An efficient simulated annealing algorithm for a No-Wait Two Stage Flexible Flow Shop Scheduling Problem

M. Rabiee[1], P. Ramezani[2] and R. Shafaei[3]

[1] *Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran*
meysam_rabiee@yahoo.com
[2]*Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran*
prh_79@yahoo.com
[3] *Department of Industrial Engineering, K. N. Toosi University of Technology, Tehran, Iran*
shafaei@kntu.ac.ir

ABSTRACT

In this paper, no wait two stage flexible flow shop scheduling problem (FFSSP) is solved using two meta-heuristic algorithms. This problem with minimum makespan performance measure is NP-Hard. The proposed algorithms are Simulated Annealing and Genetic Algorithm. The results are analyzed in terms of Relative Percentage Deviation of Makespan. The performance of the proposed algorithms are studied and compared with that of MDA algorithm. For this propose a number of problems in different sizes are solved. The results of the studies proposes the effective algorithm. This is followed by describing the outline of the study, concluding remarks and suggesting potential areas for further researches

KEYWORDS

No wait, Two stage, Flexible flow shop, Makespan, Simulated annealing, Genetic algorithm.

1. INTRODUCTION

Flexible flow shop is a typical classical flow shop, which consists of m stages each one with one or more than one identical parallel machines, n given jobs to be processed on m subsequent stages. . Flexible flow shop also known as a hybrid flow shop, flexible flow line or flow shop scheduling problem with parallel machines . This area has been studied by many researchers [1-4]. For a literature review in this area the readers are referred to those conducted by Ruiz Et al [5] and Ribas Et al [6].

A prominent class of scheduling problems is identified by a no-wait production environment, in which there is no storage between the machines. Thus, jobs must be processed from the start to finish, deprived of any interruption on machines or between them. Consequently, the processing of a job on the initial machine may need to be delayed to guarantee that no waiting takes place on any subsequent machine. There are two main motives for a no wait environment: the type of procedure, or a lack of storage between intermediate machines (work stations). 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 [11], food processing[12], concrete ware production [13], pharmaceutical processing [14] and production of steel, plastics, aluminum products [15].

DOI : 10.5121/ijait.2011.1402

These kinds of manufactures can often be modeled as a 'pure' flow shop environment in which all of n jobs follow the same sequence at each stage, each of n jobs consist of two operations owning a predetermined processing order through machines. Each job is to be processed without preemption and interruption on two stages or between them. That is, once a job is started on the first machine, it has to be continuously processed through the subsequent machine without interruption. In addition, each machine can handle no more than one job at a time and each job has to visit each machine exactly once. Therefore, it may occur a condition that the starting time of a job on the first machine must be delayed in order to meet the no-wait requirement [16]. As an example, in a chemical industry if the waiting time is allowed between each subsequent stage, it may lead to deteriorate the material property (e.g. degrading the polymer). Therefore, in such industries, once an operation is completed, the subsequent operation shall be started without any delay. For a further study in no wait manufacturing the readers are recommended to review the paper present by Hall and Sriskandarajah [12]

The problem studied in this paper is makespan minimization in a no wait two stages flexible flow shop scheduling problem as it has a multiplicity of practical implementations in both manufacturing industry and service corporations. This particular hybrid flow shop(i.e. two stage hybrid flow shop) scheduling problem with the objective of minimizing makespan is first considered in Riane et al. [7]. Narasimhan,S.& Mangiameli,P [8] also introduced a heuristic algorithm known as the Cumulative Minimum Deviation (CMD) rule for a two-stage hybrid flow shop scheduling problem with single machine in stage one and two independent machines in stage 2. The results of the study reveals that this algorithm outperforms SPT and LPT in reducing the machine idle time and in-process waiting time at the second stage.

Liu Zhixin et al [17] presented a heuristic algorithm named Least Deviation (LD) rule for two-stage no-wait hybrid flow shop scheduling with a single machine in each stage and makespan performance measure. The results of the study indicate that this algorithms is superior than L,D and Johnson. Also they shown LD algorithm has low computational complexity and is easy to implementation thus it is of favourable application value. Jinxing Xie et al. [18] 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 their study show that MDA outperformed partition method, partition method with LPT, Johnson's as well as the modified Johnson's algorithms.

Jinxing Xie and Xijun Wang [19] considers the two-stage flexible flow shop scheduling problem with availability constraints. They discussed the complexity and the approximability of the problem and provide some approximation algorithms with worst case performance bounds for some special cases of the problem. Their results showed that the problems studied are much more difficult to approximate than the case without availability constraints. Huang et al [20] 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 ACO results revealed that the efficiency of the proposed algorithm is superior to those solved by integer programming while having satisfactory solutions.

In this study a no wait flexible flow shop problem is solved using two methaheuristic algorithms based on Simulated Annealing and Genetic Algorithm. The aim is to find an effective algorithm with minimum makespan. The rest of the article is formed as follows: Section 2 briefly describes the problem addressed in this paper. Section 3 presents the framework of the proposed algorithms. Numerical problems developed to study the performance of the proposed algorithms are introduced in section 4. This is followed by presenting the results of simulation study. Finally Section 5 describes the outline of the research with the concluding remarks and suggestion for further researches.

2. PROBLEM DEFINITION

The no-wait flexible flow shop scheduling problem (NWFFSSP) can be described as follows: given the processing time p(i, j) of Stage *i* on Job *j*, each of n jobs will be sequentially processed in stage 1, 2 respectively. At each stage there are mi machines. In addition at any event of time, each machine can process at most one job. Similarly each operation of a job can only be processed on one machine. Once the sequence of the jobs at the first stage is defined, the same sequence is applied for the second stage. To fulfill the no-wait restrictions, the completion time of a job on a given machine must be equal to the start time of the job on the next machine. In other words, there must be no waiting time between the processing of any consecutive operations of each job. The problem is to find a sequence that the maximum completion time, i.e., makespan (C_{max}), is minimized.

Let $\pi = \{\pi_1, \pi_2, ..., \pi_n\}$ denote the schedule or permutation of jobs to be processed, $L(\pi_{j-1}, \pi_j)$ the minimum delay on the first machine between the start of job π_{j-1} and π_j restricted by the no-wait constraint. Then L can be calculated as follows:

$$L(\pi_{j-1}, \pi_j) = P(\pi_{j-1}, 1) + \max[0, \max_{2 \le k \le s}\{\sum_{h=2}^{k} p(\pi_{j-1}, 1) - \sum_{h=1}^{k-1} p(\pi_j, h)\}]. \quad (1)$$

Thus, the makespan can be defined as $C_{\max}(\pi) = \sum_{j=2}^{n} L(\pi_{j-1}, \pi_j) + P_{sum}(\pi_n)$ (2)

Where $P_{sum}(\pi_n) = \sum_{k=1}^{m} p(\pi_n, k).$

The no-wait FFSSP with the makespan criterion is to find a permutation π^* among the all permutations Π such that $\pi^* = \arg\{C_{\max}(\pi)\} \to \min \ \forall \pi \in \Pi.$ Where *arg* represents the arranged permutation.

The problem is shown by $F_2(m_1, m_2)$ |no-wait| C_{max} . If the number of parallel machines in each stage is not input variables, the problem can be shown by $F_2(p)$ |no-wait| C_{max} . In this study the operations' set up times are assumed to be independent from the job sequences and hence is added to the operation times. This problem is schematically depicted in Figure 1.

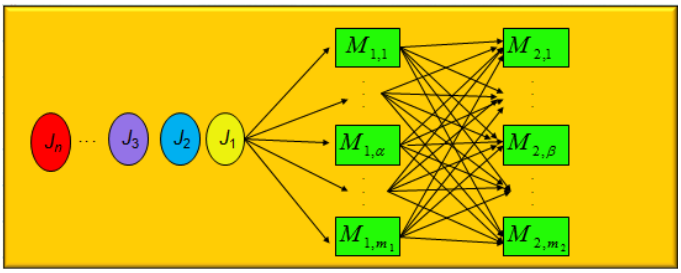

Figure 1. Schematic of the problem

As illustrated above, each job has $m_1 \times m_2$ possible schedules and hence n jobs have $n!$ possible schedules. Therefore in total there are $n! \times m_1^n \times m_2^n$ possible solutions for this problem [21]. The two stage no wait flexible flow shop problems are NP-hard in the strong sense [22]. Therefore, all exact algorithms for even a simple flow shop and simple parallel machines will

most likely have running times that increase exponentially with the problem Size. In this paper we propose metaheuristic algorithms to the problem described above. The framework of this algorithm is explained in the next section.

3. FRAMEWORK OF THE PROPOSED ALGORITHMS

The proposed algorithms in this study are based on GA and SA methods. They are categorized based on the algorithms applied for job sequencing and machine assignment. With reference to the algorithms presented in this paper, the job sequences are generated using the stochastic search process of SA and GA. The jobs are then allocated to the machines using a constructive heuristic algorithm which allocates the job with the first priority to the machine with the earliest available time.

In the remaining part of this section, first the schematic of a chromosome/ solution is presented. Then the structure of the Genetic Algorithm and Simulated Annealing are presented. This is followed by describing the constructive heuristic algorithm is applied for machine assignment in the metaheuristic algorithms.

3.1. The representation of chromosome/solution

3.1.1. GA and SA chromosome/solution representation

Suppose that there are n jobs each one to be assigned to two identical machines in each stage. The proposed algorithms first choose the initial solutions using a uniform distribution in a range between 0 and one. For the sake of clarification, suppose that there are 5 jobs and two machines in each stage. At first 5 random numbers are generated in a range between 0 and 1. Assume that the outcomes are 0.45, 0.63, 0.13, 0.33 and 0.77. Figure 2 shows the corresponding chromosome/solution.

Figure.2 Initialized chromosome for GA and SA

The job sequences are defined according to the ascending values of the chromosome/solutions shown in Figure 3. Figure 6 shows the resulting sequence.

Figure 3. The Job sequence corresponding the GA/SA algorithm

Having generated the job sequences, at each stage the jobs are assigned to the machines using the constructive heuristic algorithm accordingly. It is described at below.

3.2. Constructive heuristic

The jobs are assigned to the machine with the earliest available time. The details are explained at below.

$Step\ 0$: $Set\ \ t_{1i} = 0\ (i = 1,2,\ldots,m_1)$ and $t_{2l} = 0\ (l = 1,2,\ldots,m_2)$

$\min(t_{2l}) = T_2, \min(t_{1i}) = T_1,\ T = T_2 - T_1$

$Set\ \pi = \{\pi_1, \pi_2, \ldots, \pi_n\}$ as job sequence and Q to set of scheduled jobs

Step 1 : select the machine with earliest available time for both stages and compute t_{2j} accordingly. the job sequence.

$Set\ \pi \rightarrow \pi - \{j\}$ and add j to the Q

Step 2 : if $P_{1J} - T \geq 0$ then $t_{1j} = t_{1j} + p_{1j}$ and $t_{2j} = t_{1j} + p_{2j}$

if $P_{1J} - T < 0$ then $t_{1j} = t_{2j}$ and $t_{2j} = t_{2j} + p_{2j}$

Step 3 : if $\pi = \varnothing$, stop and compute objective function. otherwise , go to step 1

Where t_{1i} and t_{2l} are current processing times in first stage and second stage respectively, P_{1j} and P_{2j} represent processing time of the job j in first stage and second stage respectively. And π denotes the job permutation.

3.3. Genetic Algorithm

Holland [23] for first time introduced genetic algorithm as an optimization tool that redefines the problems in the genotype space by converting the features of phenotype as the chromosomes. In the genetic algorithm a population of chromosomes is maintained and stochastic search operators such as crossover, mutation and selection are used to finding better answers [24]. The combinations of genes are evolved through the genetic operators so that the chromosomes move toward the optimal solution generation by generation. There are three genetic operators: selection, crossover and mutation. Crossover operators can produce new solutions optimistically retaining good characteristics from parents, and mutation operator can enhance diversity and provide a chance to escape from local optima. So far, GAs have been widely applied in many fields, especially in operations research [25, 26].

3.3.1. Selection operator

The main goal of selection operators is choosing better chromosomes to obtain better solutions. To achieve this goal, several issues should be taken into consideration; how to choose chromosomes in the population in order to create offspring and how many offspring should be created and how chromosomes are selected as population in the next generation. [27]

Parent selection is performed by roulette wheel selection scheme. In this procedure selection probability of each chromosome is calculated by following formula:

$$fitness\ value = \frac{1}{C_{\max}} \qquad (1\text{-}3)$$

In this scheme each chromosome has more chance to be selected as the fitness value increases. This rule results in finding strong off springs.

After determination of chromosome as parents, in order to transit to the next generation ($\lambda + \mu$)-selection strategy is applied as survivor selection rule [28]. In the process of survival selection, in each generation, after producing λ offspring and calculating the respective fitness values, among $\lambda + \mu$ chromosomes, the best $\lambda + \mu$ chromosomes are selected as the next population members.

3.3.2. Crossover

The crossover operation is used to effectively explore the search space. The main purpose of crossover operator is to exchange information among randomly selected parent chromosomes with the aim of producing better solution, i.e. offspring. It recombines genetic members of two parent chromosomes to produce the offspring for the next generation that retain good properties of the parent chromosomes. The exchange is also intended to search for better genes [29]. To perform a crossover, two parents are selected from the mating pool at random. With the probability of $1 - p_c$, the parents are copied as they are. On the other hand, with the probability of pc, called the crossover probability, the crossover operation is performed. Figure 4 presents the crossover operation.

Binary mask	1	0	1	0	0	1	0	1
Parent 1	0.51	0.34	0.09	0.76	0.42	0.23	0.81	0.15
Parent 2	0.43	0.86	0.52	0.29	0.16	0.91	0.73	0.31
Offspring 1	0.51	0.86	0.09	0.29	0.16	0.23	0.73	0.15
Offspring 2	0.43	0.34	0.52	0.76	0.42	0.91	0.81	0.31

Figure 4. Crossover procedure

According to the above addressed figure, firstly an array of binary mask with a length equal to the chromosome members is generated. Then the offspring is generated from a couple of nominated parents in two ways resulting in offspring 1 and 2. In offspring one, the ith member is copied from the ith member of parent 1 if the respective binary mask member value is equal to one, otherwise the value is copied from parent 2. This is continued till all members of the offspring are constructed. The second offspring member generation is similar to that of the first offspring except that, in any selection, the member from the first parent is selected if the respective binary mask value is equal to zero, otherwise the member from the second value is generated. Syswerda, G [30] shown that this crossover operator obtains good exploitations. Therefore in this paper it is applied as a cross over operator in the proposed GA algorithm.

3.3.3. Mutation

Mutation serves to maintain diversity in the population and is a way of enlarging the exploration. It acts to prevent the selection and crossover from focusing on a narrow area of the search space or the GA getting stuck in a local optimum. Mutation is done by selecting two different locations on the chromosome at random and interchanging the jobs at these locations. For each child obtained from crossover, the mutation operator is applied independently with a small probability pm. A sample of swap mutation is shown in the Figure 5.

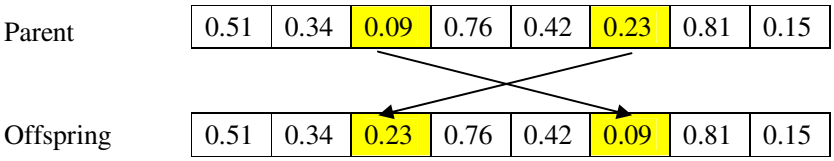

Figure 5. Mutation procedure

3.3.4. Termination criterion

The algorithm continues until all generations are created. The running time of the GA depends on the number of generations along with the population size. The GA will have a greater like hood that an optimal or near to optimal solution is found if it is to run longer. Structure of genetic algorithm which is used in this study are shown in Figure 6.

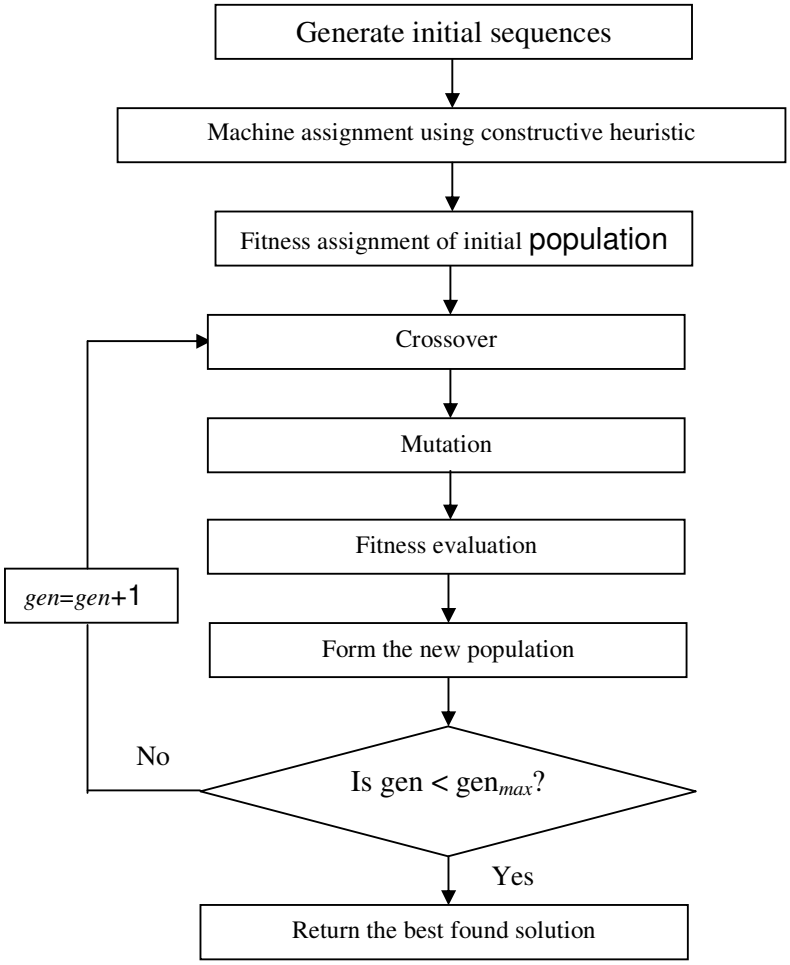

Figure 6. Proposed genetic algorithm structure

3.4. Simulated annealing algorithm

The Simulated Annealing (SA) is a search technique, which can be used to seek good solutions for various combinatorial problems. It has its origin in the material science and physics. The

motivation for simulated annealing comes from an analogy between the physical annealing of solid materials and optimization problem.

The analogy between physical annealing and simulated annealing can be summarized as follows: (i) The physical configurations or states of the molecules correspond to optimization solution; (ii) the energy of molecules corresponds to the objective function or cost function; (iii) a low energy sate corresponds to an optimal solution; (iv) the cooling rate corresponds to the control parameter which will affect the acceptance probability. The algorithm consists of four main components: (i) configurations; (ii) re-configuration technique; (iii) cost function and (iv) cooling schedule [31, 32]. Similar to genetic algorithm, simulated annealing is a stochastic search method. It aims to find an acceptable solution where it is impractical to find the optimum solution using other methods.

Simulated Annealing is a generalization of a Monte Carlo method for examining the equations of state and frozen states of a system [32]. Since the original work conducted by Metropolis on simulated annealing [32], a significant amount of works have been done on this technique and its applications in different fields. Because of the combinatorial nature of the scheduling problems, application of simulated annealing on this problem has been investigated extensively. Examples of using SA can be found in single machine, Flow shop, Job shop problems and combinations of these problems with various restrictions and job conditions [33-35]. However, to the author's knowledge, so far no work was conducted to solve no wait two stage flexible flow shop scheduling problem using simulated annealing approach.

The detailed description of the simulated annealing algorithm to solve no wait two stage flexible flow shop problems with minimum makespans is presented below.

- Step 1: Set the parameters: In this step the initial parameters of the algorithm are set. These include initial temperature, T (1), the minimum temperature, T (min), the temperature reduction multiplier, α, and the number of iterations at each Temperature, N. Also the iteration counter, n, are set to one in order to start the first iteration.
- Step 2: Set F(X) equal to the objective function for the initial solution: In this step the value of the objective function for the initial sequence (X) obtained in the first phase is set equal to F(X). X is also defined as the best solution found (Xbest) and the best objective function calculated, F(Xbest), is set equal to F(X).
- Step 3: Generate Y, a neighbourhood solution: In this step two jobs are randomly chosen and swapped in the sequence of jobs that define their priorities. Following the procedure used in phase 1, find the corresponding solution. Denote this solution as Y and the value of its objective function as F(Y).
- steps 4: Check for improvement: The improvement, δ , of the new solution compared to the previous solution is evaluated as the difference between the objective functions of solutions Y and X. If δ is less than zero, then there is an improvement and the algorithm continues with step 5 otherwise go to step 6.
- Steps 5: Compare the new solution with the best solution found. If the new solution is better, i.e. F(Y) <F (Xbest), replace the best solution and its objective function with Y and F(Y) respectively. Continue by following step 7.
- Steps 6: Accept or reject the non-improving move randomly In order to randomly accept a non-improving move that might lead to a better solution, calculate L=Exp ($-\delta/T(t)$) and compare it to R, a random number between zero and one. If L>R the non-improving move will be accepted with a hope to find a better neighbor of the solution, and the algorithm continues with step 7. Otherwise, solution Y is ignored and the algorithm returns to step 3 to generate a new solution.
- Step 7: Update solution X Since solution Y was accepted in steps 5 or 6, solution X and F(X) are replaced with Y and F(Y) respectively.
- Step 8: Update and check the iteration counter and compare the counter n with N. If n<N, increase n by one and go to step 3 to start a new iteration, otherwise increase the

temperature counter t by one, calculate the new temperature: $T(t) = \alpha \times T(t-1)$, set n=1, and continue with step 9.

- Step 9: Check the termination criteria; If the temperature of the system is less than or equal to the minimum temperature allowed, the algorithm is terminated. Otherwise continue with the new temperature iteration by going to step 3. Structure of proposed simulated annealing is shown in Figure 7.

3.5. The fitness function

The performance measure considered in this study is minimum makespan. The corresponding fitness function is calculated as follow:

Cj= Completion time of job j

Makespan= Cmax = max (Cj)

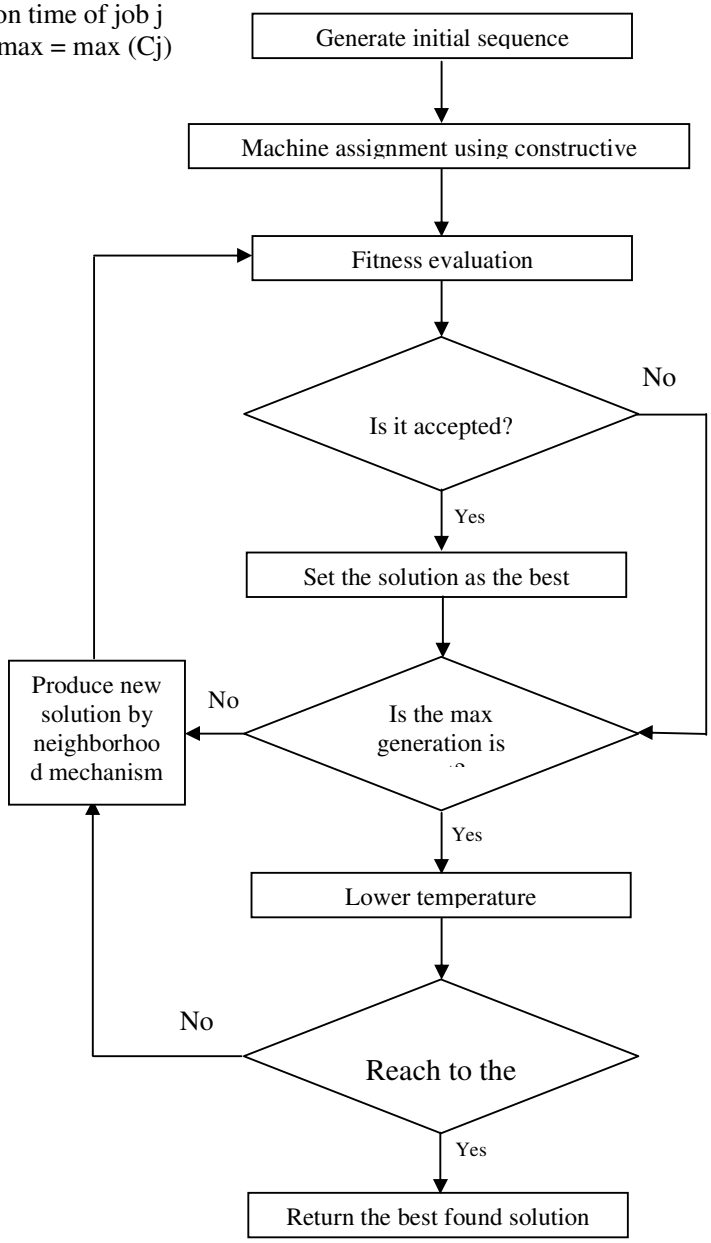

Figure 7. Proposed simulated annealing structure

4. NUMERICAL EXPERIMENTS

In this section, the performances of the proposed algorithms are studied by solving series of problems. At first the parameters used in the simulation experiments are introduced. This includes the scale of the problems in terms of the number of the jobs, number of machines at each stage and processing times. Next the results of the simulation study are illustrated and the performance of the proposed algorithms is compared with that of the others by solving various problems in different sizes.

4.1. Parameters of the simulation model

In this study we investigated the performance of the proposed approach for a fairly large number of the problems. In terms of the number of machines, the problems studied in this research are classified into two main categories namely small and large problem. Table.1 shows the problem size for each category.

Table.1 Problem size

Problem	Number of machines
Small	$(m_1 = 3, m_2 = 4), (m_1 = 2, m_2 = 2), (m_1 = 3, m_2 = 2)$
Large	$(m_1 = 8, m_2 = 10), (m_1 = 10, m_2 = 10), (m_1 = 12, m_2 = 10)$

For each problem illustrated in Table 2 (i.e. $m_1 = 3, m_2 = 4$), three different size of orders (i.e. number of jobs) are considered. They are twice, 4 times and 6 times of total number of machine respectively. In addition the operation times are generated using a uniform distribution with a couple of ranges. Table.2 shows the number of jobs and operations distribution.

Table.2 Number of jobs and distribution of processing times

N(number of jobs)	Small&Large	$(2\times(m_1+m_2)), (4\times(m_1+m_2)), (6\times(m_1+m_2))$
DT(distribution of processing times)	Small	$U(4,40)$
	Large	$U(60,300)$

4.2. Parameter tuning

Regarding the metaheuristic algorithms, the value of the parameters used in each algorithm affect its performance. For this purpose, the simulation was performed for a wide range of the parameters' values for both GA and SA algorithms and the values corresponding to the best solutions are selected. Table.3 shows the summary of the tuned parameters' value for the proposed algorithms.

Table. 3 Parameters setting for the algorithms

	Parameters	Small size	Large size
GA	Population size	100	200
	Generation number	200	500
	Crossover rate	0.7	0.7
	Swapping mutation rate	0.4	0.4
SA	T_0	200	400
	T_f	0.001	0.0001
	Iteration number	200	500
	α	0.9	0.95

4.3. The simulation process

This section describes the results of the computational experiments performed to evaluate the performance of the proposed algorithms compared with that of MDA which is considered as an efficient algorithm for a no wait two stage flexible flow shop. The model was developed using MATLAB 2008a in a personal computer with 3.4 GHz, 895 MB memory. The performance of the algorithms was testified by solving 36 different problems (18 problems in small scale and 18 problems in large scale). Regarding the performance measure, Relative Percentage Deviation (RPD) of Makespan over the best solution is used. It is calculated as follows:

$$RPD = \frac{\left|Method_{sol} - Best_{sol}\right|}{Best_{sol}} \times 100 \qquad (3)$$

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

4.4. Simulation results

As described in section one, review of the literature reveals that MDA obtains good solutions for the problems studied in this paper. Therefore in our study the performance of the proposed algorithms were compared with each other and MDA algorithm for 36 different problems. Simulation results are shown in two scales separately. (See Table.4 and Table.5)

Results shown in Table 4 indicate that for small problems, in 17 out of 18 cases, the Simulated Annealing (SA) outperforms the other the algorithms. In addition RPD performance measure with % 95 confidence interval was calculated in both small and large scale problems. Table.6 shows the results. This indicates that in terms of RPD and with %95 confidence interval the range of the results obtained using the algorithms are not in overlapped. In addition the performance of the SA algorithm has lower Confidence interval. This confirms the superiority of the proposed SA compared with the other algorithms.

Table 5 presents the results for large scale problems. Similar to the small scale problems, here SA also outperforms the all other algorithms; also similar to small scale problems in terms of RPD and with %95 confidence interval the range of the results obtained using the algorithms are not in overlapped. So this confirms the superiority of the proposed SA compared with the other algorithms

Table 4 and 5 also present the performance of each algorithm in terms of CPU time. The results indicate that MDA has normally yielded the shortest CPU time. Such a processing speed is obviously due to the fact that MDA is a one iteration simple algorithm compared with the other algorithms in which the solutions are obtained though multi iteration based rules. Despite the fact that the proposed SA has significantly longer CPU time than that of MDA, but comparable with that of the other algorithms, its CPU time still is short. For instance for the largest problem with 132 jobs and 12 and 10 machines in stage one and two respectively, its CPU time is 230 second which is fairly short. Furthermore, for statistical evaluating the algorithms, mean and interval (at 95% confidence interval) are plotted for small and large scale problems in Figure 8 and 9,respectively. with respect to these Figures it could be seen poroposed simulated annealing outperform GA and MDA in both scales. So, As a result, the proposed SA can be considered as an effective algorithm for no wait two stage flexible flow shop scheduling problem with minimum makespan.

Table.4 simulation results for small scale

			C_{max}			Cpu time(s)		
no. jobs	M1	M2	SA	GA	MDA	SA	GA	MDA
	3	4	78	78	100	3.031	3.469	0.049
8	2	2	116	116	121	3.031	3.453	0.046
	3	2	115	115	126	2.031	3.453	0.048
	3	4	104	104	128	2.234	1.938	0.054
10	2	2	150	153	165	2.344	1.938	0.051
	3	2	144	143	154	2.266	1.938	0.053
	3	4	108	109	145	2.859	1.953	0.061
14	2	2	157	160	188	2.813	1.984	0.057
	3	2	133	133	137	2.813	1.969	0.058
	3	4	117	118	171	2.828	2.578	0.063
16	2	2	175	179	218	2.938	1.984	0.063
	3	2	156	156	170	2.906	2.578	0.063
	3	4	174	177	214	3.719	3.875	0.066
20	2	2	261	269	300	3.703	3.283	0.064
	3	2	234	236	247	3.781	4.359	0.066
	3	4	194	196	216	4.250	5.031	0.072
24	2	2	289	304	305	4.344	5.047	0.069
	3	2	254	256	273	4.438	5.031	0.071

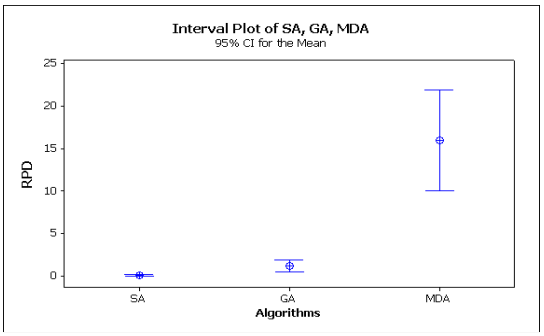

Figure 8. Means and interval plot for small scale problems in terms of RPD

Table 5. Simulation results for large scale problems

			C_{max}			Cpu time(s)		
no. jobs	M1	M2	SA	GA	MDA	SA	GA	MDA
	8	10	1822	1856	2106	41.016	38.438	0.091
72	10	10	1500	1602	1782	34.734	38.641	0.103
	12	10	1395	1468	1571	35.719	41.313	0.164
	8	10	1959	2014	2261	38.000	42.406	0.168
80	10	10	1634	1724	1901	37.641	43.516	0.254
	12	10	1546	1609	1719	37.766	43.937	0.350
	8	10	2214	2268	2543	81.344	71.672	0.447
88	10	10	1840	1950	2153	82.922	73.750	0.485
	12	10	1691	1799	1854	82.938	72.797	0.511
	8	10	2454	2550	2655	141.172	100.438	0.577
108	10	10	2074	2231	2303	149.922	123.453	0.635
	12	10	1966	2075	2128	147.672	293.156	0.646
	8	10	2784	2949	3005	330.792	106.938	0.666
120	10	10	2450	2609	2582	169.109	106.750	0.746
	12	10	2321	2439	2470	166.828	108.438	0.772
	8	10	3153	3282	3496	229.859	196.625	0.867
132	10	10	2656	2850	3031	225.563	197.219	0.945
	12	10	2493	2611	2715	230.547	210.844	1.020

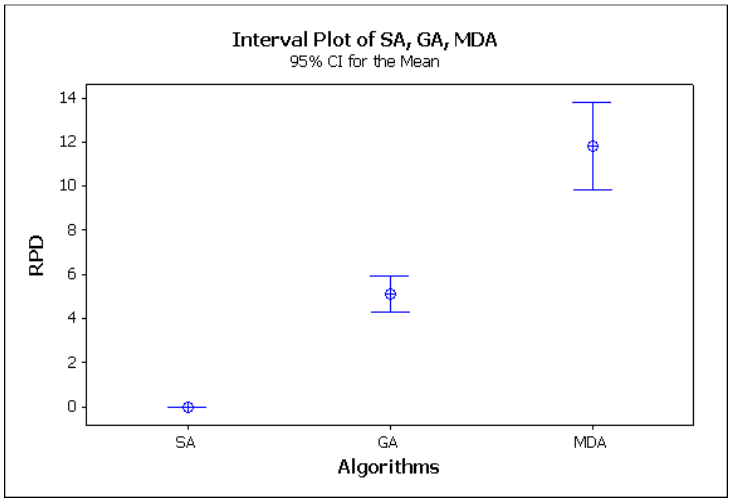

Figure 9. Means and interval plot for large scale problems in terms of RPD

4. CONCLUSION

This paper studied the performance of the proposed four metaheuristic algorithms for a no wait flexible flow shop scheduling problem with minimum makespan performance measure. The performances of these algorithms were also compared with that of MDA. A simulation model was developed to study the performance of the algorithms. For this purpose, 36 problems in

different sizes in terms of the number of job and number of machine in each stage were solved. The results of the simulation study revealed that the proposed Simulated Annealing Algorithms (SA) outperforms the other algorithms in terms of minimum makespan. Such superiority becomes more significant as the problem size increases. Therefore the proposed SA can be considered as an efficient algorithm for a no wait two stage flexible flow shop. As a further research it is recommended to study the performance of the proposed algorithm for the problem with more than two stage. In addition it is worthwhile to investigate the performance of the proposed algorithms for the problems with sequence depended set up times.

References:

[1] Zandieh, M. Gholami, M. (2009), An immune algorithm for scheduling hybrid flow shop with sequence-dependent setup times and machines with random breakdowns, International Journal of Production Research 47(24), 6999-7027.

[2] Behnamian, J. Fatemi Ghomi, S.M.T. Zandieh, M. (2010), Development of a hybrid metaheuristic to minimizing earliness and tardiness in hybrid flowshop with sequence-dependent setup times, International Journal of Production Research, 48(5), 1415-1438.

[3] Lin , L. Jiazhen, H. and Ou, T., (2010) 'A hybrid flowshop scheduling problem for a cold treating process in seamless steel tube production', International Journal of Production Research,, First published on: 22 September 2010 (iFirst)

[4] Henry W.T. and Hunsucker, J.L., (2004) A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage, European Journal of Operational Research, vol. 152, issue 1, pages 96-114 .

[5] Ruiz, R. and Jose Vazquez-Rodriguez, A., (2010) , The hybrid flow shop scheduling problem, European Journal of Operational Research 205 , 1–18.

[6] Ribas , I., Leisten, R., and.Framin˜an, J.M, (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.

[7] Riane F, Artiba A, Elmaghraby SE. (2002) Sequencing a hybrid two-stage flowshop with dedicated machines. International Journal of Production Research;40:4353–80.

[8] Narasimhan,S.and Mangiameli,P.(1987).A comparison of sequencing rules for a two-stage hybrid flow shop. Decision Science, 18, 250-265.

[9] G. -C. and Kim, Y. -D.(2004) 'A branch-and-bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing total tardiness', International Journal of Production Research, 42: 22, 4731 – 4743.

[10] Mohamed Haouari, Rym M'Hallah, (1997) Heuristic algorithms for the two-stage hybrid flowshop problem. Operation Research Letters 21(1): 43-53.

[11] Rajendran C. A no-wait flow shop scheduling heuristic to minimize makespan. Journal of the Operational Research Society 1994; 45:472-8.

[12] Hall NG, Sriskandarayah C. A survey of machine scheduling problems with blocking and no-wait in process. Operations Research 1996; 44:510-25.

[13] Grabowski J., Pempera J. Sequencing of jobs in some production system. European Journal of Operational Research 2000; 125:535-50.

[14] Raaymakers W, Hoogeveen J. Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing, European Journal of Operational Research 2000; 126:131-51.

[15] Aldowaisan T, Allahverdi A (2004) A New heuristics for m-machine no-wait flowshop to minimize total completion time.Omega 32:345–352

[16] Ke Pan et al,(2008) An improved iterated greedy algorithm for the no-wait flow shop scheduling problem with makespan criterion, The International Journal of Advanced Manufacturing Technology ,38, 7-8, 778-786.

[17] Z. Liu, J. Xie, J. Li and J. Dong, A heuristic for two-stage no-wait hybrid flowshop scheduling with a single machine in either stage, Tsinghua Science and Technology 8, 43-48, (2003

[18] Jinxing Xie , Wenxuan Xing, Zhixin Liu and Jiefang Dong (2004) Minimum Deviation Algorithm for Two-Stage No-Wait Flowshops with Parallel Machines. Computer and Mathematics with Application 47: 1857-1863.

[19] Jinxing Xie and Xijun wang (2005) Complexity and Algorithms for Two-Stage Flexible Flow shop Scheduling with Availability Constraints. Computer and Mathematics with Application 50: 1629-1638.
[20] Huang RH et al (2009) no-wait two-stage multiprocessor flow shop scheduling with unit setup. International Journal of Advanced Manufacturing Technology 44:921–927.
[21] Rabiee M, Shafaei R, Mirzaeian M, 2010. Generating an Efficient Schedule in a No-Wait Two Stage Flexible Flow Shops with Maximizing Utilization, The international conference on industrial engineering and business management (ICIEBM), Indonesia, 555-561.
[22] C.Sriscandarajah and P. Ladet (1986) some no wait shops scheduling problems: Complexity aspects. European journal of Operational research 24: 424-445.
[23] Holland, J. H. (1975). Adaptation in natural and artificial systems. Ann Arbor: University of Michigan Press.
[24] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley; 1989.
[25] Defersha, Fantahun M. and Chen, Mingyuan(2010) 'A hybrid genetic algorithm for flowshop lotstreaming with setups and variable sublots', International Journal of Production Research, 48: 6, 1705 - 1726.
[26] FranÇa, P. M. , Tin Jr, G. and Buriol, L. S.(2006) 'Genetic algorithms for the no-wait flowshop sequencing problem with time restrictions', International Journal of Production Research, 44: 5, 939 – 957.
[27] Eiben, A. E., Smith, J. E., "Introduction to evolutionary computing", springer 1st edition, 2003, ISBN: 3-540-40184-9
[28] Bäck, T., Hoffmeister, F. and Schwefel, H., A survey of evolution strategy. In Proceedings of the 4th International Conference on Genetic Algorithms, pp. 2–9, 1991 (Morgan Kaufmann: San Mateo, CA).
[29] S.K. Iyer, B. Saxena, Improved genetic algorithm for the permutation flowshop scheduling problem / Computers & Operations Research 31 (2004) 593–606.
[30] Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Proceedings of the Third International Conference on Genetic Algorithms, pages 2-8. Morgan Kauffman.
[31] P. Tian, Z. Yang, An improved simulated annealing algorithm with genetic characteristics and travelling salesman problem, J. Inform. Optim. Sci. 14 (3) (1993) 241–255.
[32] Metropolis, N., A. Rosenbluth, M. Rosenbluth, A. Teller (1953), "Equation of State Calculations by Fast Computing Machines", J. Chem. Phys., 21, 6, pp. 1087-1092
[33] Kouvelis P and Chiang W (1992) A simulated annealing procedure for single row layout problems in flexible manufacturing systems. International Journal Production Research. 30, 717-732.
[34] Chinyao Low,(2005). Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Computers & Operations Research 32, 2013–2025.
[35] D. Spinellis; C. Papadopoulos; J. MaCgregor Smith(2000) Large production line optimization using simulated annealing , . International Journal Production Research. 30, 717-732.

Authors:

Meysam Rabiee completed his B.Sc. in Industrial Eng. at Bu-Ali Sina University (2005_2009), and M.Sc. in Industrial Eng. at K.N. Toosi University Technology (2009_2011). His research interests are scheduling, intelligent systems, metaheuristic and heuristic algorithms, prediction and vehicle routing problems.

Pezhman Ramezani completed his B.Sc. in Industrial Eng. At Kerman, Iran (2003-2007) and M.Sc. in Industrial Eng. at K.N. Toosi University Technology, Tehran,Iran (2009_2011). His research interests are production scheduling, continuous optimization, financial engineering, multi-criteria decision making and supply chain management.

Rasoul Shafaei completed his B.Sc. in Isfahan university of technology (in iran) and his M.Sc in Tarbiat moddaress university (in iran) and his Phd in UMIST (in united kingdom, Manchester). Currently, he is an Asistant Professor at Industrial Eng. Department, K.N.Toosi University of technology, Iran. His research interests are supply chain management, ERP, scheduling and Strategy Management.