International Journal of Industrial Engineering & Production Research

March 2014, Volume 25, Number 1
pp. 55-63

http://IJIEPR.iust.ac.ir/

pISSN: 2008-4889

A Two-Stage Hybrid Flowshop Scheduling Problem with Serial Batching

E. Ghafari & R. Sahraeian*

Esmaeel Ghafari, *Master of Science Student, Shahed University, Tehran, Iran, esmail_ghafary@yahoo.com*
Rashed Sahraeian, *Assistant Professor of Industrial Engineering, Shahed University, Tehran, Iran, sahraeian@shahed.ac.ir*

KEYWORDS

scheduling,
Hybrid flowshop,
Serial batching,
Genetic algorithm,
Taguchi method

ABSTRACT

In this paper the problem of serial batch scheduling in a two-stage hybrid flow shop environment with minimizing Makesapn is studied. In serial batching it is assumed that jobs in a batch are processed serially, and their completion time is defined to be equal to the finishing time of the last job in the batch. The analysis and implementation of the prohibited transference of jobs among the machines of stage one in serial batch is the main contribution of this study. Machine set-up and ready time for all jobs are assumed to be zero and no Preemption is allowed. Machines may not breakdown but at times they may be idle. As the problem is NP-hard, a genetic algorithm is developed to give near optimal solutions. Since this problem has not been studied previously, therefore, a lower bound is developed for evaluating the performance of the proposed GA. Many test problems have been solved using GA and results compared with lower bound. Results showed GA can obtain a near optimal solution for small, median and large size problems in reasonable time.

1. Introduction

As industries are facing increasingly competitive situations, many classical manufacturing systems shift to novel environments such as hybrid flow shop in which a combination of flow shop and parallel machines operates together. A hybrid flow shop environment is similar to flow shop but in at least one stage the number of machines is greater than one. There are two types of batch productions, namely, serial batches and parallel batches. In serial batches a number of jobs within the same batch are processed sequentially, while in parallel batches a group of jobs go through a machine and are processed simultaneously Ribas et al. [1]. Implementations of hybrid flow shop can be found in various industries including automotive industry, chemical industry, metallurgical industry, and iron manufacturing. In this paper, hybrid flow shop with serial batching has been considered. It is assumed that jobs in a batch are processed serially, and their completion time is defined to be equal to the finishing time of the last job in the batch.

The processing time of a batch equals to the sum of the processing time of all the jobs in the batch. The literature in this subject consists of two sections; first section is considering the hybrid flow shop scheduling and a second one is considering batch scheduling. A survey of scheduling literature in hybrid flow shop environment was conducted by Ribas et al [1]. They considered previous research works in three different points of view, including of processing complexity, scheduling criteria and approaches to hybrid flow shop (HFS) scheduling.

Researches show HFS problems in processing complexity situations are usually grouped into three categories: (1) two-stage HFS, (2) three-stage HFS,

* **Corresponding author: Rashed Sahraeian**
Email: *sahraeian@shahed.ac.ir*
Paper first received March 04, 2012, and in accepted form May 13, 2013.

and (3) k-stage HFS. Research works in scheduling criteria show two types of categories: (1) based on flow time and (2) based on due dates. Researches in terms of approaches to the hybrid flow shop scheduling are grouped into three categories: (1) Exact algorithms, (2) Heuristics and (3) Metaheuristics approaches.

In last decade most of researches work on HFS, impose some constraints on the problem to get closer to the real world problems.

Botta-Genoulaz [2] has considered scheduling in hybrid flow shop environment with precedence constraints and time lags to minimize maximum lateness. He presented six heuristics to solve this problem). Sawik [3] has used mixed integer programming for scheduling flexible flow lines with Limited intermediate buffers to minimize C_{max}. Riane et al. [4] have presented an integrated production planning and scheduling system for hybrid flowshop organizations. Gupta et al. [5] have considered heuristics for hybrid flow shop scheduling with controllable processing times and assignable due dates. They proposed constructive algorithms using job insertion techniques and iterative algorithms based on local search.

Oguz et al. [6] have proposed Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flowshop. Engin and Doyen [7] have proposed an artificial immune system approach, for solving the hybrid flow shop scheduling problem with minimizing maximum completion times. Oguz et al. [8] have considered hybrid flow shop scheduling problem with multiprocessor task system and minimizing maximum completion time.

Problems with multi-processor task systems relax the limitation of the classical parallel model by permitting tasks that require more than one processor simultaneously. Morita and Shio [9] have used hybrid branch and bound method with genetic algorithm for flexible flowshop scheduling problem.

Tang et al. [10] have used heuristic combined artificial neural networks to schedule hybrid flow shop with sequence dependent setup times. Ruiz and Maroto [11] have used a genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility (2006). Tang et al. [12] have considered a new lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time. Zandieh et al. [13] have used an immune algorithm approach to hybrid flow shops scheduling with sequence dependent setup times. Vob and Witt [14] studied hybrid flow shop scheduling as a multi-mode multi-project scheduling problem with batching requirements by minimizing the weighted tardiness.

Their mathematical model was based on the well-known resource constrained project scheduling problem to provide a formal description. Chen and Chuen [15] considered Bottleneck-based heuristics to minimize total tardiness for the flexible flow line with unrelated parallel machines. Figielska [16] has used a genetic and a simulated annealing algorithm combined with column generation technique for solving the problem of scheduling in the hybrid flowshop with additional resources.

Naderi et al. [17] have considered an improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness. Jabbarizadeh et al. [18] Studied Hybrid flexible flowshops with sequence-dependent setup times and machine availability constraints .they proposed 3 heuristics and 2 metaheuristics based on genetic algorithm and simulated annealing.

Behnamian and Fatemi Ghomi [19] proposed Hybrid flowshop scheduling with machine and resource-dependent processing times with minimize makespan and total resource allocation costs.

The solution methodology, as can be seen from the literature, for the hybrid flow shop problems can be classified in three groups; (1) the exact method such as branch and bound technique and mathematical programming, (2) the metaheuristic approach such as tabu search, simulated annealing and genetic algorithm, (3) the heuristic algorithms. There are many researches in the literature considering batch scheduling. Sawik [20] proposed a mixed integer programming formulations for serial batch scheduling in flexible flow lines with limited intermediate buffers. Yuan et al. [21] have studied the unbounded single machine parallel batch scheduling problem with family jobs and release dates to minimize C_{max}.

They proposed a dynamic programming algorithm to solve it. Li and Yuan [22] have considered a dynamic programming algorithm in order to minimize the C_{max}, machine occupation time and stocking cost in the single machine parallel batch scheduling problem. Ruiz and Maroto [23] considered a genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. Husseinzadeh Kashan et al. [24] considered a hybrid genetic heuristic for scheduling parallel batch processing machines with arbitrary job sizes. They proposed a hybrid genetic heuristic (HGH) to minimize makespan objective. Nong et al. [25] have studied online scheduling in the single-machine environment considering parallel-batching of jobs.

Bellanger and Oulamara [26] They have considered Scheduling hybrid flowshop term with parallel batching machines and compatibilities, where two stages with the second one containing batching machines is solved using tailored heuristics.

The remainder of the paper is organized as follows. Section 2 describes the problem in detail and presents a lower bound. Section 3 explains the developed genetic algorithm.

The calibration of the proposed algorithm is presented in Section 4. In Section 5, experimental design and comparison of the proposed algorithm with lower

bound reported. Section 6 concludes the paper and provides some directions for future studies.

2. Problem Statement

2.1. Problem Description

In this paper, a hybrid flow shop serial batch (HFSB) is put forth for study. It is assumed that the completion time of all jobs in a batch is defined as the finishing time of the last job in the batch, i.e., the processing time of a batch equals to the sum of the processing times of all jobs in the batch. It is supposed that there are *n* jobs brought into the batches which should be processed on *k* stages. In stage *j* we have a number of machines; all the jobs in batches have the same process route.

Each job should undergo all stages and be processed in each stage by only one machine. Machine set-up and ready time for all jobs are assumed to be zero and no Preemption is allowed. Machines may not breakdown but at times they may be idle.

Furthermore, it is also assumed that machines in each stage are identical and Jobs arrive in batches with a different batch size. To the best of our knowledge, no general method has been proposed for the HFS with serial batch. In this study, it is considered two stages in HFSB environment which consists of *m* identical parallel machines in each stage. The first aim is to schedule the batches on the machines of stage one and second one is to schedule jobs which released from stage one to the machines of stage two such that the makespan is minimized.

Here, a two-stage HFSB system is considered which encompasses restriction of jobs transportation among machines in stage one, e.g. obstruction of jobs transportation among machines in x-ray stage due to thick insulation walls. Therefore, in stage one all jobs of the selected batch should be performed on only one of the machines. Consequently, in stage two jobs can be done on each of machines. The processing times and the batches size are known and non-identical.

Considering the well-known three field notation α/β/ϒ for scheduling problems and the extension for hybrid flow shops proposed by Ribas et al. [1], the real production problem considered here can be noted as $HF2(P_m^1, P_m^2)|S-batch|C_{max}$.

The proposed notation shows serial batch arrival in a hybrid flow shop environment with two stages that each of them have *m* identical parallel machine. The aim of this paper is not to propose a mathematical formulation, but a novel problem scheme is proposed and a GA is developed to solve the proposed problem. The proposed algorithm is intended to determine batches and jobs sequence in a two stages hybrid flow shop.

The evolutionary search approaches have been successfully applied to a number of combinatorial optimization problems (Husseinzadeh Kashan et al. [24]; Jabbarizadeh et al. [18]). An evolutionary search approach based on a GA can generate a good solution to the model in a reasonable computational time.

2.2. NP-Hardness of the Problem

Gupta [27], as well as Hoogeveen et al. [28], proved that the two stage hybrid flow shop scheduling problem is NP-hard when the objective is to minimize the makespan even if there is only one machine on the first stage and there are two machines on the second stage. On the other hand if there is only one stage in which there are identical parallel machines, the problem under study HFSB reduces to $P \parallel C_{max}$ which itself is NP-hard Garey and Johnson [29]. Therefore, looking this way also proves the NP-hardness of the problem.

2.3. A Lower Bound on the Optimal Makespan

In this section a lower bound (LB) is developed for the $HF2(P_m^1, P_m^2)|S-batch|C_{max}$ problem based on lower bounds of $Pm\|C_{max}$. The lower bound is utilized to evaluate performance of results obtained from proposed GA. A general lower bound for $Pm\|C_{max}$ is presented by Haouari et al. [30] as follows, which p_i shows the processing times of jobs:

$$LB = \left\lceil \frac{1}{m} \sum_{i=1}^{n} p_i \right\rceil \quad (1)$$

Based on general LB in Eq. (1) a lower bound for $HF2(P_m^1, P_m^2)|S-batch|C_{max}$ can be derived by Eq. (2):

$$LB = \left\lceil \frac{1}{m} \sum_{i=1}^{n} p_{i1} \right\rceil + \min(p_{i2}) \quad (2)$$

Where p_{i1} denotes processing time of job *i* on stage 1, p_{i2} processing time of job i on stage 2 , *m* denotes number of machines on stage 1, and n shows number of all jobs.

Example1.2. to illustrate the LB considering a $HF2(P_3^1, P_3^2)|S-batch|C_{max}$ problem with 5 batches, the processing time of each batch in stages, and batch sizes are given in table 1 and LB computed as follow:

$$LB = \left\lceil \frac{1}{3} \sum_{i=1}^{25} p_{i1} \right\rceil + \min(p_{i2}) =$$

$$\left\lceil \frac{1}{3} x[(5x7) + (6x9) + (4x8) + x(7x10) + (3x5)] \right\rceil +$$

$$4 = 73$$

Tab. 1. Processing times of jobs for example 1.2

Batch number	Batch size	Job number(i)	P_{i1}	P_{i2}
1	5	1-2-3-4-5	7	4
2	6	6-7-8-9-10-11	9	7
3	4	12-13-14-15	8	5
4	7	16-17-18-19-20-21 -22	10	6
5	3	23-24-25	5	8

This lower bound is obtained from Eq. (1) by adding the minimum processing time of jobs in stage 2.

3. Genetic Algorithm

In Ruiz et al. [31] they considered several genetic algorithms for solving the flow shop environment. The good results acquired allow us to think of an adaptation of some of these algorithms for the problem considered in this paper. A genetic algorithm differs from other metaheuristics in the fact that it works with a set of encoded solutions to the problem, called population. Every solution or chromosome in the population is evaluated and determined a fitness value. The idea is that the best individuals should also be the fittest. In a GA the population evolves over generations until some stopping pattern is reached.

A generation begins with the selection mechanism that randomly picks some members from the population. The fittest chromosomes should have a greater chance of being selected. Then, these individuals mate in a process called crossover and generate some new individuals, which are often called offspring. Then, some offspring might undergo another process called mutation in which some of the parts of the chromosomes or genes can change as in natural species mutation.

Lastly, the new population is evaluated again and the whole process is repeated [32] .The effectiveness of a GA depends on the choice of its operators and parameters; that is, encoding, selection, population size, crossover, mutation and their probabilities. In the following, we explain how these operators and parameters are set.

1-Chromosomes: The *strings* or *chromosomes* that represent a solution to the problem being studied are the structure blocks of GAs. In our problem, a chromosome represents a schedule. A set of these chromosomes is called a *population* [33].

2-Fitness function: GA manipulate solutions at the chromosome level based on fitness values to distribute similarities among the high performance strings to the next population using reproduction operators such as mutation and crossover. The fitness function corresponds to the objective function under consideration [33].

3-*Reproduction*: Individual solutions are selected from the current population based on their fitness function values and new solutions (offspring) generated by recombining the genes of these solutions. Selection of the parent strings may be biased in consideration of solutions with better fitness values. Crossover and mutation are the main mechanisms for recombining the chromosomes. Crossover combines the genes of two parents to produce offspring, while mutation randomly alters individual genes to ensure diversity of solutions [33].

Based on the fitness values of the objective function, in this case C_{max}, two best parents are selected by employing a tournament-based selection for crossover [32]. The crossover mechanism is the single-point crossover and an example of this mechanism, for the problem under study, is shown in Table 2. The chromosome of the two parents chosen for crossover and the crossover point is shown in the table. The chromosome of each parent can be divided into two portions; one portion to the left of the crossover point and the other to the right of the crossover point. Each offspring is formed by first copying the left portion of the corresponding parent's chromosome as it is. The remaining portion of the offspring is obtained by rearranging the jobs on the right side of the corresponding parent's chromosome as per the sequence in which they appear on the other parent which was chosen for the crossover. For example, the jobs in the right part of parent 1 (5-2-10-7-9) are rearranged according to the sequence in which they appear in the chromosome of parent 2 and appended to offspring 1. Thus, 5-2-10-7-9 is rearranged as 10-7- 2-5-9 and appended to 4-6-8-3-1 to obtain the chromosome of offspring 1.

Tab. 2. Crossover mechanisms.

Before crossover						**Crossover Point**				
Parent 1	4	6	8	3	**1**	5	2	10	7	9
Parent 2	3	10	7	4	**2**	6	8	5	9	1
After crossover										
Offspring 1	4	6	8	3	1	10	7	2	5	9
Offspring 2	3	10	7	4	2	6	8	5	9	1

The two offspring obtained are later mutated, with a defined mutation probability (P_m), in order to ensure that their characteristics are different from the parents, thereby ensuring diversity. The mutation process employed by this problem is a one opt mutation [32]. An example of the mutation process is shown in Table 3. The two offspring shown in Table 2 are mutated as shown in Table3. The mutation point is randomly chosen and the jobs on either side of the mutation point are changed.

Tab. 3 Mutation mechanisms.

Before Mutation						Mutation Point				
Offspring 1	4	**6**	8	3	1	10	7	2	5	9
Offspring 2	3	**10**	7	4	2	6	8	1	5	9
After Mutation										
Offspring 1	4	8	6	3	1	10	7	2	5	9
Offspring 2	3	7	10	4	2	6	8	1	5	9

The Pseudo-Code for the GA is as Follows

1. Prepare initial generation of chromosomes.
2. Set generation = 0.
3. Assign the processing times of jobs in each batch and calculate the C_{max}.
4. Apply a tournament based approach to select the two parents.
5. Apply single point crossover and a one opt mutation with a mutation probability P_m.
6. Compare and replace the offspring to form a new set of parents for the next generation.
7. Repeat steps 3–6 until generation = N (iteration).

4. Parameter Tuning

In this section, we aim at analyzing the behavior of the proposed GA considering the above mentioned operators and parameters. Doing so, there exist various approaches to statistically design an experimental investigation. Each of these approaches is effective depending on the situation of experiment. Although the most widely used approach is a full factorial design, this approach is not always efficient because it becomes increasingly difficult to perform investigation when the number of factors is significantly high. To reduce the number of required tests, fractional factorial experiment (FFE) was developed [34]. FFE allow only a portion of the total possible combinations to estimate the main effect of factors and some of their interactions.

Taguchi [35] developed a family of FFE matrices which finally reduce the number of experiments, but still provide sufficient information. In Taguchi method, orthogonal arrays are used to study a large number of decision variables with a small number of experiments. Taguchi separates the factors into two main groups: controllable and noise factors. Noise factors are those over which we have no direct control. Since elimination of the noise factors is impossible, the Taguchi method look for minimize the effect of noise and to determine optimal level of important controllable factors based on the concept of robustness [36].

Besides determining the optimal levels, Taguchi identifies the relative significance of individual factors in terms of their main effects on the objective function. Taguchi has created a transformation of the repetition data to another value which is the measure of variation. The transformation is the signal-to-noise (S/N) ratio which explains why this type of parameter design is called robust design [37,36]. Here, the term ‘‘signal’’ denotes the admirable value (mean response variable) and ‘‘noise’’ denotes the undesirable value (standard deviation). So the S/N ratio indicates the amount of variation presents in the response variable. The aim is to maximize the signal-to-noise ratio.

Taguchi classifies objective functions into three categories: the smaller-the-better type, the larger-the-better type, and nominal-is-best type. Since almost all objective functions in scheduling are classified in the smaller-the-better type, their corresponding S/N ratio [36] is:

$$S/N ratio = -10\log_{10}(objective function)^2$$

As explained earlier, in this study, the GA factors are: iteration (A), population size (B) and mutation probability in each solution (C). Different levels of above mentioned factors are shown in Table 4.

Tab. 4. Factors and their levels

Factor	Symbol	Level	Type
Iteration	A	4	A(1)-1000 A(2)-10000 A(3)-50000 A(4)-100000
Population size	B	4	B(1)-5 B(2)-8 B(3)-10 B(4)-15
Mutation probability	C	4	C(1)-0.4 C(2)-0.6 C(3)-0.85 C(4)-1.00

Tab. 5. The modified orthogonal array L_{16}

Trial	Levels of control factors		
	A	**B**	**C**
1	A(1)	B(1)	C(1)
2	A(1)	B(2)	C(2)
3	A(1)	B(3)	C(3)
4	A(1)	B(4)	C(4)
5	A(2)	B(1)	C(2)
6	A(2)	B(2)	C(1)
7	A(2)	B(3)	C(4)
8	A(2)	B(4)	C(3)
9	A(3)	B(1)	C(3)
10	A(3)	B(2)	C(4)
11	A(3)	B(3)	C(1)
12	A(3)	B(4)	C(2)
13	A(4)	B(1)	C(4)
14	A(4)	B(2)	C(3)
15	A(4)	B(3)	C(2)
16	A(4)	B(4)	C(1)

The associated degree of freedom for these three factors is 15. So, the selected orthogonal array should have a minimum of 16 rows and three columns to assess the three factors. From standard table of orthogonal arrays, the L16 is selected as the fittest orthogonal array design which carries out all our

minimum requirements. But this orthogonal array still entails some modifications to conform itself to our experimental design. Table 5 shows the modified orthogonal arrayL16.

Tab. 6. Analysis of Variance for SN ratios

Factor	DF	Seq SS	Adj SS	Adj MS	F	P-Value
A	3	0.00528	0.00528	0.00176	58.34	0.000
B	3	0.00022	0.00022	0.00007	2.44	0.163>0.05
C	3	0.00031	0.00031	0.00010	3.43	0.093>0.05
Residual Error	6	0.00018	0.00018	0.00003		
Total	15	0.00599				

To conduct the experiment, we generate an instance as follows: a hybrid flow shop environment with 2 stages, 5 machines in each stage, and 20 batches comprised of 300 jobs. Processing time of batches in each stage is generated from uniform distribution ranging (1, 15). We implement the algorithms in MATLAB 7.0 and run on a PC with 2.0 GHz Intel Core i5 and 4 GB of RAM memory. After obtaining the results of Taguchi experiment, results are transformed into S/N ratio. Fig. 1 shows the average S/N ratio obtained at each level of the different factors. As indicated in Fig.1, the optimal level of the factors A, B and C are A (4), B (4) and C (1), respectively.

To explore the relative significance of individual factors in terms of their main effects on the objective function, the analysis of variance (ANOVA) test is conducted. The results of analysis are presented in Table 6. Iteration has the greatest effects on the quality of the algorithm. After iteration factor, mutation probability is placed in the second rank; and factor B (population size) has the least effect on performance of our GA. This interestingly shows the GA is almost independent of the choice of iteration.

In sum, the chosen levels are as follows: iteration: 100000, population size: 15, and mutation probability: 0.4.

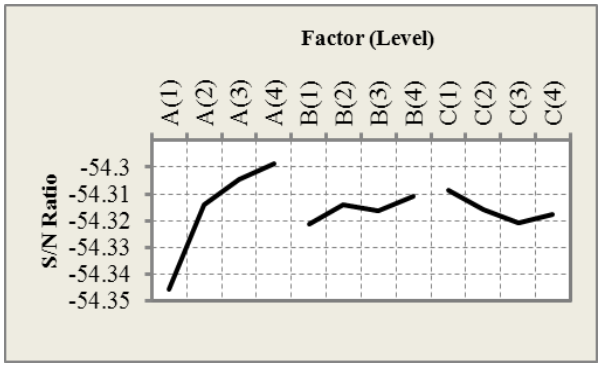

Fig.1: The mean s/n ratio plot for each level of the factors

5. Computational Experiments

In this section the performance of the proposed algorithm would be evaluated and compared by conducting some experiments. Several test problems with considering some parameters were generated and the results analyzed. The following subsections describe the details of the experiments.

5.1. Generating Data

To evaluate the performance of the algorithm in varied situations, four parameters were characterized in generating the test problems. These parameters include the number of jobs *n*, the number of machines on stages *m*, the processing times p_{ij} and the batch size s_j. Factors and their levels are shown in Table 7. For example, for parameter job, three levels of low, medium, and high, with 100, 300, and500 jobs respectively, were considered. Processing times, sizes of the batches and number of machines in two stages were generated from the discrete uniform distribution.

Tab. 7. Different Parameters for test problems.

Parameters	Levels
Number of jobs	100, 300, 500
P_{ij}	Uniform [1,15]
Batch size	Uniform [2,10], Uniform [10,15] and Uniform [15,30]
Number of machines	Uniform [2, 5] and Uniform [5,10]

There are 18 types of problems (3×3×2×1=18) generated when combining different amounts given for these 4 parameters, and 40 data sets are generated randomly for each type, creating 720 problems all together. The proposed algorithm is coded in MATLAB 7 and implemented on an Intel Corei5, 2.40 GHz PC.

Table 8 presents the results obtained from the GA for 10-job instances. Column 1 represents the run code for the instances. For example, J1m1s1p1 implies that the test instance is for 100 jobs with number of machines, batch sizes and processing times generated at level 1. Columns 2 and 3 report the average of makespans and average of relative deviation from lower bound (DEVLB) for GA. Table 9 and 10 present the results for the 300 and 500 jobs instance, respectively.

Tab. 8. Results for 100 job instances

Run code	Average of makespans	Average of DEVLB
(1)	(2)	(3)
J1m1s1p1	273	0.07
J1m1s2p1	167	0.05
J1m1s3p1	241	0.08
J1m2s1p1	140	0.02
J1m2s2p1	116	0.15
J1m2s3p1	146	0.05

Tab. 9. Results for 300 job instances

Run code	Average of makespans	Average of DEVLB
(1)	(2)	(3)
J1m1s1p1	666	0.03
J1m1s2p1	804.03	0.01
J1m1s3p1	756.13	0.07
J1m2s1p1	578.11	0.10
J1m2s2p1	460.02	0.12
J1m2s3p1	458.14	0.09

5.2. Experimental Results

All randomly generated problems (720) were solved by the proposed genetic algorithm. In other words, GA solved 720 test problems. Since there is no general algorithm for two stages HFSB problem, we cannot carry out any direct comparison with other researchers or methods, and also there is no "standard" or benchmark test data in the open literature about this research. So to measure the performance of our GA the results has been compared with the lower bound obtained in section 2.

The comparison is implemented by defining a percentage of relative deviation from lower bound (DEVLB) for each solution as follows:

$$Relative\ deviation = \frac{obtained\ makespan\ from\ the\ GA - lower\ bound}{lower\ bound}$$

Results showed the following outcomes can be concluded for GA. By increasing the number of jobs, the average of makespans enhanced. As the number of machines grew, the average of makespans decreased. Furthermore, the computational results showed GA can obtain a near optimal solution for small, median and large size problems.

Tab. 10. Results for 500 job instances

Run code	Average of makespans	Average of DEVLB
(1)	(2)	(3)
J1m1s1p1	1413.17	0.06
J1m1s2p1	1367.26	0.01
J1m1s3p1	993.06	0.02
J1m2s1p1	994.12	0.09
J1m2s2p1	688.08	0.12
J1m2s3p1	511	0.08

6. Conclusions and Future Research

In this paper, a serial batch scheduling problem in a two-stage hybrid flow shop environment with the objective function of minimizing makespan has been proposed. No previous work in the literature of scheduling has dealt with the serial batching problem of this kind (to the best of our knowledge). Since it is a generalized form of $P||C_{max}$ then the problem is NP-hard. A lower bound for this problem has been proposed based on other researches and our heuristic. The genetic algorithm has been used to solve the problem. In order to evaluate the performance of the GA, a large number of randomly problems generated and results compared with lower bound. Results showed GA has obtained a near optimal solution. In the future research, other scheduling objectives such as minimizing the sum of earliness/tardiness and maximum lateness can be tested. The restriction of machine eligibility is quite common in practical. As a consequence, the development of a heuristic for problems with machine eligibility is a practical area of research. Finally, the concept of batch arrivals can be extended to batch delivery (Wangand and Cheng [38]), which is also encountered quite often in the real world.

References

[1] Ribas, I., Leisten, R., Framinan, JM., *"Review and Classification of Hybrid Flow Shop Scheduling Problems Drom a Production System and a Solutions Procedure Perspective."* Computers & Operations Research, Vol. 37, pp. 1439–1454, 2010.

[2] Botta-Genoulaz, V., *Hybrid Flow Shop Scheduling with Precedence Constraints and Time Lags to Minimize Maximum Lateness*. International Journal of Production Economic, Vol. 64(1), pp. 101-111, 2000.

[3] Sawik, T., *Mixed Integer Programming for Scheduling Flexible Flow Lines with Limited Intermediate buffers*. Math Comput Model, Vol. 31(13), pp. 39-52, 2000.

[4] Riane, F., Artiba, A., Iassinovsk, iS., *An Integrated Production Planning and Scheduling System for Hybrid Flow Shop Organizations.* International Journal of Production Economic , Vol. 74(1), pp. 33-48, 2001.

[5] Gupta, J., Kruk, K., Lau, V., Werner, F., Sotskov, Y., *Heuristics for Hybrid Flow Shops with Controllable Processing Times and Assignable Due Dates*. Computers & Operations Research, Vol. 29(10), pp. 1417-1439, 2002.

[6] Oguz, C., Ercan, MF., Cheng, TCE., Fung, YF., *Heuristic Algorithms for Mmultiprocessor Task Scheduling in a Two-Stage Hybrid Flow Shop*. European Journal of Operational Research, Vol. 149(2), pp. 390-403, 2003.

[7] Engin, O., Doyen, A., *A New Approach to Solve Hybrid Flowshop Scheduling Problems by Artificial Immune System.* Future Generation Comput System, Vol. 20(6), pp. 1083-1095, 2004.

[8] Oguz, C., Zinder, Y., Do, V., Janiak, A., *Lichtenstein, M., Hybrid Flowshop Scheduling Problems with Multi Processor Task Systems.* European Journal of Operational Research, Vol. 152(1), pp. 115-131, 2004.

[9] Morita, H., Shio, N., *Hybrid Branch and Bound Method with Genetic Algorithm for Flexible Flowshop Scheduling Problem.* Jsme Int J Ser C Mech Syst Mach Elm Manuf, Vol. 48(1), pp. 46-52, 2005.

[10] TangL, X., Zhang, YY., *Heuristic Combined Artificial Neural Networks to Schedule Hybrid Flowshop with Sequence Dependent Setup Times*. Adv Neural Netw—Isnn, Vol. 3496, pp. 788-793, 2005.

[11] Ruiz, R., Maroto, C., A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research, Vol. 169(3), pp. 781–800, 2006.

[12] Tang, L., Xuan H., *Lagrangian Relaxation Algorithms for Real-Time Hybrid Flowshop Scheduling with Finite Intermediate Buffers.* J Oper Res Soc, Vol. 57(3), pp. 316-324, 2006.

[13] Zandieh, M., Fatemi Ghomi, SMT., Moattar Husseini, SM., *An Immune Algorithm Approach to Hybrid Flow Shops Scheduling with Sequence Dependent Setup Times*. Appl Math Comput, Vol. 180(1), pp. 111-127, 2006.

[14] Stefan, Vob., Andreas Witt., *Hybrid Flow Shop Scheduling as a Multi-Mode Multi-Project*. International Journal of Production Economic, Vol. 105, pp. 445–458, 2007.

[15] Chun-Lung, Chen ., Chuen-Lung, Chen., *A Bottleneck-Based Heuristic for Minimizing Makespan in a Flexible Flow Line with Unrelated Parallel Machines.* Computers and Operations Researc, Vol. 36(11), pp. 3073-3081, 2009.

[16] Figielska, Ewa., *A Genetic Algorithm and a Simulated Annealing Algorithm Combined with Column Generation Technique for Solving the Problem of Scheduling in the Hybrid Flowshop with Additional Resources.* omputers and Industrial Engineerin, Vol. 56(1), pp. 142-151, 2009.

[17] Naderia, B., zandieh, M., khaleghi Goshe balagh, A., roshanaei, V., *An Improved Simulated Annealing for Hybrid Flowshops with Sequence-Dependent Setup and Transportation Times to Minimize Total Completion Time and Total Tardiness.* Expert Systems with Application, Vol. 36, pp. 9625–9633, 2009.

[18] Jabbarizadeh, F., Zandieh, M., Talebi, D., *Hybrid Flexible Flowshops with Sequence-Dependent Setup Times and Machine*. Computers & Industrial Engineerin, Vol. 57, pp. 949–957, 2009.

[19] Behnamian, J., Fatemi Ghomi, S.M.T., *Hybrid Flowshop Scheduling with Machine and Resource-Dependent*. Applied Mathematical Modelling, Vol. 35(3), pp. 1107-1123, 2011.

[20] Sawik, T., *An Exact Approach for Batch Scheduling in Flexible Flow Lines with Limited Intermediate Buffers*. Mathematical and Computer Modelling, Vol. 36, pp. 461–471, 2002.

[21] Yuan, J.J., Liu, Z., Ng, C.T., Cheng, T.C.E., *The Unbounded Single Machine Parallel Batch Scheduling Problem with Family Jobs and Release Dates to Minimize Makespan*. Theoretical Computer Science, Vol. 320, pp. 199–212, 2004.

[22] Wenhua, L., Jinjiang, Y., *Single Machine Parallel Batch Scheduling Problem with Release Dates and Three Hierarchical Criteria to Minimize Makespan*, machine occupation time and stocking cost. International Journal of Production Economic, Vol. 102(1), pp. 143-148, 2006.

[23] Ruiz, R., Concepcion, M., *A Genetic Algorithm for Hybrid Flowshops with Sequence Dependent Setup Times and Machine Eligibility*. European Journal of Operational Research, Vol. 169(3), pp. 781-800, 2006.

[24] Husseinzadeh Kashan, A., Karimi, B., Jenabi , M., *A Hybrid Genetic Heuristic for Scheduling Parallel batch Processing.* Computers & Operations Research, Vol. 35, pp. 1084 – 1098, 2008.

[25] Nong, Q., Yuan, J., Fu, R., Lin, L., Tian, J., *The Single Machine Parallel Batching on Line Scheduling Problem with Family Jobs to Minimize Makespan*. International Journal of Production Economic, Vol. 111, pp. 435–440, 2008.

[26] Bellanger, A., Oulamara, A., *Scheduling Hybrid Flowshop with Parallel Batching Machines and Compatibilities.* Computers & Operations Research, Vol. 36(6), pp. 1982-1992, 2009.

[27] Gupta, J.N.D., *Two Stage Hybrid Flow Shop Scheduling Problem.* Journal of Operational Research Society, Vol. 39(4), pp. 359–364, 1988.

[28] Hoogeveen, J.A., Lenstra, J.K., Veltman, B., *Minimizing the Makespan in a Multiprocessor Flow Shop is Strongly NP-Hard.* European Journal of Operational Research, Vol. 89, pp. 172–175, 1996.

[29] Garey, M.R., Johnson, D.S., *Strongly NP-Completeness Results: Motivation Examples and Implication.* Journal of Association for Computing Machinery, Vol. 25, pp. 499–508, 1978.

[30] Haouari, M., Gharbi, A., Jemmali M., *Tight Bounds for the Identical Parallel.* International Transactions in Operational research, Vol. 13, pp. 529-548, 2006.

[31] Ruiz, R., Maroto, C., Alcaraz, J., *Solving the Flowshop Scheduling Problem with Sequence Dependent Setup Times using Advanced Metaheuristics*. European Journal of Operational Research, Vol. 165, pp. 34–54, 2005.

[32] Michalewicz, Z., *Genetic Algorithms + Data Structures=Evolution Programs*, third ed. Berlin : Springer-Verlag, 1996.

[33] Goldberg DE., *Genetic Algorithms in Search*, optimization, and machine learning. Addison-Wesley, 1989.

[34] W.G. Cochran., G.M, Cox., *Experimental Designs*, 2nd ed. USA : Wiley, 1992.

[35] Ross, R.J., *Taguchi Techniques for Quality Engineering*. USA : McGraw-Hill, 1989.

[36] Phadke, M.S., *Quality Engineering Using Robust Design*. USA : Prentice-Hall, 1989.

[37] Al-Aomar, R., *Incorporating Robustness into Genetic Algorithm Search of Stochastic Simulation Outputs*. Simulation Modeling Practice and Theory, Vol. 14, pp. 201–223, 2006.

[38] Wang, X., Cheng, T.C.E., *Heuristics for Parallel Machine Scheduling with Job Class Setups and Delivery to Multiple Customers.* International Journal of Production Economic, Vol. 119, pp. 199–206, 2009.