A Memetic Algorithm for Hybrid Flowshops with Flexible Machine Availability Constraints

Fariborz Jolai[a], Mostafa Zandieh[b,*], Bahman Naderi[c]

[a]*Department of industrial engineering, faculty of engineering, University of Tehran, Tehran, Iran*

[b]*Department of industrial management, management and accounting faculty, Shahid Beheshti University, Tehran, Iran*

[c] *Department of indusrial engineering, Amirkabir University of Technology, Tehran, Iran*

Received 5 Oct., 2008; Revised 2 Dec., 2008; Accepted 9 Apr., 2009

Abstract

This paper considers the problem of scheduling hybrid flowshops with machine availability constraints (MAC) to minimize makespan. The paper deals with a specific case of MAC caused by preventive maintenance (PM) operations. Contrary to previous papers considering fixed or/and conservative policies, we explore a case in which PM activities might be postponed or expedited while necessary. Regarding this flexibility in PM activities, we expect to obtain more efficient schedule. A simple technique is employed to schedule production jobs along with the flexible MACs caused by PM. To solve the problem, we present a high performing metaheuristic based on memetic algorithm incorporating some advanced features. To evaluate the proposed algorithm, the paper compares the proposed algorithm with several well-known algorithms taken from the literature. Finally, we conclude that the proposed algorithm outperforms other algorithms.

Keywords: Scheduling; Hybrid flowshops; sequence dependent setup times; machine availibility constraints; memetic algorithm.

1. Introduction

One of the well studied scheduling problems is flowshop (FS). In FS, a given set of *n* jobs need to be processed in a set of *m* stages each of which has one machine. The processing routes of all the jobs are the same. In sight of the fact that researchers intend to bridge the existing gap between theory and practice of the scheduling, many assumptions to more actualize the problem of flowshops are recently made. In practice, there might be more than one single machine in each stage. By having machines in parallel, we are capable of eliminating or reducing the impact of bottleneck stages on the overall shop floor capacities. Although the machines in each stage are identical, each job is processed by only one machine in each stage. With respect to the corresponding explanation, we address hybrid flowshops (HFS) to deal with a complex while realistic case of flowshops under minimization of makespan. Moreover, In HFS it is assumed that the jobs are independent. Each machine can only process a job at a time while each job can be processed by at most one machine at a time. The jobs are Non-preemptive i.e. the process of a job on a machine cannot be interrupted. There exist unlimited place for work-in-process jobs between stages. Additionally, majority of papers in the literature consider an unrealistic assumption of the continuous machine availability. However, a machine can be unavailable for many reasons, such as unforeseen breakdowns (stochastic unavailability) or due to a scheduled preventive maintenance (PM) (deterministic unavailability). It is well known that PM has a vital role in many industries [11], such as semiconductor and plastic industry; therefore, it should be carefully considered. According to practical experience, a poor scheduling of maintenance may greatly reduce the shop productivity. As a result, the presentation of techniques to integrate production and PM activities is a key issue in the field of scheduling. Almost all the paper in the literature consider fixed or/and conservative policies (i.e. the PM operation must be scheduled at exactly predetermined intervals). We here with apply a flexible criterion to consider PM operations along with productions jobs to gain more effective schedule. Since HFS belongs to a special class of combinatorial

[*] Corresponding author. Telfax.: +98-21-29902382; e-mail: m_zandieh@sbu.ac.ir

optimization problems known to be a nondeterministic polynomial-time hard one (NP-hard), there is no exact method solving the problem in reasonable amount of time. Hence, several heuristics and metaheuristics have been presented to tackle the problems. Kurz and Askin [2] studied hybrid flowshops with setup times separated and proposed some heuristics to solve the problem. Later, Kurz and Askin [3] considered the same problem and adapted a well-known genetic algorithm, called random key genetic algorithm (RKGA). They showed that RKGA outperformed their heuristics proposed aforetime. Zandieh et al. [13] presented an immune algorithm that worked better than RKGA. Recently Naderi et al. [7] addressed hybrid flexible flowshops and introduced a novel metaheuristic based on the concept of variable neighborhood search.

In a nutshell, the contribution of this paper is to introduce a flexible criterion to integrate production and PM operations, and to propose a high performing metaheuristic, namely memetic algorithm, to solve hybrid flowshops with flexible machine availability constraints to minimize makespan. The reason to memetic's ever-increasing popularity among researchers is its powerful diversification capability as well as its intensification capability [12]. Besides the high diversification capability, the proposed memetic algorithm employs a very simple and fast form of simulated annealing to possess a good intensification operator as well. Its potential on solving the problem studied here is investigated against the adaptations of the some well-known algorithms in the literature through a set of instances.

The rest of the paper is organized as follows: Section 2 introduce the flexible machine availability constraints. Section 3 presents the proposed memetic algorithm. In Section 4, the computational experiment is explained. Section 5 gives some conclusions.

2. Flexible machine availability constraint

In practice, the consideration of continuous machine availability might not be true. For example, the machines might be busy processing jobs left in the previous horizon or breakdown. Many researchers consider PM activities as a most systematic reason for the MACs. Many papers have studied to schedule the production jobs along with the PM operations. The integrating criteria so far introduced are usually regarded as fixed or/and conservative policies [6, 11]. In the fixed policies, PM activities are performed at exactly pre-specified time intervals while in the conservative policies, whenever production and PM activities have overlap, the production operation is postponed and PM activities are conducted first. In this paper, we introduce a more flexible criterion to integrate production scheduling and PM activities. In other word, we assume that the starting time points of PM operations could be flexible to some extent (δ). In this case, likely more efficient schedules could be obtained. In a nutshell, our procedure of integration is as follows: Let us suppose that the time interval between two consecutive PM operations is T_{PM}. Whenever a new job is to be processed in each machine, the completion time is computed. If this time exceeds the $T_{PM} + \delta$, then the process of the next job is postponed and the PM is carried out first. It is necessary to state that since we consider the non-preemptive case, the process of a job cannot be interrupted before it completes. To better clarify the above procedure, we apply it to an example. Let us consider a shop with $T_{PM} = 15$ time units. The duration of PM operations (D_{PM}) are 3 time units. The maximum accepted delay (δ) is 5 time units. The shop has 4 jobs to process on a single machine. Table 1 shows the processing times of the jobs.

Table 1
The processing times for a problem with $n = 5$

Job *i*	Processing time
1	5
2	10
3	6
4	7

Again let us suppose the jobs are scheduled as such: {4, 3, 1, 2}. After processing jobs 4 and 3, the completion time becomes 7 + 6 = 13. To process job 1, the completion becomes 18 which is greater than $T_{PM} = 15$; However, the shop can accept a delay with the maximum of 5 time units. So, job 1 can be processed. Now, it is impossible to carry out job 2 because it has a processing time of 10 time units resulting in a completion times of 28 units, which is greater than $T_{PM} + \delta = 20$. Therefore, the process of job 2 is postponed and PM operation is carried out first. The first PM operation and job 1 complete at 21 and 31, respectively. Figure 1 shows the Gantt chart of the solution.

3. The proposed memetic algorithm

Memetic algorithm (MA) is a recent metaheuristic to solve combinatorial optimization. MA can be regarded as a combination of a population-based global search and local improvements. Some recent researches [9, 12] conclude that the performances of evolutionary metaheuristics like genetic algorithm (GA) can be significantly improved by hybridizing with a powerful

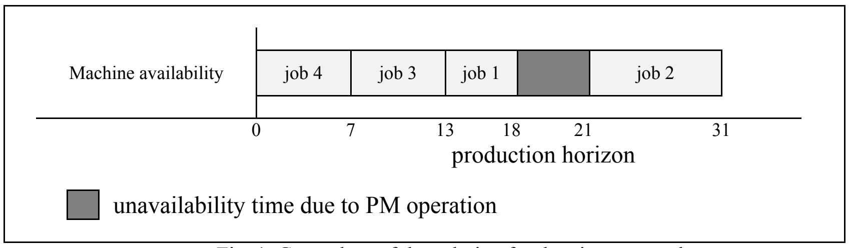

Fig. 1. Gantt chart of the solution for the given example

and fast local search-based engine. The original intention is to obtain an intelligent integration of global search and local search, and to make a well-balanced compromise between diversification and intensification mechanisms. According to [12], MAs can be enhanced through the combination of the evolutionary algorithms with local search-based strategies such as simulated annealing.

In brief, MA explores the search space through a population of encoded solutions, called *chromosomes*. According to *chromosome*'s quality, each of them is assigned a value called *fitness*. The population evolves by a set of operators so long as some stopping criterion is met. A typical iteration of MA, *generation*, can be stated as follows: The best *chromosomes* of the current *population* are directly copied to the next generation (*elite strategy*). A *selection* mechanism picks *chromosomes* of the current *population* so as to give higher chance of being selected to the *chromosome* with the better fitness value. The selected *chromosomes* are combined and produce new *offspring* through *crossover*. After the mating process, each *offspring* might mutate by another mechanism called *local search engine*. The new *offspring* constitute a new population and the procedure restarts. Figure 2 shows the general pseudo code of the proposed memetic algorithm.

```
Procedure memetic algorithm

Initialization
while the stopping criterion is not met do
    fitness evaluation
    elite operator
    crossover operator
    local search engine
endwhile
```

Fig. 2. The general pseudo code of the proposed memetic algorithm

In the following subsection, we describe the main features of the proposed memetic algorithm: chromosome representation, initialization, fitness evaluation, selection mechanism, elite strategy, crossover and local search engine.

3.1 *Chromosomes representation, initialization, fitness evaluation and selection mechanism*

In hybrid flowshops, permutation representation is the frequently used scheme to encode a solution [7]. Permutation representation lists all the jobs in a relative order by which they are scheduled in stage 1, and then by a machine assignment rule, the jobs are allocated to the machines. The job sequence in subsequent stages is determined by the earliest completion times of the jobs in the previous stage. The machine assignment rule (MAR) is to allocate the jobs to the machines in each stage. In flowshops since every stage has only one machine, we do not need any MAR, whereas in HFS, we have to employ an effective MAR. In the case of HFS, each job is assigned to the machine completing the job in the earliest time in the given stage.

It is known that the initialization procedure has a great impact on the quality of a metaheuristic. Therefore, we utilize the best so far known heuristic, NEH [10], in the literature as an initial solution. Since in MAs higher fitness values are more preferable and our objective is the minimization of C_{max}, we use 1/ C_{max} as the fitness value of a solution. For the selection of parents, we make use of binary tournament selection [1]. In binary tournament selection, two chromosomes of the current population are randomly selected, and the better one is chosen as a parent.

3.2 *Elite strategy and crossover*

To ensure that when the algorithm proceeds the best so far chromosomes are not eliminated, chromosomes with higher fitness values are directly copied to the next generation. The selected chromosomes are combined to generate new offspring through an operator called crossover. The purpose is to produce better schedule after crossing the parents. Since we use permutation representation, the crossover operators must work so as to avoid generating infeasible solutions. Our crossover is "Similar Job Order Crossover" or SJOX. This crossover has been proven to be very high performing in flow shops [10] against several other crossover operators. Therefore, we have been thinking of applying it to HFS.

3.3 Local search engine

The key feature of memetic algorithms is the local search engine. This is so because we can make a balance between the diversification capability of population-based algorithms and intensification capability of the local search-based algorithms. The local search starts from a given solution and performs a quick search around that solution. If any improvement is made, the current solution is replaced with the better one. We utilize simulated annealing (SA) as our local search engine because SA is known to be a fast and simple local search. The local search engine works as such: After crossing, the SA is applied to a fraction of the chromosomes that their C_{max} are at most r% over the best chromosome, not all the chromosomes because applying the SA to all the chromosomes would result in a very slow algorithm. In the following subsection, we shortly describe the simulated annealing we apply.

3.4 Simulated annealing

The basic procedure in simulated annealing (SA) is to produce a new job sequence k by a random operator from the neighbourhood of present sequence u. This new sequence is accepted or rejected by another random technique. A parameter t, called the temperature, controls the acceptance rule. The variation between objective values of two candidate solution is computed $\Delta C = TCT(k) - TCT(u)$. If $\Delta C \leq 0$, sequence k is accepted. Otherwise, sequence k is accepted with probability equal to $exp(-\Delta C / t_i)$. The algorithm proceeds by trying a fixed number of neighbourhood moves at temperature t_i, while temperature is gradually decreased. The procedure repeats until a stopping criterion is met. In our algorithm, SA proceeds until in five consecutive temperatures, no improvement is made.

Simulated annealing starts from an initial solution, and a series of moves are made. The algorithm checks 20 neighbours at temperature t_i. Move operator produces a new solution from current candidate solution by slightly changing it. Since it is concluded that in SAs, SHIFT operator is superior to other operators like SWAP and INVESION [5], we generate new solution using SHIFT operator in which a randomly selected job in sequence is randomly relocated. Here, we make use of exponential cooling schedule, $t_i = \alpha \,.t_{i-1}$ (where $\alpha \in (0, 1)$ is temperature decrease rate).

4. Experimental evaluation

In this section, we investigate the performance of the proposed algorithm. To conduct the experiment, we implement the algorithm in MATLAB 7.0 running on an PC with 2.0 GHz Intel Core 2 Duo and 2 GB of RAM memory. Relative percentage deviation (RPD) is used as our performance measure [6]. RPD is calculated as follows:

$$\text{RPD} = 100 \bullet (Alg_{sol} - Min_{sol}) / Min_{sol} \qquad (1)$$

where Alg_{sol} is the C_{max} obtained for a given algorithm and instance, Min_{sol} is the best solutions obtained for each instance by any of all algorithms. In the following two subsections, we first tune the parameters of the proposed memetic algorithm, and then we compare its performance against the some well-known algorithms in the literature.

4.1. Parameter tuning

In this section, we intend to set the parameters of our proposed MA by means of the full factorial experiment which is one of the DOE approaches [4]. A set of 60 random instances are generated in different size of $(n \cdot m)$. Stopping criterion is $n{\cdot}m{\cdot}0.2$ seconds computational time which allows for more time as the number of jobs or machines increases. The proposed MA has two parameters, population size (*popsize*) and r in local search engine, that need to be tuned. We investigate the following levels for *popsize*: 20, 40, 60, 80. The results demonstrate that *popsize* of 60 outperforms the other levels. Figure 3 shows the means plot and least significant difference (LSD) intervals for each level of *popsize*.

We also consider the following levels for r: 0%, 2%, 5%, 10%. Figure 4 show the results obtained by each level of the parameter r. As could be seen, the best level is r = 5%. It is interesting to see that r = 0% (i.e. MA with no local search engine) results in the worst performance.

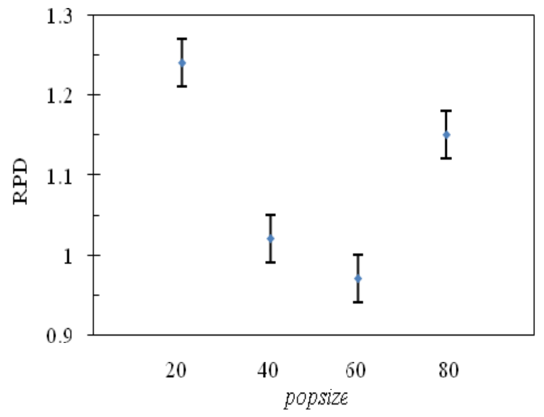

Fig. 3. The means plot and LSD intervals for different levels of *popsize*

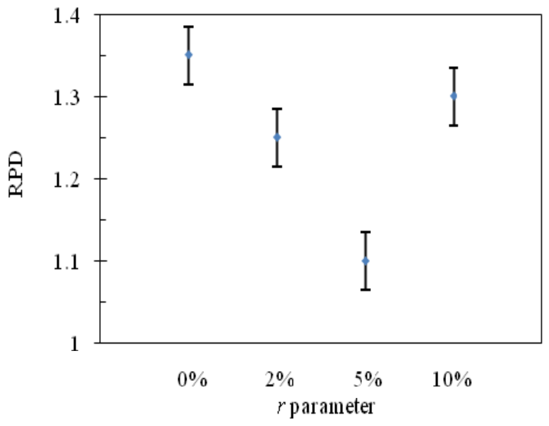

Fig. 4. The means plot and LSD intervals for different levels of *r* parameter in the local search engine

4.2. *Experimental results*

In this section, we evaluate our proposed memetic algorithm against other existing algorithms including SPTCH, FTMIH, Johnson heuristics proposed by [2], NEH of [10], RKGA of [3], immune algorithm (IA_Z) of [13] and variable neighbourhood search (VNS_N) of [8]. All the above-mentioned algorithms are adapted so as to consider the existence of flexible PM operations. The stopping criterion is $n{\cdot}m{\cdot}0.2$ second computational time. We use RPD measure (*Eq.* 1) to compare the algorithms.

To compare the performances of the algorithms, a set of instances is generated. We need to notice that data required for a problem consist of the number of jobs (n), range of processing times (p), number of stages (m), the number of machines in each stage (m_i), time interval between two consecutive PM operations (T_{PM}), duration of PM operations (D_{PM}) and also flexibility of PM operations (δ). We have n = {20, 50, 80, 120} and m = {2, 4, 8} similar to Naderi et al. (2008). To define the number of machines at each stage, we have to sets. In the first one, we have a random uniform distribution number of machines of between one and three machines per stage, and in the second one, we have a fixed number of two machines per stage. The processing times are generated from a uniform distribution over the range (1, 99). T_{PM} for each machine are distributed as a uniform distribution in the range (200, 300). D_{PM} of each machine are distributed uniformly over three ranges (1, 50). δs are randomly generated from a uniform distribution between (20, 80). The different levels of factors result in 24 different scenarios. There are 10 different instances for each scenario. Therefore, we have 240 instances.

Table 2 summarizes the results of the experiments. In this table, we report the average RPD for each combination of n and m (20 data per average). The best performing algorithm is MA with RPD of 1.03%. The second best is VNS_N with RPD of 1.71% while among the heuristics, NEH performs better. FTMIH is the worst performing algorithm with RPD of 30.27%. For further analysis of the results, we conduct an analysis of variance (ANOVA) test where the type of the algorithm is the factor and RPD is the response variable. Due to the considerable difference between the heuristics and metaheuristics, we exclude the heuristics from ANOVA experiment. There are statistically significant differences between the different types of metaheuristics with a p-value very close to zero. Figure 5 shows the means plot and LSD intervals. As could be seen in Figure 5, MA statistically outperforms all the algorithms.

5. Conclusion and future research

This paper dealt with hybrid flowshops with flexible machine availability constraints under minimization of makespan. Besides the establishment of a simple and flexible criterion to integrate the production and PM operations, we proposed a high performing metaheuristic to tackle the problem. This algorithm is memetic algorithm that employs a fast and simple simulated annealing in its local search engine. To evaluate the proposed algorithm, we compared it with some existing

Table 2
The average RPD for the algorithms grouped by *n* and *m*

n	*m*	Algorithm							
		SPTCH	FTMIH	Johnson	NEH	RKGA	IA_Z	MA	VNS_N
20	2	32.94	35.78	22.97	4.89	3.03	2.02	0.41	1.18
	4	23.08	30.86	15.16	6.28	1.10	1.82	0.61	1.28
	8	18.04	23.06	12.28	6.89	0.88	1.38	0.67	0.80
50	2	27.53	32.48	23.35	3.89	5.47	2.49	0.22	0.72
	4	19.75	32.13	20.43	5.47	2.52	1.57	1.29	2.05
	8	18.55	22.35	12.78	5.86	0.47	2.24	1.35	2.29
80	2	29.61	35.44	25.29	3.01	5.08	4.48	0.70	1.59
	4	21.25	29.68	19.38	6.28	3.71	4.27	0.78	1.62
	8	21.38	25.48	17.34	9.00	4.85	3.75	0.87	1.73
120	2	29.43	33.42	27.83	3.10	7.51	4.24	1.92	2.27
	4	27.35	33.97	24.75	7.50	8.89	5.99	1.82	2.63
	8	21.75	28.55	19.26	11.14	8.37	5.09	1.69	2.33
Average		24.22	30.27	20.07	6.11	4.32	3.28	1.03	1.71

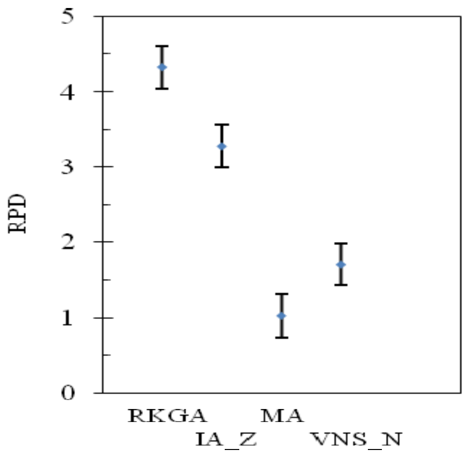

Fig. 5. Means plot and LSD intervals (at the 95% confidence level) for the type of algorithm factor

algorithms in the literature. The computational results showed the outperformance of the proposed memetic algorithm.

As future research, it is could be interesting to extend the memetic algorithm to other scheduling problems or to the studied problem in this paper with other objectives, such as total tardiness and number of tardy jobs. It is worthy working on some other realistic assumptions like transportation times or extending the work done here to other scheduling problems.

Reference

[1] D. E. Goldberg, Genetic algorithms in search, optimization and machine learning, 1st edition, Addison-Wesley, Reading, 1989.

[2] M. E. Kurz, R. G. Askin, Comparing scheduling rules for flexible flow lines. International Journal of Production Economics, 85, 371–388, 2003.

[3] M. E. Kurz, R. G. Askin, Scheduling flexible flow lines with sequence-dependent setup times. European Journal of Operational Research, 159(1), 66–82, 2004.

[4] D. C. Montgomery, Design and Analysis of Experiments. 5th edition, John Wiley & Sons, New York, 2000.

[5] B. Naderi, M. Khalili, M. T. Taghavifard, V. Roshanaei, A variable neighborhood search for hybrid flexible flowshops with setup times minimizing total completion time. Journal of Applied Sciences, 8(16), 2843–5654, 2008.

[6] B. Naderi, M. Zandieh, S. M. T. Fatemi Ghomi, Scheduling sequence-dependent setup time job shops with preventive maintenance. International Journal of Advanced Manufacturing Technology, Article in press, 2008.

[7] B. Naderi, M. Zandieh, V. Roshanaei, Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness International Journal of Advanced Manufacturing Technology, Article in press, 2008.

[8] B. Naderi, M. Zandieh, S. M. T. Fatemi Ghomi, A study on integrating sequence dependent setup time flexible flow lines and preventive maintenance, Journal of Intellegent manufaturing, Article in press, 2008.

[9] B. Qian, L. Wang, D. X. Huang, X. Wang, Scheduling multi-objective job shops using a memetic algorithm based on differential evolution. International Journal of Advanced Manufacturing Technology, 35, 1014–1027, 2008.

[10] R. Ruiz, C. Maroto, J. Alcaraz, Two new robust genetic algorithms for the flowshop scheduling problem. Omega, 34, 461–476, 2006.

[11] R. Ruiz, C. J. Garica-Diaz, C. Maroto, Considering scheduling and preventive maintenance in the flow shop sequencing problem. Computers and Operations Research, 34, 3314–3330, 2007.

[12] R. Tavakkoli-Moghaddam, N. Safaei, F. Sassani, A memetic algorithm for the flexible flow line scheduling problem with processor blocking. Computers and Operations Research, 36(2), 402–414, 2009.

[13] M. Zandieh, S. M. T. Fatemi Ghomi, S. M. Moattar Husseini, An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times. Applied Mathematics and Computation, 180, 111–127, 2006.