J. Basic. Appl. Sci. Res., 1(12)3053-3057, 2011

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

A Memetic Algorithm for Hybrid Flow Shop Scheduling with Multiprocessor Task Problems

Mohammad Akhshabi [1*], Mostafa Akhshabi [2], Javad Khalatbari[3]

[1*]Department of Industrial Engineering, Islamic Azad University Ramsar Branch, Ramsar, Iran
[2] Department of Computer Engineering, Islamic Azad University Ramsar Branch, Ramsar, Iran
[3] Department of Management, Islamic Azad University Ramsar Branch, Ramsar, Iran

ABSTRACT

Multiprocessor task scheduling problem can be stated as finding a schedule for a general task graph to execute on a multiprocessor system so that the schedule length can be minimized. Hybrid Flow Shop Scheduling with Multiprocessor Task (HFSMT) problem is known to be NP-hard. In this study we present an effective memetic algorithm to solve HFSMT problem.
We lastly explain the computational experiments carried out to evaluate the performance of two algorithms (genetic algorithm and memetic algorithm) in terms of both the quality of the solutions produced and the efficiency.
These results demonstrate that the memetic algorithm produces better quality solutions and that it is very efficient.
KEY WORDS: Hybrid Flow Shop Scheduling, Multiprocessor task, memetic algorithm.

1. INTRODUCTION

Hybrid flow shop (HFS) scheduling problems commonly have a special structure combining some elements of both the flow shop and the parallel machine scheduling problem. In the classical flow shop, each stage consists of a single machine, and jobs visit the stages of the shop in the same order of the machines. In hybrid flow shop scheduling problems, it is assumed that a set J = {1, 2, . . ., n} of n jobs must be sequentially processed on a set of k-stage flow shops. Each job is processed first at stage one, then at stage two, and finally at stage k, and each i stage has m_i number of identical parallel processors, where i=1, 2, . . ., k. It is convenient to view a job as a sequence of k tasks – one task for each stage – where the processing of any task can commence only after the completion of the preceding task. Let $size_{ij}$ be the number of processors required to process the job j at stage i and P_{ij} be the processing time of the job j at stage i. In other words, the task i of job j has to be processed on $size_{ij}$ of m_i that are identical parallel processors of stage i for p_{ij} time units [1]. It is assumed that all jobs and machines are always available during the scheduling period and the preemption is not allowed. The objective of this study is to determine a schedule minimizing the maximum completion time (makespan).The multiprocessor task scheduling problem can be stated as the formation of a schedule for a general task graph to be executed on a multiprocessor system so that the schedule length can be minimized. This problem is a generalization of the classical machine scheduling problem, and in this paper, the multiprocessor task scheduling problem is considered to minimize the makespan in the hybrid flow shop environments.

Gupta [2] and Hoogeveen, as well as Lenstra and Veltman [3], proved that the two stage hybrid flow shop scheduling problem is NP-hard in the strong sense even if there is only one machine on the first stage and there are two machines on the second stage. Recently, the multi-stage hybrid flow shop scheduling problem has received attention due to its theoretical and practical importance.

The HFS scheduling has been widely applied in different manufacturing environments like in the iron and steel, the textile, the machine driven and the electronics industries [4]. In the literature, it can be clearly seen that most of the researchers have considered the two-stage hybrid flow shop scheduling problems [5]. Brah and Hunsucker [6] proposed a branch and bound algorithm to solve the hybrid flow shop problems. Portmann et al. [5] presented an improvement with genetic algorithms for the Brah and Hunsucker problems lower bound. Riane et al. [7] treated a problem of scheduling n jobs on a three-stage hybrid flow shop of particular structure and proposed two heuristic procedures to cope with the realistic problems. Moursli and Pochet [8] also proposed a branch and bound algorithm for the hybrid flow shop scheduling problem. Soewandi and Elmaghraby [9] resented several heuristic procedures of time complexity and several lower bounds of three-stage hybrid flow shop problems. Neron et al. [10] considered the hybrid flow shop scheduling problem with the objective of minimizing the makespan. They showed that the use of satisfaction tests and time bound adjustments based on the energetic reasoning and global operations could enhance the efficiency of branch and bound procedures for solving the hybrid flow shop scheduling problem optimally. A new approach built with the artificial immune system was proposed by Engin and Doyen [11] to solve the hybrid flow shop scheduling problems. Allaoui and Artiba [12] have studied the problem of scheduling a hybrid flow shop to minimize flow time and due date-based criteria under maintenance constraints. They used three dispatching rules LPT, SPT, and EDD, the simulated annealing heuristic,

***Corresponding Author:** Mohammad Akhshabi, Department of Industrial Engineering, Islamic Azad University Ramsar Branch, Ramsar, Iran. Email: m.akhshabi@toniau.ac.ir

and a flexible simulation model to solve the problem. Tang et al. [13] proposed a neural network model and algorithm to solve the dynamic hybrid flow shop scheduling problem. The most commonly used dispatching rules were used as benchmarks. They showed that the performance of the neural network approach was much better than that of the traditional dispatching rules. Zandieh et al. [14] proposed an immune evolutionary algorithm approach to the scheduling of a sequence dependent setup time hybrid flow shop problems. They compared the results obtained with the random key genetic algorithm presented previously.

Tang and Xuan [15] investigated the problem of scheduling n jobs on parallel identical machines in J successive stages with finite buffer capacities between consecutive stages in a real-time environment. They developed a Lagrangian relaxation algorithm combined with a speed-up dynamic programming approach.

Allahverdi and Anzi [16] addressed the problem of scheduling on a multi-stage parallel processor architecture in computer centers with the objective of minimizing average completion time of a set of requests. They proposed a new three phase heuristic approach. Allaoui and Artiba [17] investigated the two-stage hybrid flow shop scheduling problem with only one machine on the first stage and m machines on the second stage to minimize the makespan. They considered that each machine is subject to at most one unavailability period. They used the branch and bound model for that problem. Vob and Witt [18] considered a real world multi-mode multi-project scheduling problem in which the resources form a hybrid flow shop consisting of 16 production stages. They presented a mathematical model based on the resource constrained project scheduling problem to provide a formal description of the problem. Caricato et al. [19] focused on a variant of the hybrid flow shop problem, in which the jobs to be processed were grouped into predefined, ordered batches. They used a TSP-based approach to model the system. Janiak et al. [20] studied the flow-shop scheduling problem with parallel machines at each stage. The scheduling criterion consisted of three parts; the total weighted earliness, the total weighted tardiness, and the total weighted waiting time. They proposed some approximation algorithms for this problem with the complex, cost-related criterion.

Jin et al. [21] have considered the multistage hybrid flow shop scheduling problems. They studied two metaheuristic algorithms that were based on shop partitioning and simulated annealing, and the proposed approaches had been implemented in a real-life printed circuit board assembly line. Alaykıran et al. [22] presented an ant colony optimization model to solve HFS problems. Also, Kahraman et al. [23] proposed a genetic algorithm for HFS problems.

Due to the importance of the multiprocessor task scheduling problem, it has been extensively studied by many researchers. Edwin et al. [24] proposed a genetic algorithm for the multiprocessor scheduling problem that was based on the deterministic model. The proposed genetic algorithm was tested with random task graphs and the task graphs of the Newton–Euler inverse dynamic equations for the stand ford and elbow manipulators.

In this paper, an efficient memetic algorithm was developed to solve the hybrid flow shop scheduling with multiprocessor task. The effectiveness of the proposed method was tested with the hybrid flow shop scheduling with multiprocessor task problems. The computational results indicated that the proposed approach was effective.

The remainder of the paper is organized as follows. In Section 2, the problem definition is presented. The details of the genetic algorithm, its parameters are presented in Section 3. The computational results are given in Section 4, and finally the conclusions are made in Section 5.

2. Problem definition

The problem considered in this paper is formulated as follows:

There is a set J of n independent and simultaneously available jobs where each job is made of Multi-Processor Tasks (MPT) to be processed in a multi-stage flow-shop environment, where stage j consists of m_j identical parallel processors (j = 1, 2, ... , k). Each $MPT_i \in J$ should be processed on $p_{i,j}$ identical processors simultaneously at stage j without interruption for a period of $t_{i,j}$ (i = 1, 2, ... , n and j = 1, 2, ... , k). Hence, each MPT_i e J is characterized by its processing time, $t_{i,j}$, and its processor requirement, $p_{i,j}$. The scheduling problem is basically finding a sequence of jobs that can be processed on the system in the shortest possible time. The following assumptions are made when modeling the problem:

- _ All the processors are continuously available from time 0 onwards.
- _ Each processor can handle no more than one task at a time.
- _ The processing time and the number of processors required at each stage are known in advance.
- _ Set-up times and inter-processor communication time are all included in the processing time and it is independent of the job sequence.

Fig. 1 shows an example to scheduling done by the list scheduling algorithm. In this example, number of jobs is n = 5 and a machine is made of two stages k = 2 where each stage contains four identical processors. Table 1 depicts the list of jobs and their processing times and processor requirements at each stage for this example. For the schedule shown in Fig. 1, it is assumed that a job sequence is given as S1 = {2, 3, 1, 4, 5}. At stage 1, jobs are iteratively allocated to processors from the list starting from time 0 onwards. As job 2 is the first in the list, it is scheduled at time 0. It is important to note that although there are enough available processors to schedule job 1 at time 0 this will violate the precedence relationship established in the list. Therefore, job 1 is scheduled to time instance 3 together with job 3 and this does not violate the precedence relationship given in S1. Once all the jobs are scheduled at first stage, a new list is produced for the succeeding stage based on the completion of jobs at previous stage and the

precedence relationships given in S1. In the new list for stage 2, S2 = {2, 1, 3, 4, 5}, job 1 is scheduled before job 3 since it is available earlier than job 3. At time instance 7, jobs 3, 4 and 5 are all available to be processed. Job 3 is scheduled first since its completion time is earlier at stage 1. Although, there is enough processor to schedule job 5 at time 8 this will again violate the order given in list S1, hence it is scheduled together with job 4. In this particular example, jobs 4 and 5 will be the last to be mapped to stage 2 and the overall completion time of tasks will be 10 units.

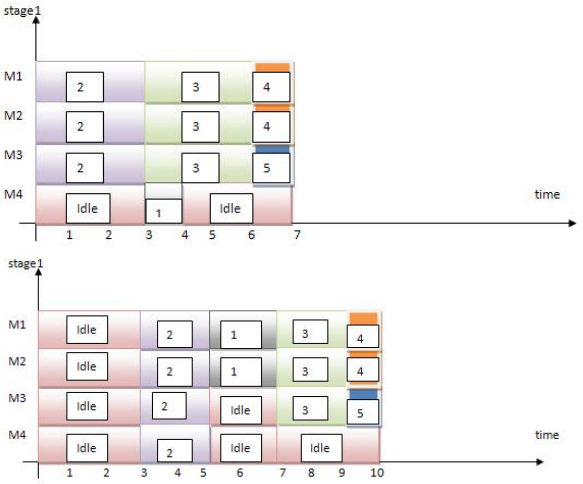

Fig. 1. The schedule of job sequence [2, 3, 1, 4, 5] after being allocated to processors of the multilayer system using the list scheduling algorithm. Idle periods of the processors are labeled as idle.

Table 1.Example jobs and their processing time and processor requirements at each stage.

Job#	Stage 1 (j = 1)		Stage 2 (j = 2)	
i	$p_{i,1}$	$t_{i,1}$	$p_{i,2}$	$t_{i,2}$
1	1	1	2	2
2	3	3	4	2
3	3	3	3	2
4	2	1	2	1
5	1	1	1	1

3. The Memetic Algorithm

A memetic algorithm is a metaheuristic based on a genetic algorithm in which a local search is performed on new solutions generated during the search. In the proposed memetic algorithm we combine the genetic algorithm with the branch-and-bound algorithm to benefit from the advantages of these approaches.

The main idea is, with probability p_{hs}, to improve an offspring with the branch-and-bound algorithm using constraint programming if the offspring is better than all of the chromosomes in the population before considering its inclusion in the population. Such a local search performed on the offspring will definitely increase the quality of the solutions obtained.

In our approach, we have chosen an incremental version of the genetic algorithm to replace the existing chromosomes with the selected better ones. In our opinion, this version, compared to a population replacement version, is able to take benefits from a new chromosome at the immediate next iteration by being able to potentially use it for a new crossover operation. Moreover, since some solutions will be generated by the branch-and-bound method, and these solutions will probably be improved a lot, we would like to be able to gain benefits from them as soon as possible.

First, an initial population of n_{pop} chromosomes is generated is randomly generated and two chromosomes, namely best and worst, are identified. The main loop is then repeated until some stopping conditions are met. Within the loop, the first operation performed is the selection of two chromosomes as parents for the crossover operator. The selection is made by the binary tournament technique. Next, the crossover operation is performed using the NXO operator and one offspring, c, is generated. The quality of the offspring, f (c), is immediately evaluated. Here f (c) refers to the value of C_l max(σ (c)) If the quality of the offspring is better than that of the best solution found so far, the mutation is not performed at this iteration, otherwise the mutation is performed with probability pm.

With a probability p_{hs}, we apply the local search using the constraint programming based branch-and-bound algorithm. One common rule applied for accepting the new chromosome to the population is that they at least improve the worst chromosome. If the new chromosome is accepted to the population, to keep a population of a fixed size, we have to remove one chromosome. To be sure to remove one of the worst, we can use the binary tournament as well.

EXPERIMENTAL RESULTS

We form a database from these randomly generated problem instances to provide a common basis for further studies and the data can be obtained from the authors. For the processor configurations, we considered two different

settings to see the effects of the processor configuration. In the first setting m_i, that is the number of processors at stage i, i = 1, 2, . . . , k, is randomly generated from the range [1; 5], whereas in the second setting mi = 5 is considered for all stages. The processor requirements, $size_{ij}$, and the processing times, p_{ij}, were generated randomly from the ranges of [1; m_i] and [1; 100], respectively, for i = 1, 2, . . . , k and j ∈ J. For each combination of n and k, 10 problem instances were generated (n = 5, 10, 20, 50, 100, k = 2, 5, 8). This resulted in 300 problems, each to be run 5 times due to the combination of five algorithms and two stopping criteria. Hence, in total, we performed 1500 runs in our computational experiments.

The performance of all the meta-heuristics described above is tested using intensive computational experiments. Similarly, performance of the basic PSO and the hybrid PSO algorithms, in minimizing the overall completion time of all jobs, is also tested using the same computational experiments. The effects of various parameters such as number of jobs and processor configurations on the performance of the algorithm are also investigated. The results are presented in terms of Average Percentage Deviation (APD) of the solution from the lower bound as given in Eq. (4):

$$APD=\frac{C_{max}-LB}{LB} \times 100$$

Here, C_{max} indicates the completion time of the jobs and LB indicates the lower bound calculated for the problem instance. The lower bounds used in this performance study were developed by O ́uz, et al. [25] and it is given with the following Eq. (5):

$$LB=\max \{\min \{\sum_{l=1}^{i-1} t_{lj}\} + 1/m_i \sum_{j \in J} p_{ij} \times t_{ij} + \min\{\sum_{l=i+1}^{l} t_{lj}\}\}$$

In the above formula, M and J represent the set of stages and set of jobs consecutively. Data set contains instances for two types of processor configurations:

(i) Random processor: In this problem set, the number of processors in each stage is randomly selected from a set of {1, . . . , 5}.

(ii) Fixed processor: In this case identical number of processors assigned at each stage which is fixed to five processors.

Table 2. APD of the algorithms for 5 random instances. Random processors case (m_j ~[1,5]).

k	n	GA	MA
2	5	1.79	1.47
	10	1.29	1.13
	20	0.89	0.87
	50	0.74	0.65
	100	0.65	0.34
5	5	10.14	8.25
	10	8.68	6.32
	20	6.49	5.47
	50	5.48	4.94
	100	5.06	4.09
8	5	14.06	12.29
	10	12.56	11.19
	20	10.28	9.63
	50	9.54	7.14
	100	7.26	6.33

Table 3. APD of the algorithms for 5 random instances. Fixed processor case (m_j = 5).

k	n	GA	MA
2	5	5.96	5.96
	10	4.14	4.14
	20	4.09	3.94
	50	3.54	3.05
	100	2.96	2.72
5	5	11.56	10.20
	10	10.25	9.36
	20	10.04	8.66
	50	9.51	8.09
	100	8.26	7.25
8	5	23.18	21.23
	10	22.65	20.16
	20	21.08	19.56
	50	18.59	17.84
	100	17.63	16.62

5. Conclusion

In this chapter, a scheduling problem, defined as hybrid flow-shops with multiprocessor tasks, is presented together with GA and MA. As the solution to this scheduling problem has merits in practice, endeavor to find a good solution is worthy. The genetic algorithm and memetic algorithms are employed to solve this scheduling problem. In this particular scheduling problem, a job is made up of interrelated multiprocessor tasks and each multiprocessor task is modeled with its processing requirement and processing time. The objective was to find a schedule in which completion time of all the tasks will be minimal. We observe that MA has a competitive performance as compared to GA.

As in many practical scheduling problems, it is likely to have precedence constraints among the jobs hence in future study hybrid flow-shops with precedence constraints will be investigated. In addition, other metaheuristics such as PSO can be applied to other scheduling problems and its performance can be exploited in other engineering problems.

REFERENCES

[1] C. O˘guz, F. Ercan, A genetic algorithm for hybrid flow shop scheduling with multiprocessor tasks, Journal of Scheduling 8 (2005) 323–351.

[2] J.N.D. Gupta, Two stage hybrid flow shop scheduling problem, Journal of Operational Research Society 39 (4) (1988) 359–364.

[3] J.A. Hoogeveen, J.K. Lenstra, B. Veltman, Minimizing the makespan in a multiprocessor flow shop is strongly NP-Hard, European Journal of Operational Research 89 (1996) 172–175.

[4] C. Kahraman, O. Engin, I˙. Kaya, R.E. Ozturk, Multiprocessor task scheduling in multistage hybrid flow-shops: a parallel greedy algorithm approach, Applied Soft Computing 10 (2010) 1293–1300.

[5] M.C. Portmann, A. Vignier, D. Dardilhac, D. Dezalay, Branch and bound crossed with GA to solve hybrid flow shops, European Journal of Operational Research 107 (1998) 389–400.

[6] S.A. Brah, J.L. Hunsucker, Branch and bound algorithm for flow shop with multiple processors, European Journal of Operations Research 51 (1991) 88–89.

[7] F. Riane, A. Artibs, S.E. Elmaghraby, A hybrid three stage flow shop problem: efficient heuristics to minimize makespan, European Journal of Operations Research 109 (1998) 321–329.

[8] O. Moursli, Y. Pochet, A branch and bound algorithm for the hybrid flow shop, International Journal of Production Economics 64 (2000) 113–125.

[9] H. Soewandi, S.E. Elmaghraby, Sequencing three stage flexible flow shops with identical machines to minimize makespan, IIE Transactions 33 (2001) 983–985.

[10] E. Neron, P. Baptiste, J.N.D. Gupta, Solving hybrid flow shop problem using energetic reasoning and global operations, Omega the International Journal of Management Science 29 (2001) 501–511.

[11] O. Engin, A. Doyen, A new approach to solve hybrid flow shop scheduling problems by artificial immune system, Future Generation Computer Systems 20 (2004) 1083–1095.

[12] H. Allaoui, A. Artiba, Integrating simulation and optimization to schedule a hybrid flow shop with maintenance constraints, Computers and Industrial Engineering 47 (2004) 431–450.

[13] L. Tang, W. Liu, J. Liu, A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment, Journal of Intelligent Manufacturing 16 (2005) 361–370.

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

[15] L. Tang, H. Xuan, Lagrangian relaxation algorithms for real-time hybrid flow shop scheduling with finite intermediate buffers, Journal of the Operational Resource society 57 (2006) 316–324.

[16] A. Allahverdi, A.F.S. Anzi, Scheduling multi-stage parallel-processor services to minimize average response time, Journal of the Operational Research Society 57 (2006) 101–110.

[17] H. Allaoui, A. Artiba, Scheduling two stage hybrid flow shop with availability constraints, Computers and Operations Research 33 (2006) 1399–1419.

[18] S. Vob, A. Witt, Hybrid flow shop scheduling as a multi-mode multiproject scheduling problem with batching requirements: a real-world application, International Journal of Production Economics 105 (2007) 445–458.

[19] P. Caricato, A. Grieco, D. Serino, Tsp-based scheduling in a batch-wise hybrid flow-shop, Robotics and Computer Integrated Manufacturing 23 (2007) 234–241.

[20] A. Janiak, E. Kozan, M. Linchtenstein, C. O˘guz, Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion, International Journal of Production Economics 105 (2007) 407–424.

[21] Z. Jin, Z. Yang, T. Ito, Metaheuristic algorithms for the multistage hybrid flow shop scheduling problem, International Journal of Production Economics 100
(2006) 322–334.

[22] K. Alaykıran, O. Engin, A. Doyen, Using ant colony optimization to solve hybrid flow shop scheduling problems, International Journal of Advanced Manufacturing

[23] C. Kahraman, O. Engin, I˙. Kaya, M.K. Yılmaz, An application of effective genetic algorithms for solving hybrid flow shop scheduling problems, International Journal of Computational Intelligence Systems 1 (2008) 134–147.

[24] S.H.H. Edwin, N. Ansari, H. Ren, A genetic algorithm for multiprocessor scheduling, IEEE Transactions on Parallel and Distributed Systems 5 (2) (1994) 113–120.

[25] Ouz, C., Zinder, Y., Do, V., Janiak, A., & Lichtenstein, M. (2004). Hybrid flow shop scheduling problems with multiprocessor task systems. European Journal of Operations Research, 152, 115–131.