*Journal of Industrial and Systems Engineering*
Vol. 8, No. 2, pp 67-85
Spring 2015

# A comparison of algorithms for minimizing the sum of earliness and tardiness in hybrid flow-shop scheduling problem

## (Unrelated parallel machines and sequence-dependent setup times)

**F. Rajaee Abyaneh[1*], S. Gholami[2]**

[1]*Department of Industrial Engineering, K.N.ToosiUniversity of Technology, Tehran, Iran*

*far_raj@yahoo.com*

[2]*Faculty of Industrial Engineering, K.N.ToosiUniversity of Technology, Tehran, Iran*

**Abstract**

In this paper, we consider the flow-shop scheduling problem with unrelated parallel machines at each stage as well as sequence-dependent setup times under minimization of the sum of earliness and tardiness. Processing times, setup times and due-dates are known in advance. To solve the problem, we introduce a hybrid memetic algorithm and a particle swarm optimization combined with genetic operators. An application of simulated annealing is also presented for the evaluation of the proposed algorithm. A Taguchi design is also conducted to set the parameters. Finally, a comparison is made via 16 small size and 24 large size test problems for 10 times each. The results of one-way ANOVA applied in our test problems, demonstrate that the proposed algorithm performs as efficient as the HSA qualitatively and with 63.77% decline in elapsed time.

**Keywords:** scheduling; hybrid flow-shop; unrelated machines; sequence dependent setup time; earliness-tardiness.

## 1- Introduction

Hybrid flow-shops (HFS) also known as flexible flow-shops (FFSP) or multi-processor flow-shops are common manufacturing environments in which a number of jobs are to be processed in a series of production stages, each of which has multiple machines operating in parallel. Some stages may have only one facility, but at least one stage must have more than one facility parallel with others. The hybrid characteristic of a flow-shop can be seen in various industries. For instance, Yaurima et al. (2009) addressed HFS for television printed circuit-board (PCB) production. Most researches in the literature are dedicated to HFS with identical parallel machines at each stage. However, in real production environments, due to the high cost of replacement of the older facilities with the new ones, newer and more efficient facilities are

*Corresponding author.
ISSN: 1735-8272, 

usually in parallel with older and slower facilities,. Moreover, scheduling problem with sequence-dependent setup times (SDST) is found in many industries. We can refer to dying operation as an example. Therefore, the hybrid flow-shop scheduling problem with unrelated machines and sequence-dependent setup times is considered in this paper.

In this paper, we are interested to study the SDST hybrid flow-shop scheduling problem with unrelated machines and Earliness-Tardiness (ET) objective function. This function is used on one hand to minimize inventory or deterioration costs and on the other hand, to reduce customer dissatisfaction by meeting their jobs' due-dates. A few researches in the literature addressed ET objective function (almost 1% of the scheduling problems according to Ruiz and Vázquez-Rodríguez (2010)) and to the best of our knowledge, no research studied HFS with unrelated machines and ET penalties simultaneously. The schematic figure of the hybrid flow-shop problem is demonstrated in figure 1.

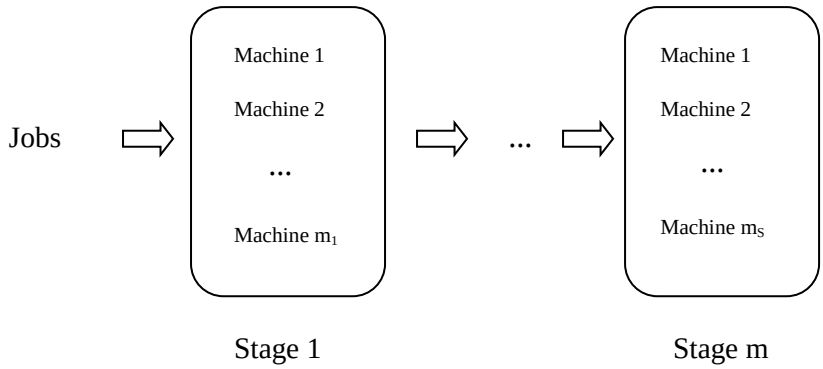



**Figure1.**Schematic figure of the hybrid flow-shop environment

As the single criterion HFS problem, made up of two stages with at least two machines available in one of the stages with make span criterion is an NP-hard problem, the bi-criteria scheduling problem can be assumed to be NP-hard also (Gupta, 1988). So developing heuristics and metaheuristics methods can provide acceptable results for this problem. As the simple genetic algorithm and the original PSO may not provide satisfactory results, they are hybridized with a local search algorithm. PSO is chosen due to its long computational time and for its rapid convergence. The PSO used in this article is a developed version initialized by Tasgetiren et al. (2004) which enables the use of original PSO in scheduling problems. It is also hybridised with genetic operators which are chosen by Taguchi method. An application of SA is developed in order to validate the obtained results. A new EDD based heuristic is also presented in order to generate the initial solution. Moreover, a heuristic algorithm is proposed and is hybridised with the metaheuristcs to find the schedule for the overall problem.

The remainder of this paper is organized as follows. In the next section, an overview regarding the problem is provided. Section 3 is dedicated to the description of the HFS problem and the MIP model. In section 4, 5 and 6, the hybrid memetic algorithm and the applications of SA and PSO are described. Parameter calibration and computational results are also provided in section 7. Finally, section 8 concludes the paper.

## 2- Literature review

Various forms of the HFS problem have been studied in the literature. The literature is reviewed according to the topic and the approaches used in this article. The literature review is summarized in table 1.

**Table1.**Litrature review

| Environment | Number of stages | Sequence-dependent | Performance measure | Solution approach | Author |
|---|---|---|---|---|---|
| HFS with Identical parallel machines | >=2 | Y | $C_{\max}$ | Several heuristics, a MILP model and RKGA | Kurz and Askin(2004) |
| HFS with identical parallel machines | >=2 | Y | $C_{\max}$ | Immune algorithm | Zandieh et al. (2006) |
| HFS with identical parallel machines, machine release dates and time lags | >=2 | Y | $C_{\max}$ | GA | Zandieh et al.(2010) |
| HFS with identical parallel machines | >=2 | Y | [1] $\overline{E}^h + \overline{T}^w + \overline{C}^v + \overline{d}^z$ | GRASP algorithm | Davoudpour and Ashrafi (2009) |
| HFS with identical parallel machines and machine release dates | >=2 | N | [2] $\overline{E}^h + \overline{T}^w + \overline{WT}^v$ | SA, TS, and hybrid SA/TS, heuristics | Janiak et al. (2007) |
| HFS with identical machines | >=2 | N | $ET$ | Heuristic | Farkhzand and Heydari (2008) |
| HFS with identical parallel machines | >=2 | Y | $\{C_{\max}, ET\}$ | Multi-phase method | Behnamian et al. (2009) |
| HFS with identical machines and limited waiting time | >=2 | Y | $\overline{E} + \overline{T^2}$ | A discrete version of colonial competitive algorithm (CCA) | Behnamian and Zandieh (2011) |
| HFS with unrelated machines and machine release dates | >=2 | Y | $\alpha C_{max}+(1-\alpha)\overline{U}$ | MILP, heuristics, GA, SA, TS | Jungwattanakit et al. (2005) |
| HFS with unrelated machines and machine release dates | >=2 | Y | $\alpha C_{max}+(1-\alpha)\overline{U}$ | A MILP model, dispatching rules, GA | Jungwattanakit et al. (2008) |
| HFS with unrelated machines and machine release dates | >=2 | Y | $\alpha C_{max}+(1-\alpha)\overline{U}$ | A MILP model, heuristics, dispatching rules, GA | Jungwattanakit et al. (2009) |
| HFS with unrelated machines and eligibility constraint | >=2 | Y | $C_{max}$ | Mathematical programming formulation, GA | Ruiz and Maroto (2006) |
| HFS with unrelated machines and transportation times | >=2 | Y | $\{\overline{F}, \overline{T}\}$ | SA | Naderi et al. (2009) |
| HFS with unrelated machines and processor blocking | >=2 | Y | $\{C_{max}, T_{max}\}$ | hybrid multi-objective parallel genetic algorithm | Rashidi et al.(2010) |

[1] $\overline{C}^v$ is the total weighted completion time

[2] $\overline{WT}^v$ is the total weighted waiting time

**Table1.**Litrature review continued

| Environment | Number of stages | Sequence-dependent | Performance measure | Solution approach | Author |
|---|---|---|---|---|---|
| HFS with unrelated machines and precedence constraints, time lags, machine eligibility and release times | >=2 | Y | $C_{max}$ | GA | Urlings and Ruiz (2010) |
| Single machine | =1 | N | $\overline{T}^{w}$ | PSO combined with local search algorithms | Tasgetiren et al. (2004) |
| Flow shop | >=2 | N | Single and multiple objectives | discrete version of a PSO with local search | Liao et al. (2007) |
| Job-shop with fuzzy processing time | – | N | $C_{max}$ | A combination of PSO with genetic operators | Niu et al. (2008) |
| Flow shop | >=2 | N | $C_{max}$ | A hybrid two phases PSO | Zhang et al. (2010) |

The research studies described above and the review paper presented by Ruiz and Vázquez-Rodríguez (2010) demonstrate that to the best of our knowledge, no studies considered unrelated machines and ET objective function in the HFS scheduling problem simultaneously. On the other hand, according to the literature review, SA and GA are the most popular metaheuristics used to solve the HFS problem, especially those problems with unrelated machines. Thus, we apply the mentioned algorithms to tackle the complexity of the problem. As mentioned before, an application of a PSO algorithm is also applied due to its rapid convergence and on the other hand due to the long running time of the other algorithms.

## 3- Notations and formulation

### 3-1- Assumptions

We introduce the following assumptions for the problem:

(1) The processing times and setup times of the jobs on each machine at each stage and their due-dates are deterministic and known in advance.
(2) The number of jobs, the number of machines and the number of stages are fixed.
(3) Preemption is not allowed.
(4) Once a job is taken in to processing, it must finish its processing completely before moving to the next stage.
(5) No splitting is allowed. It means that a job must be processed just on one machine at each stage.
(6) The jobs' release dates in the first stage are all zero.
(7) Setup times are sequence-dependent. There is also a setup time before starting the first job in the permutation of jobs in the first stage.
(8) Machines are always available without any breakdown.
(9) The machines at each stage are unrelated and the processing time of each job is calculated by dividing its processing time by the machine speed.

## 3-2- Notations

To describe the mathematical model, following notations are introduced:

$k,j$=Job index
$J$=Total number of jobs
$s$ =Stage index
$m$=Machine index
$M_s$=Number of machines in stage $s$
$P_{sjm}$=Processing time of job $j$ on machine $m$ in stage $s$
$sup_{skjm}$=Setup time from job $k$ to job $j$ on machine m in stage $s$
$M$=A large positive number
$S$=Total number of stages
$C_{sjm}$=Completion time of job $j$ on machine $m$ in stage $s$
$K_{sj}$=Departure time of job $j$ from stage $s$

$$X_{sjm} = \begin{cases} 1 & \text{, If job } j \text{ is processed on machine } m \text{ in stage } s, \\ 0 & \text{, Otherwise.} \end{cases}$$

$$W_{skjm} = \begin{cases} 1 & \text{, If job } j \text{ is processed directly after job } k \text{ on machine m in stage } s, \\ 0 & \text{, Otherwise.} \end{cases}$$

$d_j$=due-date of job $j$
$E_j$=earliness of job $j$
$T_j$ =tardiness of job $j$

## 3-3- Mixed integer linear programming model

In this section, we introduce a mixed integer linear programming model for the problem which is derived and developed from Crowder (2006):

$$\text{Min}Z = \sum_{j=1}^{J} (E_j + T_j) \tag{1}$$

s.t:

$$K_{sj} + E_j - T_j = d_j \quad , \forall j, s = S; \tag{2}$$

$$E_j \geq d_j - K_{sj} \quad , \forall j, s = S; \tag{3}$$

$$T_j \geq K_{sj} - d_j \quad , \forall j, s = S; \tag{4}$$

$$C_{sjm} \geq P_{sjm} + W_{sijm} * sup_{sijm} \quad , \forall j, i \neq j, m = 1,..,M_s, s = 1; \tag{5}$$

$$C_{sjm} - C_{sim} + M*(1 - W_{sijm}) \geq P_{sjm} + W_{sijm} * sup_{sijm} \quad , \forall s \neq 1, j, i \neq j, m = 1,..,M_s; \tag{6}$$

$$C_{sjm} - K_{(s-1)j} \geq P_{sjm} - M * (1 - X_{sjm}) + W_{sijm} * sup_{sijm} \quad , \forall s \neq 1, j, i \neq j, m = 1, \ldots, M_s ; \tag{7}$$

$$C_{sjm} \leq K_{sj} \quad , \forall s, j, m = 1, \ldots, M_s ; \tag{8}$$

$$C_{sjm} \geq K_{sj} - M * (1 - X_{sjm}) \quad , \forall s, j, m = 1, \ldots, M_s ; \tag{9}$$

$$\sum_{m=1}^{M_s} X_{sjm} = 1 \quad , \forall s, j \neq 0; \tag{10}$$

$$X_{sjm} - \sum_{j=1}^{J} W_{sijm} = 0 \quad , i \neq j, \forall s, j \neq 0, m = 1, \ldots, M_s ; \tag{11}$$

$$X_{sim} - \sum_{j=1}^{J} W_{sijm} \geq 0 \quad , i \neq j, \forall s, i, m = 1, \ldots, M_s ; \tag{12}$$

$$\sum_{i=1}^{J} W_{si0m} = 0 \quad , i \neq j, \forall s, m = 1, \ldots, M_s ; \tag{13}$$

$$X_{s0m} = 1 \quad , \forall s, j, m = 1, \ldots, M_s ; \tag{14}$$

$$T_j \geq 0 \tag{15}$$

$$E_j \geq 0 \tag{16}$$

The pairs of constraints (3), (16) and (4), (15) assure the proper minimization of the earliness and tardiness and constraint (2) reflects the earliness and tardiness for each job with respect to due-date. Constraint (5) guarantees that the completion time of job *j* be greater than or equal to the processing and setup time of job *j* in the first stage. With constraint (6), overlapping processing of jobs on the same machine in a stage is prevented. It also assures that the jobs are not interrupted during processing. Constraint (7) is used to ensure that the completion time of each job is greater than or equal to the completion time of the job in the previous stage plus the processing and setup time of the job in the current stage. Constraints (8) and (9) set the value for $K_{sj}$ as the time that job *j* leaves stage *s*. Constraint (10) is used to ensure that each job is processed on one and only one machine per stage. Constraint (11) assures that each job must follow another job *i*. Constraint (12) stipulates each job *i* might or might not be followed by another job, but it can be followed by at most one job. In constraints (13) and (14), in order to calculate the setup times for the first job processed on each machine, job 0 is considered to be processed first on every machine in each stage with zero processing time. This job does not

really exist and is a dummy job. For instance, if job 5 is the first job that is taken in to process, job 0 is considered before job 5 and the setup time required is denoted by $sup_{s05m}$.

## 4- The proposed hybrid memetic algorithm

The most important advantage of a GA is its capability of parallel search in the search space. It starts with a set of possible solutions, called population which can be generated randomly or by using some heuristics. Each individual in the population is called chromosome which is made up of genes. The individuals in the population are evaluated by fitness measure. In each generation, two types of genetic operators called crossover and mutation are used for evolution of the current population. A genetic algorithm hybridized with local search methods is called a memetic algorithm. Due to the fact that the performance of the simple genetic algorithm (GA) can be weak in this problem it is hybridized with a local search to enhance its characteristics. The resultant algorithm is called a hybrid memetic Algorithm (HMA).

Notice that the HMA as well as the HSA algorithm and the proposed PSO algorithm are used to determine the first stage sequence. This sequence will then be used as an entry to the heuristic algorithm to construct the schedule for the overall problem. The heuristic will be discussed later in section 4.3.

### 4-1- Representation

To solve a problem using MA or GA, the decision of how to represent a solution is a prerequisite action. Using job permutation in the first stage is common and straightforward in many previous works in GAs for flow-shop scheduling problems. As an example, for five jobs, this representation can be encoded as [5 2 1 4 3]. (Jungwattanakit et al., 2005, 2008 and 2009)

### 4-2- Initialization

Initial population can be generated randomly or by using some heuristics. We use random generation to initialize the population which is more common.

### 4-3-The heuristic to calculate fitness function

For each individual in the population, a heuristic algorithm is applied to find the complete schedule for the overall problem and to calculate the cost function whenever is required.

The proposed heuristic algorithm consists of two parts: The first part of the algorithm aims at completing the jobs as soon as possible and in the second part of the algorithm, starting and completion times of the jobs in the last stage is recalculated in order to minimize the sum of earliness and tardiness. The pseudo code of this heuristic is as follows:

- The first part of the algorithm:

Input: The first stage sequence

Set the release date of each job at the first stage to be zero. $(r(s, j) = 0, s = 1, j = 1,..., J)$

For stage 1 to *S*:

- If *s*=1
  - Consider the first stage sequence
- Else if *s*<>1

- Determine the sequence of the jobs in stages by using the FIFO rule.

– End

– For every machine *m* in stage *s*, set the available time of each machine to be zero: $AV(m,s)=0, \forall m=1,...,M_S$

For job 1 to *J*:

– Let $C_{sjm}$ be the completion time of job *j* on machine *m* in stage *s*, $sup_{s*jm}$ be the setup time of job *j* provided that the previous job is changed to job *j* and $P_{sjm}$. Calculate the completion time of each job on each machine in the current stage as follows, and then choose the machine with minimum completion time.

$$C_{sjm} = \max\{AV(m,s), r(s,j)\} + P_{sjm} + \sup_{s*jm};$$

Name the selected machine as $m^*$.

– Update the available time of the selected machine in stage *s*and release date of job *j* in stage *s*.

– Store the completion time of job *j*, $C_j$ as $C_{sjm^*}$.

– Calculate the sum of earliness and tardiness.

End

End

- The second part of the algorithm:

In this stage of the algorithm, the starting times of the jobs on the machines in the last stage are modified regarding to the due-dates in order to minimize earliness-tardiness.

For the last stage and each machine:

Let $\pi$ be the permutation of jobs scheduled to be processed on machine *m* $\pi = \{\pi_{m1}, \pi_{m2}, ...., \pi_{mj}, ...\}$) and $B_j$ denote the group of jobs to be processed consecutively on machine *m* after *j*th job in the permutation without any idle time and consider $LB_j$ be the position of the last operation in $B_j$ and $EB_j \subseteq B_j$ denote the subset of early jobs in $B_j$ and $ID(B_j)$ denote the machine idle time after group $B_j$. In addition, consider $SEB_j$ to be the smallest earliness of early jobs in the group.
There are two conditions which should be considered:

(1) $E(\pi_{mj}) \succ 0$ (if the *j*th job is early) ,

(2) Number of early jobs > number of tardy jobs

If these two conditions are satisfied, the group $B_j$ together with the *j*th job is shifted to right by time *t* which is calculated using the following equation: $t = \min\{SEB_j, E(\pi_{mj}), ID(B_j)\}$

Then, group $B_j$ and the values of *E*, $SEB_j$ and $ID(B_j)$ must be updated and earliness-tardiness penalty is recalculated. This procedure will be continued until one of the conditions is violated.

### 4-4-Crossover

In this algorithm, two kinds of crossover operators are used, named as the combined order and position-based crossover (OPX crossover) presented by Jungwattanakit et al. (2009) and a straightforward and simple crossover called Adapted Single Point crossover (ASP crossover) which is the adapted version of the single point crossover used in Random Key Genetic Algorithm (RKGA) or Binary Genetic Algorithm (BGA).

To apply ASP crossover, two individuals are selected using roulette wheel selection, then one position in the chromosome is randomly chosen. Child one is the head of parent one and the rest of the child structure is completed by inserting the remaining jobs in the same order as the second parent. In the same way, child two is the head of parent two and the child structure is completed by inserting the remaining jobs in the same order as the first parent. Figure 2 depicts how ASP crossover works.

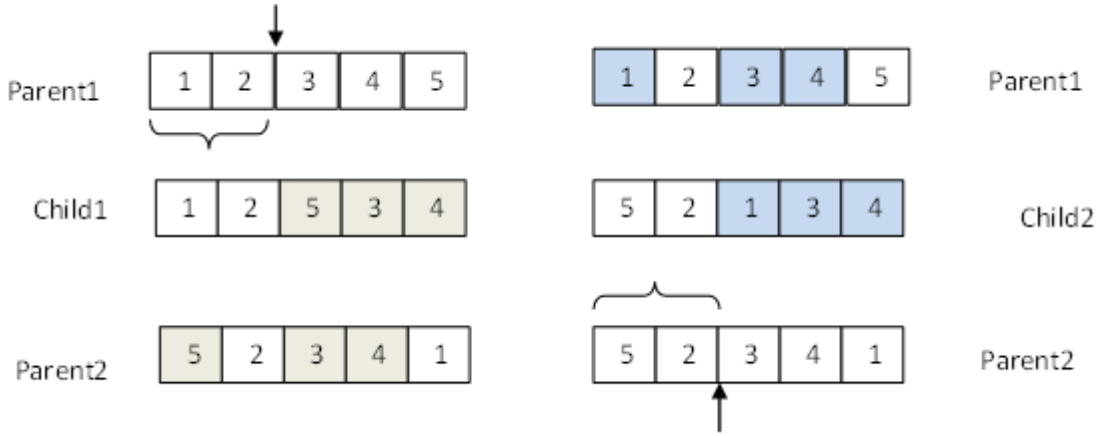



**Figure2.**ASP Crossover

### 4-5-Mutation

Mutation is an operator which is used in MA and GA to avoid getting trapped in local optimum. In this paper, three mutation operators, called insertion, inversion and swap mutation are considered which are widely used in the literature.

### 4-6-Termination criterion

The cost function does not change after a number of iterations. Hence, a termination criterion is required. In this paper, the simplest criterion is considered which is "stop after some predetermined number of generations".

### 4-7- Proposed local search procedure

As mentioned before, few experiments show that the performance of simple GA can be weaker than other similar algorithms. Thus, we apply a local search in each generation of the algorithm and to the solution found at the end of the algorithm.

To apply the local search in each generation, a number of individuals are selected at random from the population. The local search procedure is first applied to the individuals (the schedule of the first stage) with the following neighbourhoods and then it is applied to the schedule of the last stage attained by the heuristic method:

(1) A machine and two jobs processed on it are selected at random and the jobs are interchanged.

(2) A machine and a job processed on it are selected at random. The selected job is inserted in another random position on the selected machine.

(3) Two machines in parallel and one job processed on each of them are selected randomly. Then the jobs are interchanged.
(4) Two machines are selected at random and a job on the first machine is selected and inserted in a random position on the second machine.

## 5- An improved hybrid Simulated annealing

In this section, a SA algorithm which is a popular local search metaheuristic is hybridized with the local search method presented in the previous section. Representation, evaluation and other parts of the algorithm except initialization and the use of the local search method are similar to the HMA.

For initialization, three methods called random initialization, NEH heuristic and a new EDD-based heuristic are considered. The EDD-based heuristic is chosen among other methods by applying Taguchi method. The so-called EDD-based heuristic is described as follows:

- Determine the sequence of jobs according to the EDD rule.
- Interchange the first two jobs.
- Choose the sequence with minimum cost function.
- For $i$=1:$J$
  - Insert the next job between previous jobs and choose the sequence with minimum cost function.
- End

Finally, the local search method used in the HMA is applied just on the solution found by this algorithm. The termination criterion is also as the same as the one used in the HMA.

## 6- The proposed Particle Swarm Optimization

Another algorithm is a PSO algorithm hybridized with genetic operators (called GPSO).Notice that the representation of this algorithm is different from the algorithms mentioned in the previous sections and is derived from Tasgetirenet al. (2004) which enables the original continuous particle swarm optimization. Thus, in order to choose the best crossover and mutation operators, in addition to the operators used in the HMA, a new mutation operator is considered called the regeneration mutation. By applying this mutation, two positions are chosen within the particle position at random. Then, two random numbers are regenerated and placed within the selected positions. In this way, a new particle position is generated.

The termination criterion is the same as the previous algorithms and the heuristic in section 4.3 is used to obtain the schedule for the overall problem.

## 7- Experiments

### 7-1- Parameter calibration

Before presenting and comparing the results of our algorithms, the best parameter combination must be chosen for each algorithm. Hence, we perform a Taguchi design to set the parameters.

For the HMA, 10 is selected for the population size(*npop*), inverse/insertion/swap for mutation type, OPX/ASP crossover for crossover type, 0.4 for mutation rate ($P_m$), 0.9 for crossover rate ($P_c$), 0.2 for local search rate($P_{local\ search}$) and10 for neighbourhood length(*Inner N*

*length*). Figures 3 and 4 which are the main effects plot for SN ratios and main effects plot for means show these results.

In the same way for the HSA, EDD-based heuristic is selected for generating initial solution, 10 for neighbourhood length, $\left(T_f / T_0\right)^{1/MaxIt}$ [3] for temperature reduction rate, 500 for initial temperature, and 20 for maximum number of iterations per temperature.

For the GPSO, 40 is selected for initial population size, inverse and regeneration mutation for mutation type, OPX for crossover type, 0.3 for mutation rate, 0.7 for crossover rate, 2 for $C_1$and $C_2$ ,0.99 for$W_{damp}$ , (-*J*,*J*) for accepted range of particle position and (-2*J*,2*J*) for accepted range of particle velocity.

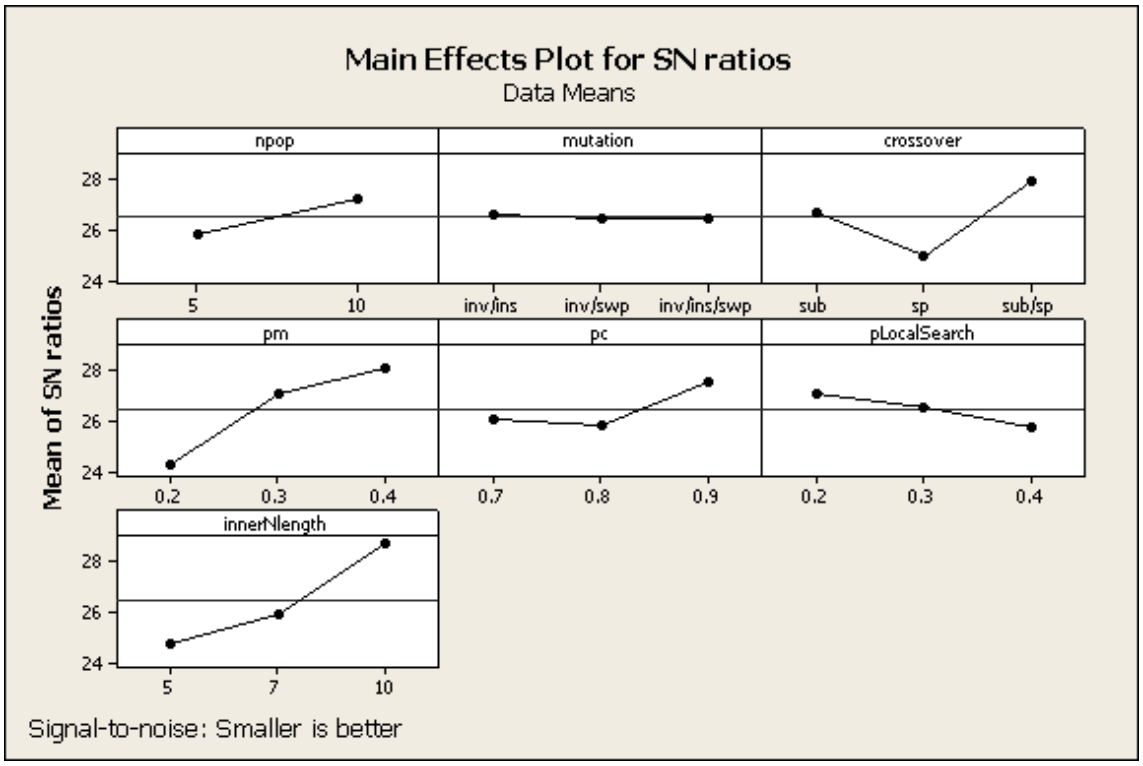



**Figure3.**Main Effects Plot for SN ratios for the HMA

[3]*MaxIt* is the maximum number of iterations

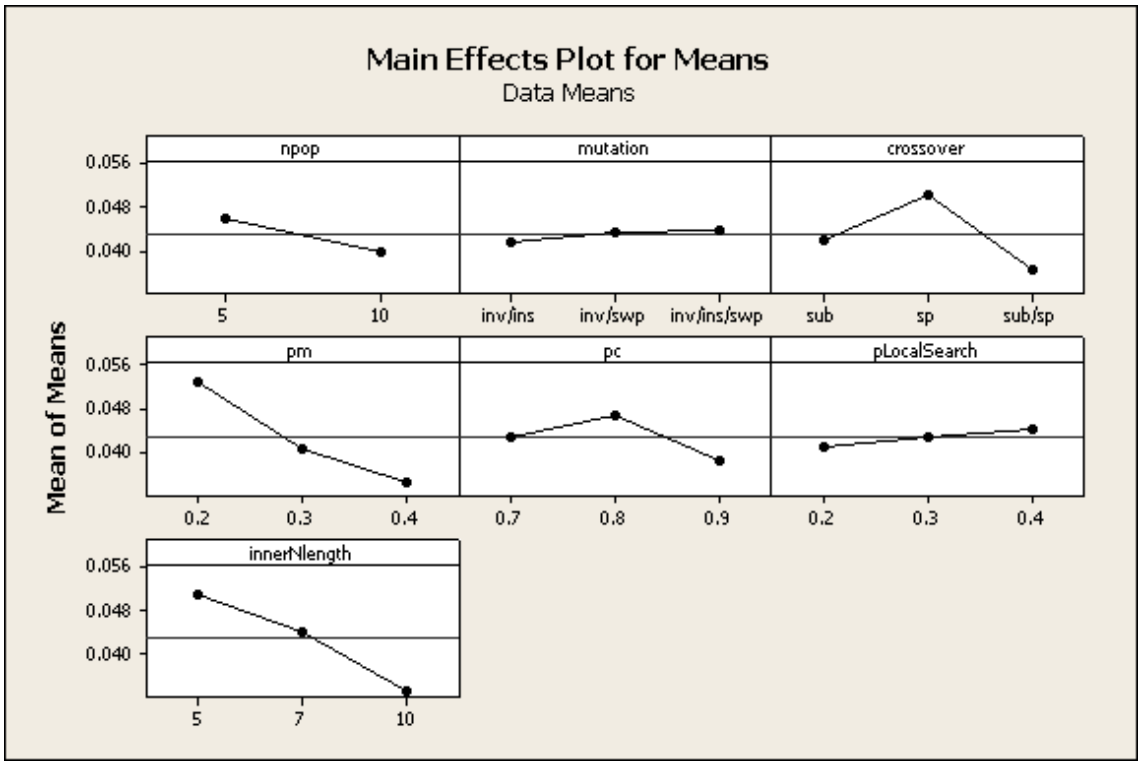



**Figure4.**Main Effects Plot for Means for the HMA

## 7-2- Data generations and settings

Ultimately, an experiment is conducted to test the performance of the proposed algorithms. Data required for the problem consists of the number of jobs, the number of stages, the number of machines in each stage, the value of the processing times, setup times which are generated according to Jung wattanakit et al. (2005, 2008 and 2009).

## 7-3- Performance analysis

All methods presented above except for the GPSO algorithm produce acceptable results with regard to the computational time in comparison with the exact method. GA performs weaker than SA in this problem and even worse than the GPSO algorithm. Hence, the performance of the proposed algorithms is compared with the MIP model for small size problems and with the HSA both for small and large size problems.

While an optimal solution is obtainable using the MIP model, for medium and large size problems, solutions are very difficult to find. Hence, this paper compares the performance of all proposed metaheuristics by using the relative performance deviation (RPD) which is calculated as follows:

$$RPD = \left(sol - min_{sol}\right) / min_{sol} \tag{17}$$

Where $min_{sol}$ is the minimum function value obtained by all algorithms and *sol* is the solution obtained by a particular of the proposed metaheuristics. The results are demonstrated in tables 1 and 2.

### 7-3-1- Small size problems

For small size problems, 16 types of problems are generated and are executed 10 times in order to reduce error of random numbers. Then, averages of results are reported.

Table 2 gives the average of the obtained results for small problems. The first item of the second column gives the number of jobs and the second item gives the number of stages (the number of jobs*the number of stages). The next columns give the results and the CPU time of the algorithms.

The results demonstrated in table 2show that the GAMS take large CPU time even in small size problems. According to table 2 and figure 5, the results of different algorithms are quite comparable; however, GPSO has attained the worst solution in some problems.

**Table 2.** A comparison between the MIP model and the proposed algorithms for small problems

| problem | | GAMS | | HMA | | HSA | | GPSO | |
|---|---|---|---|---|---|---|---|---|---|
| No. | size | Obj. | time | Obj. | time | Obj. | time | Obj. | time |
| 1 | 3*5 | 0 | 0.16 | 0 | <0.07 | 0 | <0.04 | 0 | <0.1 |
| 2 | 4*4 | 179 | 5.26 | 181 | <0.19 | 181 | <0.34 | 181 | <0.11 |
| 3 | 4*3 | 9 | 214.816 | 16 | <2.48 | 16 | <0.38 | 16 | <0.14 |
| 4 | 3*5 | 95 | 3.099 | 95 | <0.067 | 95 | <27.9 | 100 | <0.1 |
| 5 | 4*2 | 0 | 0.249 | 0 | <0.15 | 0 | <0.35 | 0 | <0.12 |
| 6 | 3*4 | 8 | 0.733 | 10 | <0.19 | 10 | <0.36 | 10 | <0.12 |
| 7 | 5*5 | 0 | 5.417 | 0 | <0.65 | 0 | <0.76 | 0 | <0.25 |
| 8 | 3*5 | 6 | 16.660 | 10 | <0.86 | 10 | <0.42 | 10 | <0.16 |
| 9 | 4*5 | 0 | 13.221 | 0 | <0.28 | 0 | <57.4 | 0 | <0.21 |
| 10 | 3*5 | 0 | 0.452 | 0 | <0.37 | 0 | <41.9 | 0 | <0.15 |
| 11 | 4*4 | 19 | 1.829 | 20 | <0.14 | 20 | <47 | 22 | <0.16 |
| 12 | 4*4 | 19 | 3.674 | 19 | <0.37 | 19 | <0.42 | 19 | <0.17 |
| 13 | 5*3 | 8 | 1.755 | 8 | <0.37 | 8 | <0.42 | 8 | <0.16 |
| 14 | 3*4 | 0 | 0.749 | 0 | <0.079 | 0 | <0.37 | 0 | <0.12 |
| 15 | 5*5 | 259 | 319.631 | 263 | <20.23 | 263 | <57.48 | 269 | <0.96 |
| 16' | 5*5 | 39 | 2070.75 | 24 | <0.64 | 24 | <0.72 | 24 | <0.31 |
| 17' | 5*5 | 12 | 2063.7 | 12 | <0.57 | 12 | <0.64 | 12 | <0.2 |

### 7-3-2- Large size problems

In this section, 24 medium and large size problems differing in the parameters are generated

and are executed 10 times and averages of results are shown in table 3.Then, the average of the RPDs and the CPU time are used for comparison as can be seen in figure 6. Moreover, we apply an analysis of variance (ANOVA) with 95% confidence level in which different algorithms are considered as factors and averaged *RPD* as a response variable in order to analyse the results more precisely.

**Table 3.** A comparison between the averaged RPD and the average running time of different algorithms for medium and large size problems.

| instance | algorithm HMA | | HSA | | GPSO | |
|---|---|---|---|---|---|---|
| | $\overline{RPD}$ | Average time | $\overline{RPD}$ | Average time | $\overline{RPD}$ | Average time |
| $5*5_1$ | 0 | 118.2700 | 0.003 | 203.1037 | 0.03 | 30.2429 |
| $5*5_2$ | 0.22 | 113.3889 | 0 | 191.9849 | 0.023 | 28.4919 |
| $5*10_1$ | 0 | 222.4250 | 0.0087 | 300.2958 | 0.012 | 45.2471 |
| $5*10_2$ | 0 | 219.8804 | 0 | 352.269 | 0 | 44.6354 |
| $5*20_1$ | 0 | 458.1384 | 0.0087 | 11967.5 | 0.023 | 89.6812 |
| $5*20_2$ | 0.12 | 484.1380 | 0.095 | 14421.5 | 0.135 | 101.9415 |
| $10*5_1$ | 0.008 | 216.1408 | 0 | 332.8105 | 0.022 | 50.734 |
| $10*5_2$ | 0.016 | 228.5433 | 0.03 | 378.2373 | 0.12 | 57.1036 |
| $10*10_1$ | 0.053 | 460.3810 | 0.04 | 671.25 | 0.086 | 100.0733 |
| $10*10_2$ | 0.055 | 444.1171 | 0.03 | 618.1 | 0.064 | 92.9471 |
| $10*20_1$ | 0.028 | 1173.2 | 0.016 | 1112.05 | 0.026 | 168.0529 |
| $10*20_2$ | 0.014 | 685.4353 | 0.011 | 1265.7 | 0.04 | 184.3565 |
| $30*5_1$ | 0.075 | 1586 | 0.037 | 2085.707 | 0.099 | 271.7298 |
| $30*5_2$ | 0.039 | 1607.2 | 0.033 | 2210.773 | 0.058 | 291.8483 |
| $30*10_1$ | 0.05 | 3674.6 | 0.046 | 4550.027 | 0.073 | 610.0317 |
| $30*10_2$ | 0.018 | 1977.5 | 0.016 | 4312 | 0.039 | 457.1351 |
| $30*20_1$ | 0.061 | 8047.6 | 0.032 | 13030.4 | 0.056 | 1022.5 |
| $30*20_2$ | 0.038 | 7411.3 | 0.028 | 8739.733 | 0.056 | 938.3468 |
| $50*5_1$ | 0.044 | 2683.1 | 0.026 | 4002.56 | 0.057 | 526.2653 |
| $50*5_2$ | 0.056 | 2965.4 | 0.032 | 4252.587 | 0.045 | 556.2132 |
| $50*10_1$ | 0.049 | 6107.3 | 0.024 | 7573.867 | 0.033 | 1052.301 |
| $50*10_2$ | 0.022 | 3232.3 | 0.025 | 6817.067 | 0.051 | 912.1478 |
| $50*20_1$ | 0.0185 | 6417.9 | 0.017 | 12765.33 | 0.034 | 3126.45 |
| $50*20_2$ | 0.048 | 12782 | 0.024 | 1538.293 | 0.063 | 2321.5 |
| **Average** | **0.035** | **2638.177** | **0.024** | **4320.548** | **0.052** | **544.999** |

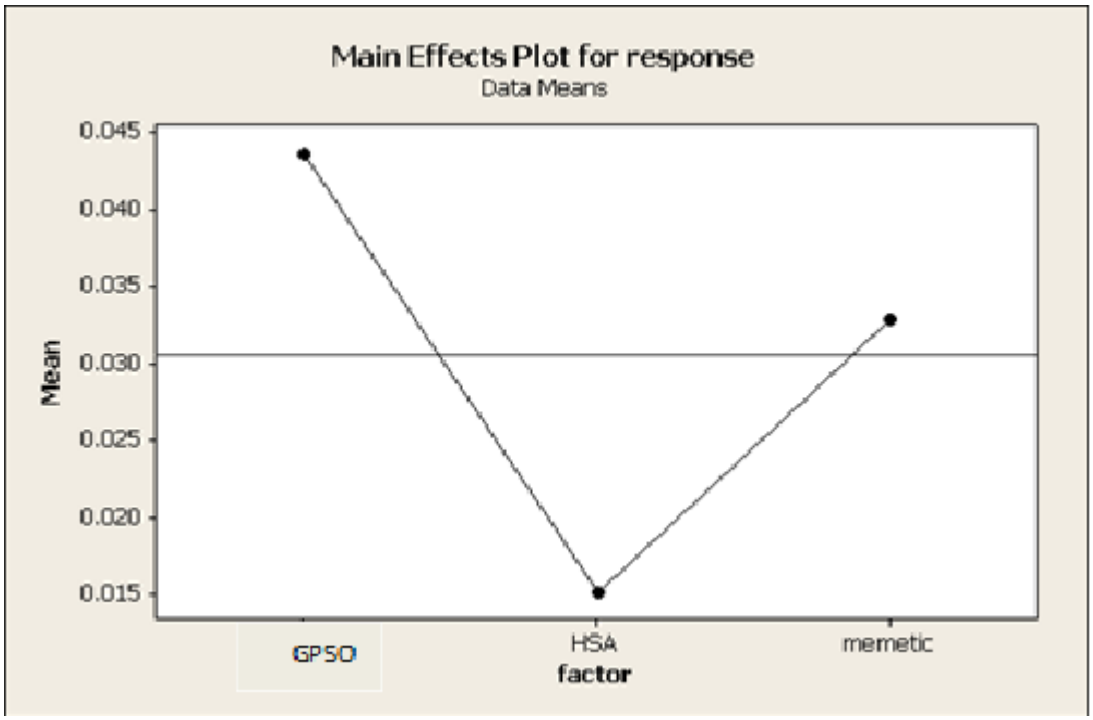



**Figure 5.** Main Effects Plot for response

To conclude, The HSA algorithm outperforms other algorithms qualitatively and the GPSO is the fastest algorithm; however, the results show that the GPSO gets trapped in local optima and using genetic operators does not seem to improve the results significantly. The results of one-way ANOVA and the Tukey method which is shown in figure 7, demonstrate that there is a large discrepancy between the HSA and the GPSO algorithm**.** Although a difference between the HSA and the HMA is spotted in the main effects plot for means, the difference is not significant according to the Tukey method. On the other hand, the HMA performs 63.77% faster than the HSA. Therefore, the HMA is preferred.

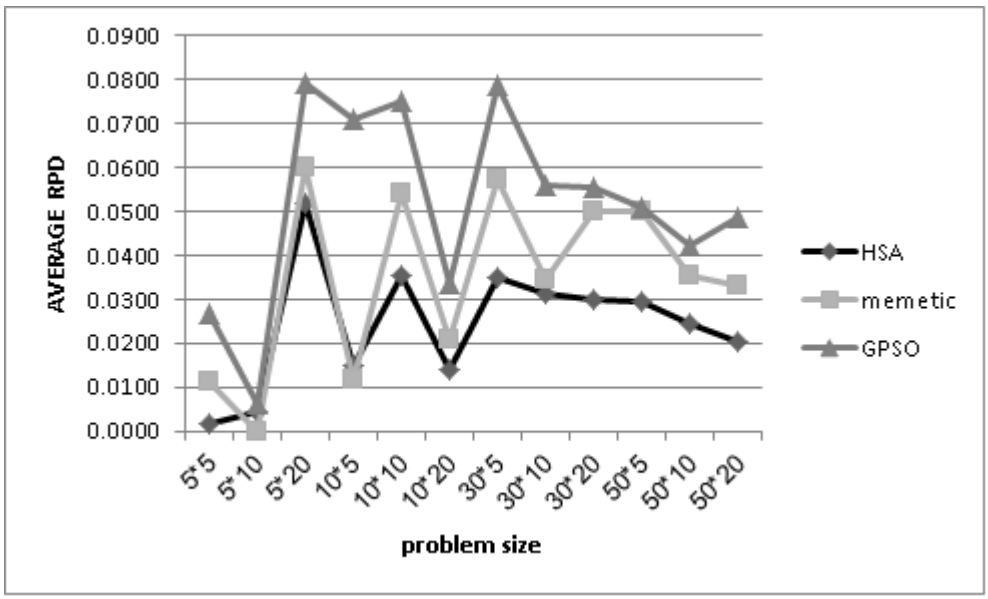
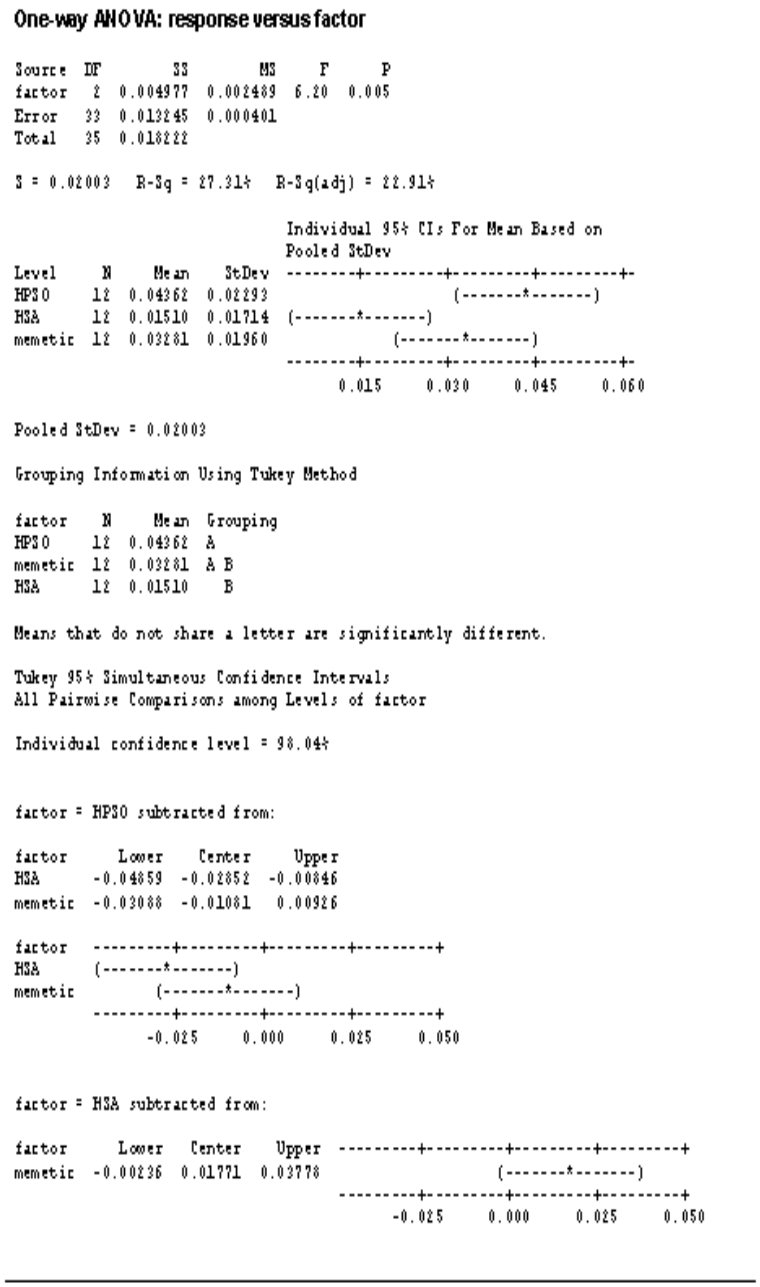



**Figure 6.**Comparison of averaged RPD versus problem size



One-way ANOVA: response versus factor

```
Source  DF        SS        MS     F      P
factor   2  0.004977  0.002489  6.20  0.005
Error   33  0.013245  0.000401
Total   35  0.018222

S = 0.02003   R-Sq = 27.31%   R-Sq(adj) = 22.91%

                              Individual 95% CIs For Mean Based on
                              Pooled StDev
Level     N     Mean    StDev --------+---------+---------+---------+-
HPSO     12  0.04262  0.02293                    (-------*-------)
HSA      12  0.01510  0.01714  (-------*-------)
memetic  12  0.03281  0.01960             (-------*-------)
                              --------+---------+---------+---------+-
                                    0.015     0.030     0.045     0.060

Pooled StDev = 0.02003

Grouping Information Using Tukey Method

factor    N     Mean  Grouping
HPSO     12  0.04262  A
memetic  12  0.03281  A B
HSA      12  0.01510    B

Means that do not share a letter are significantly different.

Tukey 95% Simultaneous Confidence Intervals
All Pairwise Comparisons among Levels of factor

Individual confidence level = 98.04%


factor = HPSO subtracted from:

factor      Lower    Center     Upper
HSA      -0.04859  -0.02852  -0.00846
memetic  -0.03088  -0.01081   0.00926

factor   ---------+---------+---------+---------+
HSA      (-------*-------)
memetic          (-------*-------)
         ---------+---------+---------+---------+
               -0.025     0.000     0.025     0.050


factor = HSA subtracted from:

factor      Lower   Center    Upper  ---------+---------+---------+---------+
memetic  -0.00236  0.01771  0.03778                    (-------*-------)
                                    ---------+---------+---------+---------+
                                          -0.025     0.000     0.025     0.050
```

**Figure 7 .**The results of one-way ANOVA and Tukey method

## 8- Conclusion

This paper is dedicated to study hybrid flow-shop scheduling problem with unrelated parallel machines per stage with sequence-dependent setup times. The objective is to minimize the sum of earliness and tardiness. For each job, the due-date, the processing times and setup times are fixed. The problem is formulated as a MIP model. Then, for this NP-hard problem, an application of GA and a GPSO algorithm are approached. The GPSO is a PSO algorithm combined with genetic operators. Just a few experiments show that a simple GA does not give near optimal solutions, even by applying the local search to the best solution. So that the local search is used both in each generation and on the best answer found by the MA algorithm which

is a GA combined with a local search. Moreover, an HSA algorithm which is widely used in the literature is developed in order to validate the results. The HSA starts with an initial solution generated by the proposed EDD-based heuristic and the local search is applied on the solution found at the end of the algorithm. In all the algorithms presented above, a heuristic algorithm is also applied to find the complete schedule for the overall problem. Finally, the results of one way ANOVA demonstrate that there is not much difference between the results of the proposed HMA and the HSA algorithm. Furthermore, the elapsed time for the HMA is 66.73% less. Thus, the HMA is superior in our test problems.

## References

Acosta, J. H. T., González, V. A. P., & Bello, C. A. L. (2013). A Genetic Algorithm for Simultaneous Scheduling Problem in Flexible Flow Shop Environments with Unrelated Parallel Machines, Setup Time and Multiple Criteria. *International Conference on Advanced Manufacturing Engineering and Technologies*. 203-210.

Attar, S.F., Mohammadi, M., & Tavakkoli-Moghaddam, R. (2009). Hybrid flexible flow-shop scheduling problem with unrelated parallel machines and limited waiting times.*Int J Adv Manuf Technol*, 68, 1583-1599.

Behnamian, j., Ghomi, S. M. T. F., & 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 Syst Appl, 35*, 11057-11069.

Behnamian, J., & Zandie, M. (2011). A discrete colonial competitive algorithm for hybrid flowshop scheduling to minimize earliness and quadratic tardiness penalties. *Expert Syst Appl, 38*, 14490-14498.

Behnamian, J., & Zandie, M. (2012).Earliness and Tardiness minimizing on a realistic hybrid flowshop scheduling with learning effect by advanced metaheuristic.*Arab J Sci Eng, 38*, 1229-1242.

Crowder, B. (2006). *Minimizing the Makespan in a Flexible Flowshop with Sequence Dependent Setup Times, Uniform Machines, and Limited BuffersMinimizing the Makespan in a Flexible Flowshop with Sequence Dependent Setup Times, Uniform Machines, and Limited Buffers.* West Virginia University, virginia.

Dai, M., Tang, D., Zheng, K. & Cai, Q. (2013). An improved Genetic-Simulated Annealing Algorithm based on a Hormone Modulation Mechanism for a flexible flow-shop scheduling problem. *Advances in Mechanical Engineering*, 2013, 13 pages, doi:10.1155/2013/124903.

Davoudpour, H., & Ashrafi, M. (2009). Solving multi -objective SDST flexible flow shop using GRASP algorithm. *Int J Adv Manuf Tech, 44*, 737-747.

Farkhzad, M. B., & Heydari, M. (2008). A heuristic algorithm for hybrid flow-shop production scheduling to minimize the sum of the earliness and tardiness costs. *JCIIE, 25*, 105-115.

Gupta, J. N. D. (1988). Two-stage, hybrid flowshop scheduling problem. *J Oper Res Soc, 39*, 359-364.

Haddad, M.N., Cota, L.P., Souza, M.J.F. & Maculan, N. (2014). A Heuristic Algorithm based on Iterated Local Search and Variable Neigbourhood Descent for solving the unrelated parallel machine scheduling problem with setup times. *Proceedings of the 16th International Conference on Enterprise Information Systems*, 376-383.

Janiak, A., Kozan, E., Lichtenstein, M., & Eguz, C. (2007). Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion. *Int J Prod Econ, 105*(2), 407-424.

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. *International Journal of Production Research*, 50(24), 7447-7466.

Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2005). An Evaluation of Sequencing Heuristics for Flexible Flowshop Scheduling Problems with Unrelated Parallel Machines and Dual Criteria. *Otto-von-Guericke-Universitat Magdeburg, 28*(5), 1-23.
Jungwattanakit, J., Reodecha, M., Chaovalitwongse, P., & Werner, F. (2008). Algorithms for flexible flow shop problems with unrelated parallel machines, setup times, and dual criteria. *Int J Adv Manuf Tech, 37*, 354-370.

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. *Comput Oper Res, 36*(2), 358-378.

Kurz, M. E., & Askin, R. G. (2004). Scheduling flexible flow lines with sequence-dependent setup times. *Euro JOper Res, 159*, 66-82.

Liao, C.-J., Tsengb, C.-T., & Luarn, P. (2007). A discrete version of particle swarm optimization for flowshop scheduling problems. *Comput Oper Res, 34*, 3099-3111.

Naderi, B., M. Z., , A. K. G. B., & , V. R. (2009). An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. *Expert Syst Appl, 36*(6), 9625–9633.

Nahavandi, N., & Gangraj, E. A. (2014). A New Lower Bound for Flexible Flow Shop Problem with Unrelated Parallel Machines. *International Journal of Industrial Engineering, 25*(1), 65-70.

Niu, Q., Jiao, B., & Gu, X. (2008). Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time. *Appl Math Comput, 205*, 148-158.

Rabiee, M., Rad, R. S., 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*(5-8), 1229-1245.

Rashidi, E., Jahandar, M., & Zandieh, M. (2010). An improved hybrid multi -objective parallel genetic algorithm for hybrid flow shop scheduling with unrelated parallel machines. *Int J Adv Manuf Tech, 49*, 1129-1139.
Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *Eur J Oper Res, 169*, 781-800.

Ruiz, R., & Vázquez-Rodríguez, J. A. (2010). The hybrid flow shop scheduling problem. *Eur J Oper Res, 205*, 1-18.

Soltani, S. A., & Karimi, B. (2014). Cyclic hybrid flow shop scheduling problem with limited buffers and machine eligibility constraints. *The International Journal of Advanced Manufacturing Technology*, 1-17.

Tasgetiren, M. F., Liang, Y.-C., Sevlili, M., & Gencyilmaz, G. (2004). *Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem*. Paper presented at the 2004 IEEE.

Urlings, T., Ruiz, R., & Şerifoğlu, F. S. (2010). Genetic algorithms with different representation schemes for complex hybrid flexible flow line problems. *IJMHeur, 1*(1), 30-54.

Yaurima, V., Burtseva, L., & Tchernykh, A. (2009). Hybrid flowshop with unrelated machines, sequence-dependent setup time, availability constraints and limited buffers. *Comput Oper Res, 56*(4), 1452–1463.

Zandieh, M., FatemiGhomi, S. M. T., & MoattarHusseini, S. M. (2006). An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. *Appl Math Comput, 180*(1), 111-127.

Zandieh, M., Mozaffari, E., & Gholami, M. (2010). A robust genetic algorithm for scheduling realistic hybrid flexible flow line problems. *J Intell Manufa, 21*, 731-743.

Zhang, C., Ning, J., & Ouyang, D. (2010). A hybrid alternate two phases particle swarm optimization algorithm for flow shop scheduling problem. *Comput. Ind. Eng., 58*, 1-11.