

	A	B	C	D	E	F
1	1	1	1	1	1	1
2	1	1	1	1	2	2
3	1	1	1	1	3	3
4	1	2	2	2	1	1
5	1	2	2	2	2	2
6	1	2	2	2	3	3
7	1	3	3	3	1	1
8	1	3	3	3	2	2
9	1	3	3	3	3	3
10	2	1	2	3	1	2
11	2	1	2	3	2	3
12	2	1	2	3	3	1
13	2	2	3	1	1	2
14	2	2	3	1	2	3
15	2	2	3	1	3	1
16	2	3	1	2	1	2
17	2	3	1	2	2	3
18	2	3	1	2	3	1
19	3	1	3	2	1	3
20	3	1	3	2	2	1
21	3	1	3	2	3	2
22	3	2	1	3	1	3
23	3	2	1	3	2	1
24	3	2	1	3	3	2
25	3	3	2	1	1	3
26	3	3	2	1	2	1
27	3	3	2	1	3	2

Table 3: The orthogonal array L_{27}.

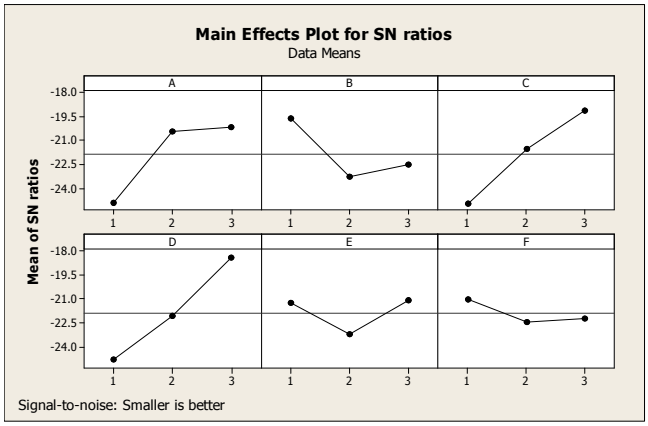

Figure 7: The S/N ratio plot in Taguchi methodology.

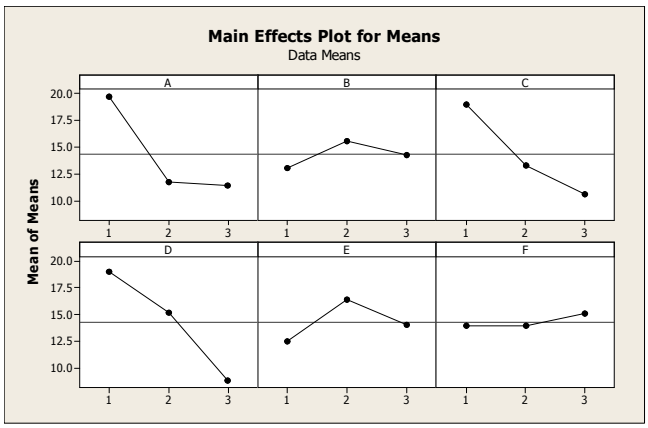

Figure 8: The Mean of objective function plot in Taguchi methodology.

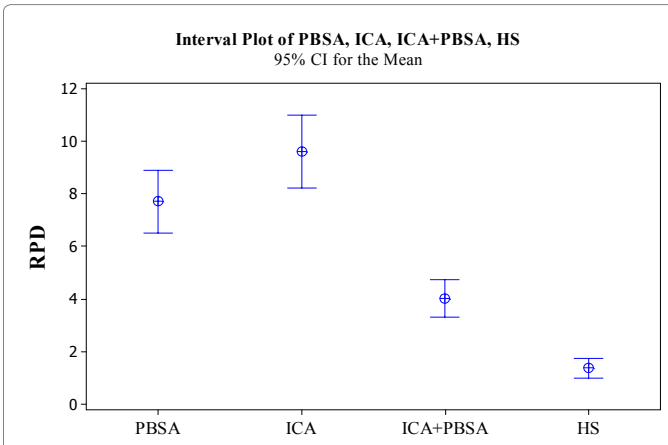

Figure 9: Means and interval plot for random problems in terms of RPD.

The effectiveness of the algorithms was testified by solving 15 different problems. Tables 3-5 show the comparative results of the three algorithms with respect to five performance measures for small and large scale problem respectively [98-106].

Regarding the performance measures, Relative Percentage Deviation (RPD) over the best solutions is used. It is calculated as follows:

Level	A	B	C	D	E	F
1	-24.88	-19.64	-24.94	-24.78	-21.27	-21.03
2	-20.43	-23.25	-21.51	-22.07	-23.21	-22.43
3	-20.16	-22.50	-19.14	-18.40	-21.09	-22.24
Delta	4.72	3.61	5.80	6.38	2.12	1.40
Rank	3	4	2	1	5	6

Table 4: SN Ratio table for parameters in Taguchi methodology.

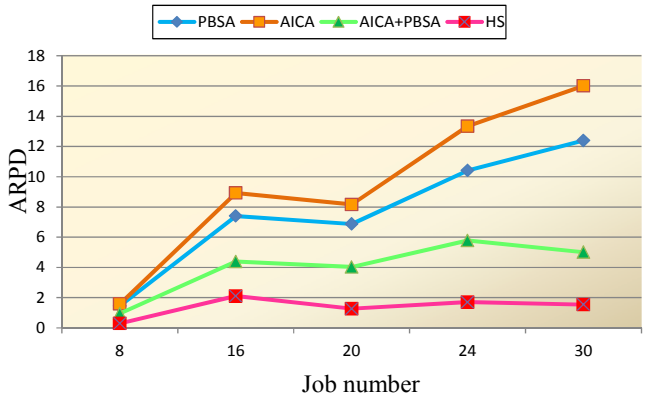

Figure 10: Means plot between the type of algorithm and number of jobs in terms of ARPD.

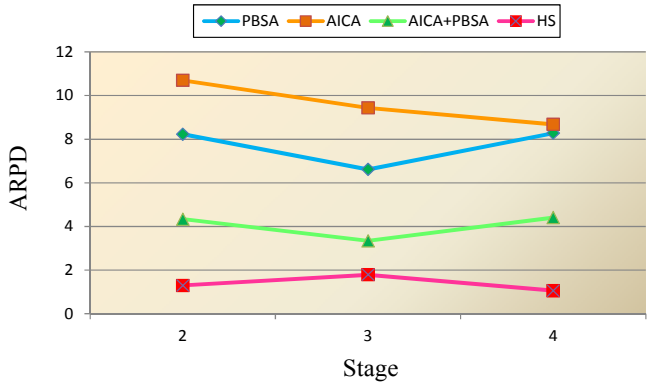

Figure 11: Means plot between the type of algorithm and number of stages in terms ARPD.

$$RPD = \frac{|Method_{sol} - Best_{sol}|}{Best_{sol}} \times 100 \quad (16)$$

Where $Method_{sol}$ is value of method and $Best_{sol}$ is the best value between the algorithms.

Summary results for test problems in terms of $\overline{RPD}$, standard deviation of RPD, best RPD and worst RPD are shown in Table 4. The results indicated that, HS strongly outperforms the three other algorithms. As can be seen in Table 4, in small size problems (with 8 jobs) two of algorithms (HS and AICA+PBSA) have reached to best solution and there is no difference among algorithms. In medium size and large size problems, HS outperformed the other algorithms. We conducted means plot and Tukey intervals for the algorithms at 95% confidence interval. Figure 9 demonstrated that the HS algorithm statistically outperformed the other algorithms. Furthermore, we analysed the behaviour of algorithms in different scenarios. Figures 10 and 11 indicated that HS have better results in terms of ARPD

No. jobs	No. Stages	ARPD				Best RPD				Worst RPD				Standard deviation of RPDs			
		PBSA	AICA	AICA+PBSA	HS	PBSA	AICA	AICA+PBSA	HS	PBSA	AICA	AICA+PBSA	HS	PBSA	AICA	AICA+PBSA	HS
	2	2/32	2/26	1/00	0/49	0/04	0/99	0/00	0/00	4/55	4/94	2/92	1/89	1/65	1/18	1/08	0/74
8	3	1/06	1/12	0/91	0/16	0/00	0/00	0/00	0/00	2/42	3/38	2/20	0/74	0/96	1/10	0/88	0/28
	4	0/98	1/36	0/93	0/22	0/00	0/00	0/00	0/00	2/74	3/58	2/90	1/32	1/02	1/33	1/19	0/42
	2	8/43	8/98	3/04	1/37	0/00	0/00	0/00	0/00	13/66	18/82	11/32	7/23	5/96	8/44	4/54	2/28
16	3	6/15	8/93	5/88	3/13	0/00	0/00	0/00	0/00	19/38	25/00	15/58	11/05	6/38	7/26	6/07	3/99
	4	7/63	8/86	4/24	1/81	0/00	0/00	0/00	0/00	12/99	16/21	13/69	4/44	5/55	6/16	5/50	1/49
	2	6/82	11/58	5/29	1/10	0/00	3/40	0/00	0/00	18/30	21/01	7/41	5/81	8/12	6/82	2/86	2/18
20	3	4/45	5/42	1/36	1/98	0/00	0/00	0/00	0/00	15/67	10/26	2/82	7/57	4/85	3/79	1/42	3/09
	4	9/37	7/48	5/43	0/72	5/02	3/71	0/00	0/00	15/70	12/94	9/60	3/48	2/74	3/12	3/75	1/46
	2	12/60	11/80	6/76	1/60	0/00	0/00	0/00	0/00	35/15	24/60	16/21	8/36	9/48	6/93	5/57	2/86
24	3	6/41	13/62	3/25	1/89	0/00	3/61	0/00	0/00	16/22	31/00	6/79	6/62	6/48	7/88	2/48	2/34
	4	12/23	14/62	7/33	1/61	0/00	0/00	0/00	0/00	17/35	21/64	17/48	8/15	5/23	8/64	6/24	2/84
	2	10/98	18/86	5/60	1/91	0/00	0/00	0/00	0/00	14/11	30/44	12/84	8/45	4/11	13/29	4/87	3/50
30	3	15/02	18/09	5/32	1/79	0/00	0/00	0/00	0/00	23/81	33/83	9/84	5/98	10/30	10/02	3/97	2/89
	4	11/20	11/08	4/11	0/93	0/00	0/00	0/00	0/00	23/26	22/72	12/03	3/14	10/36	8/61	4/91	1/28
	Average	7/71	9/61	4/03	1/38	0/34	0/78	0/00	0/00	15/69	18/69	9/57	5/61	5/55	6/30	3/69	2/11

Table 5: Summary results for small scale problems in terms of ARPD, Best RPD, Worst RPD and Standard deviation of RPDs.

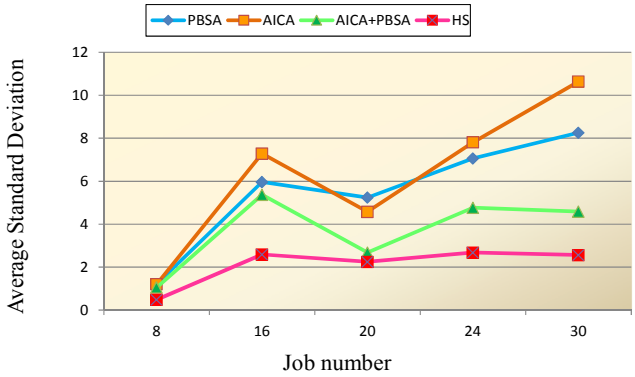

Figure 12: Means plot between the type of algorithm and number of stages in terms of average standard deviation.

for all of examples with different job numbers and different stages, respectively. We also examined our proposed algorithms in term of standard deviation. As can be seen in Figure 12, there is a same manner for algorithms except for job number 20 (Table 5).

Conclusion and Further Researches

This paper presents an industrial scheduling problem as a no-wait hybrid flow shop problem with sequence dependent setup times, different ready times and machine availability time. This problem has many applications in wide ranges of modern manufacturing and service industries. To our knowledge, there is no other published work that considers finding an optimal schedule for this problem. We propose an effective harmony search algorithm to tackle the considered problem. To validate the proposed algorithm, we used fifteen test problems and evaluated the performance and the reliability of the proposed algorithm. Computational simulations and comparisons demonstrated the effectiveness and efficiency of the proposed HS algorithm. There are a number of research directions that can be considered as useful extensions of this approach. As a direction for further researches in this area, the influence of the starting solution should be investigated. Moreover, hybrid algorithms should be developed by using a local search like simulated annealing or variable neighbourhood search within a HS. Other issues that are worthy of future research includes developing and testing of novel meta-heuristics like firefly algorithm, graph colouring-based algorithm. Developing models with some practical assumptions like emergency maintenance, learning effect and deterioration may be other fruitful topics for future investigations.

References

1. Marichelvam MK, Tosun Ö, Geetha M (2017) Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time. Applied Soft Computing 55: 82-92.
2. Fanjul-Peyro L, Perea F, Ruiz R (2017) MIP models and mataheuristics for the unrelated parallel machine scheduling problem with additional resources. European Journal of Operational Research.
3. Low C, Wu GH (2016) Unrelated parallel-machine scheduling with controllable processing times and eligibility constraints to minimize the makespan. Journal of Industrial and Production Engineering 33: 286-293.
4. Yin Y, Wu WH, Cheng TCE, Wu CC (2015) Single-machine scheduling with time-dependent and position-dependent deteriorating jobs. International Journal of Computer Integrated Manufacturing 28: 781-790.
5. Joo CM, Kim BS (2015) Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability. Computers & Industrial Engineering 85: 102-109.
6. Liaw CF (2016) A branch-and-bound algorithm for identical parallel machine total tardiness scheduling problem with preemption. Journal of Industrial and Production Engineering 33: 426-434.
7. Pei J, Liu X, Fan W, Pardalos PM, Migdalas A, et al. (2016) Scheduling jobs on a single serial-batching machine with dynamic job arrivals and multiple job types. Annals of Mathematics and Artificial Intelligence 76: 215-228.
8. Jungwattanakit J, Reodecha M, Chaovalitwongse P, Werner F (2009) A comparison of scheduling algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. Computers & Operations Research 36: 358-378.
9. Ruiz R, Şerifoğlu FS, Urlings T (2008) Modeling realistic hybrid flexible flowshop scheduling problems. Computers & Operations Research 35: 1151-1175.
10. Haouari M, Hidri L (2008) On the hybrid flowshop scheduling problem. International Journal of Production Economics 113: 495-497.
11. Behnamian J, Ghomi SF, Zandieh M (2009) A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. Expert Systems with Applications 36: 11057-11069.
12. Yaurima V, Burtseva L, Tchernykh A (2009) Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. Computers & Industrial Engineering 56: 1452-1463.
13. Naderi B, Gohari S, Yazdani M (2014) Hybrid flexible flowshop problems:

Models and solution methods. Applied Mathematical Modelling 38: 5767-5780.

14. Bozorgirad MA, Logendran R (2013) Bi-criteria group scheduling in hybrid flowshops. International Journal of Production Economics 145: 599-612.
15. Pan QK, Wang L, Li JQ, Duan JH (2014) A novel discrete artificial bee colony algorithm for the hybrid flowshop scheduling problem with makespan minimisation. Omega 45: 42-56.
16. Behnamian J, Zandieh M (2011) A discrete colonial competitive algorithm for hybrid flowshop scheduling to minimize earliness and quadratic tardiness penalties. Expert Systems with Applications 38: 14490-14498.
17. Zandieh M, Hashemi AR (2015) Group scheduling in hybrid flexible flowshop with sequence-dependent setup times and random breakdowns via integrating genetic algorithm and simulation. International Journal of Industrial and Systems Engineering 21: 377-394.
18. Shao W, Pi D, Shao Z (2016) A hybrid discrete optimization algorithm based on teaching-probabilistic learning mechanism for no-wait flow shop scheduling. Knowledge-Based Systems 107: 219-234.
19. Riahi V, Kazemi M (2016) A new hybrid ant colony algorithm for scheduling of no-wait flowshop. Operational Research 1-20.
20. Linn R, Zhang W (1999) Hybrid flow shop scheduling: a survey. Computers & industrial engineering 37: 57-61.
21. Ruiz R, Vázquez-Rodríguez JA (2010) The hybrid flow shop scheduling problem. European Journal of Operational Research 205: 1-18.
22. Ribas I, Leisten R, Framiñan JM (2010) Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. Computers & Operations Research 37: 1439-1454.
23. Rajendran C (1994) A no-wait flowshop scheduling heuristic to minimize makespan. Journal of the Operational Research Society 45: 472-478.
24. Hall NG, Sriskandarajah C (1996) A survey of machine scheduling problems with blocking and no-wait in process. Operations research 44: 510-525.
25. Grabowski J, Pempera J (2000) Sequencing of jobs in some production system. European Journal of Operational Research 125: 535-50.
26. Raaymakers WH, Hoogeveen JA (2000) Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. European Journal of Operational Research 126: 131-151.
27. Aldowaisan T, Allahverdi A (2004) A New heuristics for m-machine no-wait flow shop to minimize total completion time. Omega 32: 345-352.
28. Chang JL, Gong DW (2007) A heuristic genetic algorithm for no-wait flowshop scheduling problem. Journal of China University of Mining and Technology 17: 582-586.
29. Gilmore PC, Gomory RE (1964) Sequencing a one-state variable machine: a solvable case of the travelling salesman problem. Operations Research 12: 655-679.
30. Cheng TCE, Wang G, Sriskandarajah C (1999) One-operator-two-machine flowshop scheduling with setup and dismounting times. Computers and Operations Research 26: 715-730.
31. Allahverdi A, Aldowaisan T (2004) No-wait flowshops with bicriteria of makespan and maximum lateness. European Journal of Operational Research 152: 132-147.
32. Li X, Wang Q, Wu C (2008) Heuristic for no-wait flow shops with makespan minimization. International Journal of Production Research 46: 2519-2530.
33. Pan QK, Tasgetiren MF, Liang YC (2008) A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. Computers & Operations Research 35(9): 2807-2839.
34. Framinan JM, Nagano MS (2008) Evaluating the performance for makespan minimisation in no-wait flowshop sequencing. Journal of materials processing technology 197: 1-9.
35. Ruiz R, Allahverdi A (2009) New heuristics for no-wait flow shops with a linear combination of makespan and maximum lateness. International Journal of Production Research 47: 5717-5738.
36. Samarghandi H, ElMekkawy TY (2011). An efficient hybrid algorithm for the two-machine no-wait flow shop problem with separable setup times and single server. European Journal of Industrial Engineering 5: 111-131.
37. Ben Chihaoui F, Kacem I, Hadj-Alouane AB, Dridi N, Rezg N (2011) No-wait scheduling of a two-machine flow-shop to minimise the makespan under non-availability constraints and different release dates. International Journal of Production Research 49: 6273-6286.
38. Shafaei R, Rabiee M, Mirzaeyan M (2011) An adaptive neuro fuzzy inference system for makespan estimation in multiprocessor no-wait two stage flow shop. International Journal of Computer Integrated Manufacturing 24: 888-899.
39. Moradinasab N, Shafaei R, Rabiee M, Ramezani P (2013) No-wait two stage hybrid flow shop scheduling with genetic and adaptive imperialist competitive algorithms. Journal of Experimental & Theoretical Artificial Intelligence 25: 207-225.
40. Rabiee M, Zandieh M, Jafarian A (2012) Scheduling of a no-wait two-machine flow shop with sequence-dependent setup times and probable rework using robust meta-heuristics. International Journal of Production Research 50: 7428-7446.
41. Arabameri S, Salmasi N (2013) Minimization of weighted earliness and tardiness for no-wait sequence-dependent setup times flowshop scheduling problem. Computers & Industrial Engineering 64: 902-916.
42. Nagano MS, Araújo DC (2014) New heuristics for the no-wait flowshop with sequence-dependent setup times problem. Journal of the Brazilian Society of Mechanical Sciences and Engineering 36: 139-151.
43. Allahverdi A, Aydilek H (2014) Total completion time with makespan constraint in no-wait flowshops with setup times. European Journal of Operational Research 238: 724-734.
44. Samarghandi H, ElMekkawy TY (2014) Solving the no-wait flow-shop problem with sequence-dependent set-up times. International Journal of Computer Integrated Manufacturing 27: 213-228.
45. Gupta JND, Strusevich VA, Zwaneveld CM (1997) Two-stage no-wait scheduling models with setup and removal times separated. Computer and Operation Research 24: 1025-1031.
46. Aldowaisan T, Allahverdi A (1998) Total flowtime in no-wait flow shops with separated setup times. Computer and Operation Research 25: 757-765.
47. Aldowaisan T (2001) A new heuristic and dominance relations for no-wait flowshops with setups. Computer and Operation Research 28: 563-584.
48. Sidney JB, Potts CN, Sriskandarajah C (2000) A heuristic for scheduling two-machine no-wait flow shops with anticipatory setups. Operations Research Letters 26: 165-173.
49. Liu Z, Xie J, Li J, Dong J (2003) A heuristic for two-stage no-wait hybrid flowshop scheduling with a single machine in either stage. Tsinghua Science and Technology 8: 43-48.
50. Xie J, Xing W, Liu Z, Dong J (2004) Minimum deviation algorithm for two-stageno-wait flowshops with parallel machines. Computers & Mathematics with Applications 47: 1857-1863.
51. Huang RH, Yang CL, Huang YC (2009) No-wait two-stage multiprocessor flow shop scheduling with unit setup. International Journal of Advanced Manufacturing Technology 44: 921-927.
52. Jolai F, Sheikh S, Rabbani M, Karimi B (2009) A genetic algorithm for solving no-wait flexible flow lines with due window and job rejection. International Journal of Advanced Manufacturing Technology 42: 523-532.
53. Jolai F, Rabiee M, Asefi H (2012) A novel hybrid meta-heuristic algorithm for a no-wait flexible flow shop scheduling problem with sequence dependent setup times. Int J Prod Res 50: 7447-7466.
54. Rabiee M, Rad RS, Mazinani M, Shafaei R (2014) An intelligent hybrid meta-heuristic for solving a case of no-wait two-stage flexible flow shop scheduling problem with unrelated parallel machines. The International Journal of Advanced Manufacturing Technology 71: 1229-1245.
55. Asefi H, Jolai F, Rabiee M, Araghi MT (2014) A hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flowshop scheduling problem. The International Journal of Advanced Manufacturing Technology 75: 1017-1033.
56. Khalili M, Naderi B (2014) A bi-objective imperialist competitive algorithm for no-wait flexible flow lines with sequence dependent setup times. The International Journal of Advanced Manufacturing Technology 76: 461-469.
57. Marichelvam MK, Geetha M (2016) Application of novel harmony search algorithm for solving hybrid flow shop scheduling problems to minimise makespan. International Journal of Industrial and Systems Engineering 23: 467-481.

58. Gao KZ, Suganthan PN, Pan QK, Chua TJ, Cai TX, et al. (2016) Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives. Journal of Intelligent Manufacturing, 27: 363-374.

59. Guo Z, Shi L, Chen L, Liang Y (2017) A harmony search-based memetic optimization model for integrated production and transportation scheduling in MTO manufacturing. Omega 66: 327-343.

60. Wang GG, Gandomi AH, Zhao X, Chu HCE (2016) Hybridizing harmony search algorithm with cuckoo search for global numerical optimization. Soft Computing 20: 273-285.

61. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: Harmony search. Simulation 76: 60-68.

62. Lee KS, Geem ZW (2004) A new meta-heuristic algorithm for continues engineering optimization: harmony search theory and practice. Comput. Meth. Appl. Mech. Eng. 194: 3902-3933.

63. Geem ZW, Kim JH, Loganathan GV (2002) Harmony search optimization: application to pipe network design. International Journal of Modelling and Simulation 22: 125-133.

64. Montgomery DC (2000) Design and analysis of experiments, (5thedn) Wiley, New York, USA

65. Ruiz R, Maroto C, Alcaraz J (2006) Two new robust genetic algorithms for the flowshop scheduling problem. Omega 34: 461-476.

66. Cochran WG, Cox GM (1992) Experimental Designs, 2nd ed., Wiley, USA, p: 640.

67. Naderi B, Ghomi SF, Aminnayeri M (2010) A high performing metaheuristic for job shop scheduling with sequence-dependent setup times. Applied Soft Computing 10: 703-710.

68. Ross PJ (1988) Taguchi Techniques for Quality Engineering. McGraw-Hill, New York, p: 329.

69. Phadke MS (1989) Quality engineering using robust design: Robuste Prozesse durch Quality Engineering. Deutsche Übersetzung: G. LIESEGANG. GFMT-VERLAG, MÜNCHEN 33.

70. Coello CAC, Veldhuizen DAV, Lamont GB (2002) Evolutionary algorithms for solving multi-objective problems. Singapore: Kluwer Academic.

71. Deb K (2001) Multiobjective optimization using evolutionary algorithms. Chichester UK, JohnWiley and Sons Ltd.

72. Deb K, Pratap A, Agarwal S, Meyarivan T (2002a) Fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6: 182-197.

73. Atashpaz-Gargari E, Lucas C (2007) Imperialist competitive algorithm: An algorithm for optimization inspired by imperialist competitive. IEEE Congress on Evolutionary computation, Singapore.

74. Shokrollahpour E, Zandieh M, Dorri B (2011) A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. International Journal of Production Research 49: 3087-3103.

75. Eiben AE, Smith JE (2003) Introduction to evolutionary computing, Springer 1st edition.

76. Furtuna R, Curteanu S, Leon F (2011) An elitist non-dominated sorting genetic algorithm enhanced with a neural network applied to the multi-objective Optimization of a polysiloxane synthesis process. Engineering Applications of Artificial Intelligence 24: 772-785.

77. Ghiasi H, Pasini D, Lessard L (2011) A non-dominated sorting hybrid algorithm for multi-objective optimization of engineering problems. Engineering Optimization 43: 39-59.

78. Hsieh JC, Chang PC, Hsu LC (2003) Scheduling of drilling operations in printed circuit board factory. Computers and Industrial Engineering 44: 461-73.

79. Hwang C, Yoon K (1981) Multiple Attribute Decision Making: Methods and Applications. Springer, Berlin.

80. Johnson SM (1954) Optimal two and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1: 61-68.

81. Khalili M (2012) Multi-objective no-wait hybrid flowshop scheduling problem with transportation times. Int J Comput Sci Eng 7: 147-154.

82. Khalili M (2013) A multi-objective electromagnetism algorithm for a bi-objective hybrid no-wait flowshop scheduling problem. Int J Adv Manuf Technol, pp: 1-11.

83. Kim DW, Na DG, Chen FF (2003) Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. Robotics and Computer Integrated Manufacturing 19: 173-181.

84. Kishor A, Yadav SP, Kumar S (2009) Interactive fuzzy multiobjective reliability optimization using NSGA-II. Opsearch 46: 214-224.

85. Knowles JD, Corne DW (1999) Local Search, Multiobjective Optimization and the Pareto Archived Evolution Strategy. In: Mckay B, Sarker R, Yao X, Tsujimura Y, Namatame A, et al. (eds.) Proceedings of The Third Australia-Japan Joint Workshop on Intelligent and Evolutionary Systems, Ashikaga, Japan, Ashikaga Institute of Technology, pp: 209-216.

86. Knowles JD, Corne DW (1999) The Pareto Archived Evolution Strategy: A New Baseline Algorithm for Multiobjective Optimization. 1999 Congress on Evolutionary Computation, Piscataway, NJ, July 1999. IEEE Service Center. pp: 98-105,

87. Knowles JD, Corne DW (2000) Approximating the non-dominated front using the Pareto Archived Evolution Strategy. Evolutionary Computation 8: 149-172.

88. Lin BM, Lin FC, Lee RCT (2006) Two-machine flow-shop scheduling to minimize total late work. Engineering Optimization 38: 501-509.

89. Lin SW, Ying KC (2016) Minimizing makespan for solving the distributed no-wait flowshop scheduling problem. Computers & Industrial Engineering 99: 202-209.

90. Minella G, Ruiz R, Ciavotta M (2008) A review and evaluation of multiobjective algorithms for the flowshop scheduling problem. INFORMS Journal on Computing 20: 451-471.

91. Karimi N, Zandieh M, Najafi AA (2010) Group scheduling in flexible flow shops: a hybridised approach of imperialist competitive algorithm and electromagnetic-like mechanism. International Journal of Production Research 49: 4965-4977.

92. Nagano MS, Miyata HH, Araújo DC (2015) A constructive heuristic for total flowtime minimization in a no-wait flowshop with sequence-dependent setup times. Journal of Manufacturing Systems 36: 224-230.

93. Pan QK, Wang L, Qian B (2009) A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems. Computers & Operations Research 36: 2498-2511.

94. Qian B, Wang L, Huang DX, Wang WL, Wang X (2009) An effective hybrid DE-based algorithm for multi-objective flow shop scheduling with limited buffers. Computers & Operations Research 36: 209-233.

95. Rabiee M, Zandieh M, Ramezani P (2012) Bi-objective partial flexible job shop scheduling problem: NSGA-II, NRGA, MOGA and PAES approaches. International Journal of Production Research 50: 7327-7342.

96. Rahimi-Vahed AR, Javadi B, Rabbani M, Tavakkoli-Moghaddam R (2008) A multi-objective scatter search for a bi-criteria no-wait flow shop scheduling problem. Engineering Optimization 40: 331-346.

97. Ramezani P, Rabiee M, Jolai F (2015) No-wait flexible flowshop with uniform parallel machines and sequence-dependent setup time: a hybrid meta-heuristic approach. Journal of Intelligent Manufacturing 26: 731-744.

98. Sivakumar K, Balamurugan C, Ramabalan S (2011) Simultaneous optimal selection of design and manufacturing tolerances with alternative manufacturing process selection. Computer-Aided Design 43: 207-218.

99. Sriskandarajah C, Ladet P (1986) Some no-wait shops scheduling problems: complexity aspect. European Journal of Operational Research 24: 424-438.

100. Steuer RE (1986) Multiple criteria optimization: theory, computation, and applications. Wiley Series in Probability and Statistics, p: 546.

101. Syswerda G (1989) Uniform crossover in genetic algorithms. In Proceedings of the 3rd International Conference on Genetic Algorithms.

102. Tasgetiren MF, Pan QK, Suganthan PN, Liang YC (2007) A discrete differential evolution algorithm for the no-wait flowshop scheduling problem with total flowtime criterion. In Computational Intelligence in Scheduling, IEEE Symposium 251-258.

103. Tavakkoli-Moghaddam R, Rahimi-Vahed A, Mirzaei AH (2007) A hybrid multi-objective immune algorithm for a flow shop scheduling problem with bi-objectives: weighted mean completion time and weighted mean tardiness. Information Sciences 177: 5072-5090.

104. Yu L, Shih HM, Pfund M, Matthew CW, Fowler JW (2002) Scheduling of unrelated parallel machines: an application to PWB manufacturing. IIE transactions 34: 921-931.

105. Zandieh M, Karimi N (2011) An adaptive multi-population genetic algorithm to solve the multi-objective group scheduling problem in hybrid flexible flowshop with sequence-dependent setup times. Journal of Intelligent Manufacturing 22: 979-989.

106. Zhou G, Min H, Gen M (2003) A genetic algorithm approach to the bi-criteria allocation of customers to warehouses. International Journal of Production Economics 86: 35-45.