Scheduling of a Hybrid Flow Shop with Multiprocessor Tasks by a Hybrid Approach Based on Genetic and Imperialist Competitive Algorithms

Javad Rezaeian[a,*], Hany Seidgar [b], Morteza Kiani [c]

[a] *Assistant Professor, Department of industrial engineering, Mazandaran University of Science and Technology, Babol, Iran*
[b] *MSc, Department of industrial engineering, Mazandaran University of Science and Technology, Babol, Iran*
[c] *MSc, Department of industrial engineering, Mazandaran University of Science and Technology, Babol, Iran*
Received 12 September, 2012; Revised 29 January, 2013; Accepted 20 February, 2013

Abstract

This paper presents a new mathematical model for a hybrid flow shop scheduling problem with multiprocessor tasks in which sequence dependent set up times and preemption are considered. The objective is to minimize the weighted sum of makespan and maximum tardiness. Three meta-heuristic methods based on genetic algorithm (GA), imperialist competitive algorithm (ICA) and a hybrid approach of GA and ICA are proposed to solve the generated problems. The performances of algorithms are evaluated by computational time and Relative Percentage Deviation (RPD) factors. The results indicate that ICA solves the problems faster than other algorithms and the hybrid algorithm produced best solution based on RPD.
Keywords: Hybrid flow shop scheduling; Multi-processor tasks; sequence dependent setup time; preemption.

1. Introduction

The flow shop scheduling is one of the main problems in the category of machine scheduling. This problem was introduced by Johnson (1954) in which a set of jobs flow through multiple stages in the same machine order, where each stage consists of only one machine (processor). Nowadays, the need to increase capacity or to balance the capacities of the stages has led to the duplication of some machines in some stages. This extended layout is usually addressed as Hybrid Flow Shop (HFS), Flexible Flow Shop (FFS), or flow shop with parallel machines. This paper refers to this shop floor configuration as HFS. Thus, the HFS problem has two basic characteristics as follows: (1) a set of *n* jobs is sequentially processed in a series of *m* stages, and (2) at least one of the stages has two or more machines in parallel. The early work on HFS was conducted by Rao (1970) and the first model on HFS was proposed by Arthanari and Ramamurthy (1971). The structure of a hybrid flow shop system is illustrated in Fig 1. Gupta (1998) considered a special case with two-stage and one single machine in the first stage and two identical machines in the second stage, and showed that the case was NP-hard. In literature, the studies about Hybrid flow shops with sequence dependent setup times (SDST) are scarce. Kurz and Askin (2004) formulated the SDST/HFS as an Integer Programming (IP) model. Allahverdi et al. (2008) presented an extended survey regarding setup time consideration, with and without sequence dependency. Zandieh et al. (2006) proposed an immune algorithm, and compare it against the random keys genetic algorithm of Kurz and Askin (2004). Naderi et al. (2009) proposed a dynamic dispatching rule and an iterated local search algorithm for the problem. Zandieh and Rashidi (2009) presented an effective hybrid genetic algorithm for HFS with SDST and processor blocking. Ruiz et al (2010) and Ribas et al (2010) conducted a comprehensive review of relevant research on HFS problems, revealing that among them only a few studies discussed multiprocessor tasks where each job is processed on a number of identical machines simultaneously at each stage. Oguz et al. (2003) examined the multiprocessor task in a two-stage HFS problem motivated by a computer vision system. The problem can be encountered in a number of industrial environments, such as berth allocation of container terminals, real time machine-vision systems, and work force management. Allahverdi and Anzi (2006) addressed the problem of scheduling on multi-stage parallel processor architecture in computer centers with the objective of minimizing average completion time of a set of requests. Ying and Lin (2009) developed the first

* Corresponding author E-mail: j_rezaeian@ustmb.ac.ir

simple constructive heuristic algorithm for the mentioned problem. Kahraman et al. (2010) developed a parallel greedy algorithm for the HFS problem with multiprocessor tasks. Lahimer et al. (2011) developed the climbing depth-bounded adjacent discrepancy search which was shown to be effective for both small and large problems. Recently, metaheuristic algorithms have been developed for multiprocessor task-scheduling in an HFS system, including simulated annealing (SA) by Wang et al. (2011), genetic algorithm by Engin (2011), particle swarm optimization (PSO) algorithm by Chou (2013), and A hybrid artificial bee colony algorithm with bi-directional planning by Lin et al. (2013). In this study a hybrid flow shop problem with multiprocessor task and sequence dependent set up times is presented in which job preemption is allowed and meta-heuristic approaches of GA, ICA and HGICA are used to solve the problem.

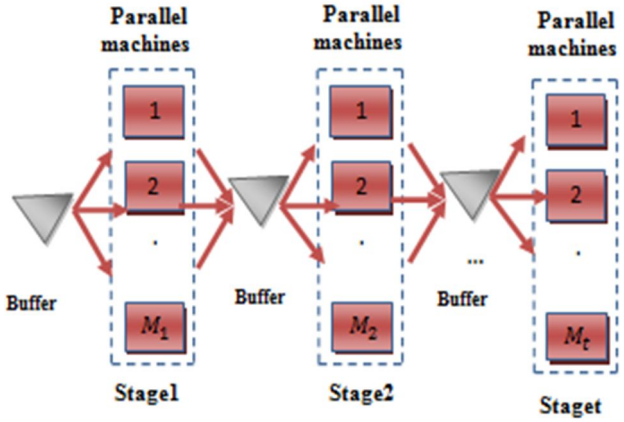

Fig 1. Hybrid flow shop system

The remainder of this study is organized as follows: Section 2 describes and formulates the hybrid flow shop problem. The Proposed GA and ICA are described in Sections 3 and 4 respectively, and the hybrid approach is described in Section 5. Design of experiments and parameter tuning are presented in Section 6. The computational experiments on the generated problems and analysis of the results are provided in Section 7. Finally, conclusions and future research are presented in Section 8.

2. Problem Description and Mathematical Model

In this study, a hybrid flow-shop problem with sequence dependent set up times is considered in which jobs are multi-task and preemption is allowed. Each job contains several tasks that may be operated in separated times and tasks of a job can be processed simultaneously on parallel machines. There is no precedence constraint between tasks of a job. Each task must be operated on only one machine and a machine can process only one task at a time. Without loss of generality, at the first and the end of sequences of jobs on each machine at each stage, a dummy job is inserted. In this problem, two kinds of set up time are regarded. The first one is the initial set up time for arrival tasks at the beginning of the operations on a machine. The second one is the dependent setup time that is considered between two different successive jobs on a machine. Job preemption is allowed but each task of a job is not separable. It is presumed that there is enough capacity between every two consecutive stages and transportation time is negligible. Due date of each job is predefined and the objective function is to minimize the sum of weighted makespan and maximum tardiness. Table 1 shows the characteristics of an example problem in which two jobs with three tasks should be processed on three machines at two stages. An achievable schedule of the considered problem is illustrated in Fig 2, in which (2-1) indicates the first task of job 2. After initial setup times at stage one, the first and second tasks of job 2 are processed on machines 1 and 2 respectively. Subsequently the third task of job 2 is done on machine 2, and job 2 will be completed and moved to next stage. After preparation of machine 3, the third and first tasks of job 1 are processed respectively. The second task of job 1 is processed on machine 1 after dependent setup time (job2-job1) and job 1 is completed by then. The scheduling of jobs at stage 2 can be inferred the same as stage 1.

Table 1
Illustrative example

	Stage1						Stage2					
Jobs	Job1			Job2			Job1			Job2		
Due date	25			20			25			20		
Initial setup time	4			3			7			4		
Tasks	1	2	3	1	2	3	1	2	3	1	2	3
Processing time	2	3	6	9	7	3	6	6	3	6	6	5
Dependent Setup times	(job1-job2)			(job2-job1)			(job1-job2)			(job2-job1)		
	5			3			4			3		

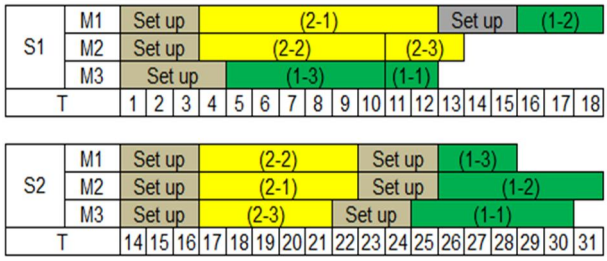

Fig. 2. the schedule of the illustrative example

The following notations are used through the paper.

2.1. Notations

Indices

i,j indices for jobs ($i,j = 1,2,\ldots,n$)
m index for machines ($m = 1,2,\ldots,M_t$)
t index for stages ($t = 1,2,\ldots, NS$)

Parameters

Q_{it} Number of tasks of job *i* at stage *t*
S_{ij}^{t} Setup time between jobs *i* and *j* at stage *t*
S_{jt} Setup time of job *j* at stage *t*
P_{jt}^{q} Process time of qth part of job *j* at stage *t*
d_j Due date for job *j*
NS Number of stages
α *A* real number between 0 and1
M A large number

Decision variables

C_{jt}^{q} Completion time of *qth* part of *jth* job at stage *t*
C_{jt} Completion time of job *j* at period *t*
$Cmax$ Maximum completion time
$Tmax$ Maximum Tardiness
T_j Tardiness of job *j*

X_{ijmt}^{pq}: 1 if pth part of ith job is processd before qth part of jth job on mth machine at stage t; 0 otherwise

Model:

$$\min z = \alpha Cmax + (1-\alpha)Tmax \quad (1)$$

S*t:*

$$\sum_{m=1}^{Mt}\sum_{i=0}^{n}\sum_{p=1}^{Pit} X_{ijmt}^{pq} = 1 \quad \forall j,q,t \quad (2)$$

$$\sum_{m=1}^{Mt}\sum_{j=1}^{n+1}\sum_{q=1}^{Qjt} X_{ijmt}^{pq} = 1 \quad \forall i,p,t \quad (3)$$

$$\sum_{j=1}^{n+1}\sum_{q=1}^{Qjt}\sum_{p=1}^{Pit} X_{0jmt}^{pq} = 1 \quad \forall m,t \quad (4)$$

$$\sum_{i=0}^{n}\sum_{q=1}^{Qjt}\sum_{p=1}^{Pit} X_{i\,n+1\,m\,t}^{pq} = 1 \quad \forall m,t \quad (5)$$

$$\sum_{i=0}^{n}\sum_{p=1}^{Pit} X_{ikmt}^{pr} = \sum_{j=1}^{n+1}\sum_{q=1}^{Qjt} X_{kjmt}^{rq} \quad \forall m,t,k,r \quad (6)$$

$$C_{jt}^{q} \geq \left(P_{jt}{}^{q} + set_{jt}\right) + \left(X_{0jm1}^{pq} - 1\right).M \quad \forall j,p,q,m \quad (7)$$

$$C_{jt}^{q} \geq (C_{it}^{p} + P_{jt}{}^{q} + setup_{ij}^{t}) + \left(X_{ijmt}^{pq} - 1\right).M \quad \forall i > 0, j,p,q,m,t \quad (8)$$

$$C_{jt}^{q} \geq \left(C_{jt-1} + P_{jt}{}^{q} + set_{jt}\right) + \left(X_{0jmt}^{pq} - 1\right).M \quad \forall j,p,q,m,t > 1 \quad (9)$$

$$C_{jt}^{q} \geq (C_{jt-1} + P_{jt}{}^{q} + setup_{ij}^{t}) + \left(X_{ijmt}^{pq} - 1\right).M \quad \forall i > 0, j,p,q,m,t > 1 \quad (10)$$

$$C_{jt} \geq C_{j\,NS}^{q} \quad \forall j,t,q \quad (11)$$

$$Cmax \geq C_{j\,NS} \quad \forall j \quad (12)$$

$$T_j = C_{jv}^{q} - d_j \quad \forall j,q \quad (13)$$

$$Tmax \geq T_j \quad \forall j \quad (14)$$

$$T_j, C_{jt}, C_{jt}^{q}, Tmax \geq 0, X_{ijmt}^{pq} = \{0,1\}$$

Eq. (1) expresses the objective function, minimizing the weighted sum of makespan and maximum tardiness. Eq. (2) and (3) determine the sequence of tasks for each job. Eqs. (4) and (5) ensure that the precedence of the first task and successor of the last task on a machine are dummy jobs. Eq. (6) ensures the construction of a consistent sequence at every stage. Eqs. (7)- (10) determine the tasks completion time of a job. Eq. (11) computes the completion time of a job at each stage. Eq. (12) returns the value of makespan. Eq. (13) calculates the tardiness of each job. Eq. (14) shows the maximum tardiness.

By increasing the characteristic size of the problem and the growth of complexity, the search space will expand and the computational time will increase exponentially. To solve such problems, the meta-heuristic algorithms have been employed due to their power in seeking the search space and finding high-quality solutions in a reasonable amount of time.

3. Proposed Genetic Algorithm

The main idea of the genetic algorithm was at first proposed by Holland (1962). This algorithm which is inspired by the natural evolution mechanism, devises a guided random approach to search the solution space. In the following sections, the details of the GA for solving the proposed problem are described.

3.1. Chromosome encoding and initial solution

In the first step of genetic algorithm, an initial population of chromosomes must be generated in order to form the first generation of solutions. The initial population can be produced using different mechanisms among them random production is utilized to make sure of population diversity according to Gen (1997). In this study, each chromosome is shown by a

2×(($\sum_{i=1}^{n}\sum_{t=1}^{ns} Q_{it}$)+ns-1) matrix. At first row, the value of each gene represents the machine number and genes of the second row indicate the tasks of jobs. For illustration, consider a problem with 2 jobs, each one containing two tasks, 2 stages and 3 machines at each stage. A typical chromosome is shown in Fig. 3. The symbol (*) cuts the chromosome in different stages. For instance, at stage 1 the first task of job 1 and the second task of job 2 are done on machine 1. The first task of job 2 is processed on machine 2 and the second task of job 1 is processed on machine 3 respectively. At stage 2, the first task of job 2 and second task of job1 are done on machine 3 and the processes of second task of job 2 and first task of job 1 are done on machine 2 respectively.

Machines	1	2	1	3	*	3	2	2	3
Jobs	1	2	2	1	*	2	2	1	1

Fig. 3. Illustration of chromosome

3.2. Selection mechanism

There are some selection mechanisms with the responsibility to choose parent chromosomes from the population to form the mating pool. In this study the roulette wheel selection mechanism is used in which the solutions with higher fitness values have more chance to be selected. Since our objective is to minimize the weighted sum of makespan and maximum tardiness, the fitness value is computed by Eq.(15):

$$Fitness\ Value=\frac{1}{\alpha C_{max}+(1-\alpha)Tmax} \quad (15)$$

3.3. Genetic operators

3.3.1. Reproduction

A number of chromosomes with the highest fitness values from the current generation will be copied to the next one. This is called the elite strategy and it is an effort to wish for production of better individuals in the next generation out of the current high quality chromosomes.

3.3.2. Crossover operator

Crossover is the main operator of the GA. It works with chromosomes that are selected from the selection mechanism as parents, and mixes them to produce children. Crossover operator makes an optimistic attempt to swap parents sections and produce children that inherit promising features from their parents.

In this study the modified position based crossover (PBX) is applied through following steps:

Step 1: A set of random positions are selected from parent 1.

Step 2: The genes located in the selected positions will be copied to the same positions into the offspring. Then the positions in parent 2 containing equal values with the selected genes will be identified and omitted.

Step 3: The unfilled positions of the offspring will be placed by the genes of the remainder positions from the parent 2 respectively.

The mentioned steps are shown schematically in Fig. 4.

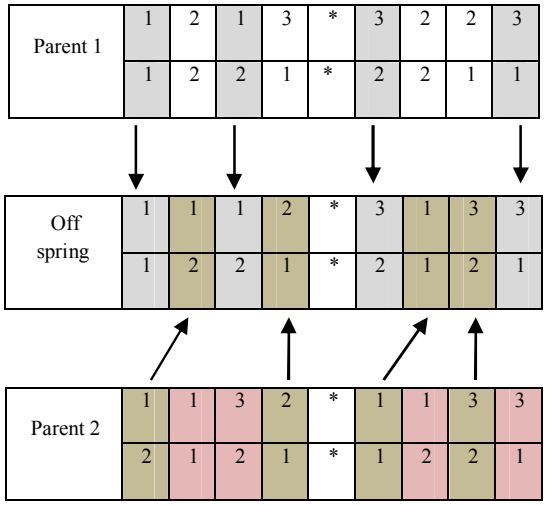

Fig. 4. Crossover operator

3.3.3. Mutation operator

The mutation operator revolves the structure of some chromosomes in the generation in order to guaranty diversification of solutions. It also helps to prevent from falling into local optimum trap .The simple mutation operator that is used here, randomly selects two positions in a chromosome and exchange their genes with each other. (Fig. 5)

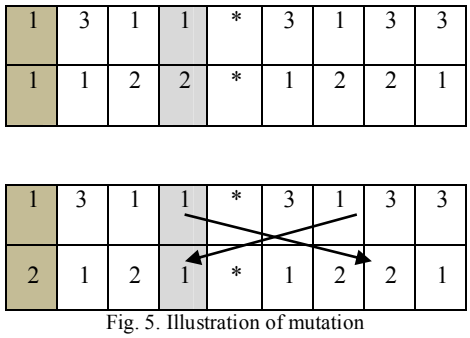

Fig. 5. Illustration of mutation

This procedure will be repeated in all iterations and GA will be terminated after predefined number of generations and final best solution will be taken.

4. Imperialist Competitive Algorithm (ICA)

ICA is one of the newest methods in Meta-heuristic field that was suggested and developed by Atashpaz-Gargari (2007). This method is based on human social and political behavior. Similar to GA, the ICA is a population based algorithm and the population is consisted of some countries which are classified in two

categories: imperialists and colonies. Imperialist is a country that rules a number of countries which are called colonies. In other words, the policies, culture, religion and other social measurements of a colony are delineated by the imperialist in power.

After producing initial population randomly, at first step each country must be specified to either imperialist or colony. The countries have several attributes including culture, language, religion, economic policy, etc. and are shown by vector F:

$F= [f1, f2 \ldots f_N]$

In which f_i indicates *i'th* attribute of a country.

Next, the fitness value of countries will be calculated and those with lowest cost are determined as imperialists and other countries will be considered as colonies. Then the numbers of each imperialist's colonies are calculated by Eqs. (16)- (19):

$$c_i = Cost\ (country) \quad (16)$$

$$C_n = max_i\{c_i\} - c_n \quad (17)$$

$$P_n = \left|\frac{C_n}{\sum C_i}\right| \quad (18)$$

$$number\ of\ clonies_n = round(P_n . N_{col}) \quad (19)$$

Where c_n and C_n are the cost of *nth* imperialist and its normalized cost respectively. P_n is imperialist's power and imperialists with lower cost are more powerful to increase their chance of getting more colonies. After specifying imperialists and their colonies, ICA devices assimilation policy and revolution operators start to search for better countries.

Assimilation policy is performed on all colonies to form new ones. In this policy each imperialist absorbs colonies by making changes in their attributes such as social, cultural, regional and etc. Next, the cost of new colonies that are absorbed to the current imperialist must be recalculated. If the new colony is better than its imperialist, they will be exchange with each other immediately. The second operator is revolution that creates diversification in countries. This operator randomly selects two attributes in a country and exchanges their values. Then, the power of each emperor is calculated by Eqs. (20)- (22):

$$T.C_n = Cost(Imperialist_n) + \zeta . mean\{Cost(Colonies\ of\ Empire_n)\} \quad (20)$$

$$N.T.C_n = max_i\{T.C_i\} - T.C_n \quad (21)$$

$$P_{Pn} = \left|\frac{N.T.C_n}{\sum_{i=1}^{Nimp} N.T.C_i}\right| \quad (22)$$

Where $T.C_n$ equivalent to total is cost of nth empire and ζ is a positive number which is considered to be lower than1, $N.T.C_n$ is power of *nth* empire and P_{Pn} is possession possibility of each emperor.

The emperor with the lowest cost will be considered as the weakest emperor. Then the other emperors compete with each other to take possession of the weakest colonies of the weakest emperor. ICA uses a selection process to choose an emperor that will pick up the mentioned colonies.

The vector R created uniformly distributed random number as following:

$R= [r_1, r_2, \ldots, r_{Nimp}]$

From vector D in Eq. (23), the emperor with the highest value will be selected and the weakest colony of the weakest emperor will be assigned to it.

$$D=P-R=[D_1, D_2, \ldots, D_{Nimp}]= [p_{P1} - r_1, p_{P2} - r_2, \ldots, p_{p_{Nimp}} - r_{p_{Nimp}}] \quad (23)$$

The stopping criterion of the algorithm is the remaining of one emperor.

5. Hybrid Approach Based on GA and ICA

In proposed hybrid approach based on genetic and imperialist competitive algorithms (HGICA), the best solutions of GA after several iterations will be saved and considered as starting solutions of ICA. Then ICA uses these solutions as initial population to seek for better solutions. In this approach, while GA is implemented to seek for high quality solutions in the search space, the ICA is used in an effort to intensify the search within the region of solution space close to high quality solutions. The procedure of this approach is illustrated in Fig. 6.

6. Design of Experiments

In this section, the behavior of the proposed algorithms under different levels of factors is studied in order to achieve the desired effects. In Table 2, there are five factors of GA and six factors of ICA. The approximate levels of the factors are obtained experimentally and then Response Surface Methodology (RSM) is applied to find the appropriate value of each factor for each size of problem. RSM is a technique to determine and represent the cause-and-effect relationship between true mean responses and input control variables influencing the responses as a two-or three-dimensional hyper surface Montgomery (1991).

In order to find the best values of the factors, the problems are grouped into two categories: A and B.

Category A includes problems with 15 jobs or lower, and problems with more than 15 jobs are presented by B.

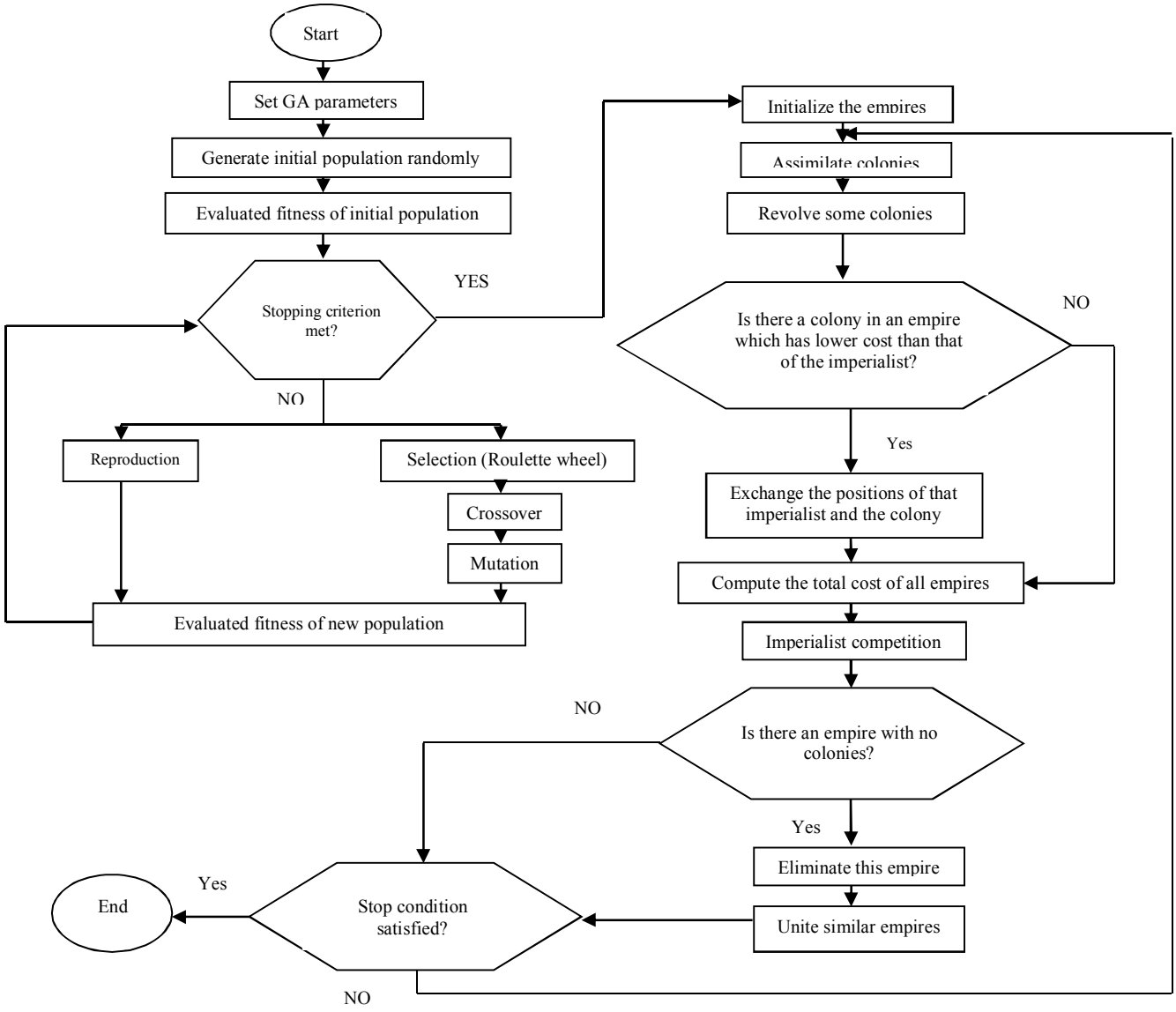

Fig. 6. Hybrid approach, based on GA and ICA

Table 2
Range of factors for GA and ICA

Method	Factors	Level
GA	Pop-Size	(20-50)
	Cross over Ratio	(0.8-0.9)
	Mutation Ratio	(0.03-0.05)
	PBX Ratio	(0.2-0.4)
	Iteration	(200-500)
ICA	Number of Countries	(150-350)
	Number of Iteration	(200-500)
	Number of Imperialists	(6-10)
	Assimilation Ratio	(0.8-0.9)
	Revolution Ratio	(0.1-02)
	Zeta (ξ)	(0.04-0.06)

For each category of problem and each algorithm, some experiments are designed and the best parameters are obtained and revealed in Tables 3 and 4.

Table 3
Tunned parameters of GA

Due date	Process Time	First Setup Time	Middle Setup Time	Tasks of Jobs
[20,100]	[5,20]	[1,5]	[1,6]	[1,4]

Table 4
Tunned parameters of ICA

Category	POP-Size	Crossover	Mutation	PBX	Iteration
A	50	0.8	0.03	0.2	500
B	48	0.8	0.03	0.2	500

Table 5
Problem characteristics

Category	Country	Imperialist	Assimilation	Revolution	Zeta	Iteration
A	150	6	0.2	0.18	0.48	200
B	150	6	0.4	0.2	0.47	200

7. Computational Results

In this section, many test problems have been designed to evaluate the proposed algorithms. The data of the problems are randomly generated using uniform distribution with parameters based on Table 5, and all experiments have been performed on Intel CORE i5 CPU and 4GB of RAM under Windows 7. As mentioned in previous section, the problems are grouped in categories A and B, and for simplicity and without loss of generality, it is assumed that there are two stages in each problem and the number of machines in stages are shown by (m1, m2) in Table 6. CPU time (second) and Related Percentage of Deviation (RPD) factors are implemented to compare the efficiency of the algorithms.

$$RPD=\frac{Alg_{sol}-Min_{sol}}{Min_{sol}} \quad (24)$$

Where, Alg_{sol} is the value of the solution obtained by related algorithm and Min_{sol} is the Minimum obtained value among the solutions of algorithms. In order to compare the performance of the algorithms, the obtained results are arranged according to the solving methods and mean of RPD presented in Table 7.

Table 6
Results of expriments comparsion between GA, ICA, HGICA

problem	(m_1,m_2)	GA			ICA			HGICA		
		OB.value	RPD	Time	OB.value	RPD	Time	OB.value	RPD	Time
A3	(2,2)	60.4	0	62	60.4	0	8	60.4	0	84
	(2,4)	48.2	0	49	48.2	0	9	48.2	0	64
	(4,2)	45.6	0	68	45.6	0	12	45.6	0	97
	(4,4)	32.4	0	63	32.4	0	12	32.4	0	102
A4	(2,2)	121.2	0	110	121.2	0	41	121.2	0	147
	(2,4)	104.8	0	85	121	0.154	31	104.8	0	96
	(4,2)	101.6	0	82	101.6	0	30	101.6	0	124
	(4,4)	62.6	0	117	69.2	0.105	45	62.6	0	135
A5	(2,2)	121.6	0	63	132.8	0.092	17	121.6	0	221
	(2,4)	68	0.03	115	68.8	0.042	21	66	0	101
	(4,2)	121	0	124	121	0	22	121	0	197
	(4,4)	63.2	0	126	63.2	0	20	63.2	0	211
A6	(2,2)	141.8	0.007	212	148.2	0.052	19	140.8	0	227
	(2,4)	84.4	0.014	137	86.4	0.038	30	83.2	0	216
	(4,2)	134.6	0	341	149.8	0.112	17	134.6	0	275
	(4,4)	70.2	0.014	295	72	0.04	19	69.2	0	274
A7	(2,2)	169.2	0.024	190	204.6	0.238	56	165.2	0	295
	(2,4)	98.8	0.046	216	116.4	0.238	86	94.4	0	277
	(4,2)	160.2	0.165	229	180.2	0.311	85	137.4	0	337
	(4,4)	83.4	0.039	151	100.2	0.311	38	80.2	0	341
A8	(2,2)	189.8	0.049	195	189.6	0.049	89	180.8	0	464
	(2,4)	128.2	0.064	352	135.2	0.122	140	120.4	0	457
	(4,2)	192.8	0.081	274	178.2	0	91	179.8	0.052	397
	(4,4)	94.8	0.041	103	98	0.076	73	91	0	312
A9	(2,2)	223.8	0.144	490	255.2	0.144	35	195.6	0	505
	(2,4)	187.6	0.07	222	204.2	0.165	47	175.2	0	273
	(4,2)	192.8	0.018	237	221	0.166	28	189.4	0	574
	(4,4)	120.8	0	215	126.8	0.049	95	120.8	0	527
A10	(2,2)	276.4	0.097	352	278.4	0.096	121	252	0	649
	(2,4)	220.6	0.048	355	232.4	0.098	48	210.4	0	340
	(4,2)	227.6	0.038	366	240.8	0.098	112	219.2	0	626
	(4,4)	155.4	0.07	215	156	0.074	79	145.2	0	624
B15	(2,2)	403.6	0.049	314	437.4	0.137	82	384.6	0	512
	(2,4)	330.4	0.094	317	341.2	0.129	99	302	0	469
	(4,2)	298.2	0.189	345	317.8	0.268	75	250.6	0	625
	(4,4)	211.4	0	347	228	0.078	111	233.2	0.103	567
B20	(2,2)	553.6	0.057	717	583.6	0.115	234	523.4	0	964
	(2,4)	435.2	0.069	720	471.6	0.158	222	407	0	1014
	(4,2)	403	0.204	685	451.4	0.349	188	334.6	0	987
	(4,4)	281.4	0.05	628	312.4	0.166	124	267.8	0	998
B25	(2,2)	570	0.027	886	572	0.03	174	555	0	1013
	(2,4)	492.6	0.164	903	507.4	0.199	216	423	0	1042
	(4,2)	374	0.16	1211	401	0.244	233	322.2	0	1645
	(4,4)	281.6	0.25	1058	342.8	0.522	406	225.2	0	1784
B30	(2,2)	762.8	0.07	1919	816.8	0.145	392	712.8	0	2042
	(2,4)	635	0.09	1259	667.2	0.145	214	582.2	0	2280
	(4,2)	567	0.132	1920	622.8	0.243	290	500.8	0	2326
	(4,4)	394.2	0.087	1342	435.8	0.201	440	362.6	0	2035

Table 7
Comparsion of GA, ICA, HGICA based on mean of RPD

Problem	Method	Performance				$\overline{RPD}$
		(2,2)	(2,4)	(4,2)	(4,4)	
A3	GA	0	0	0	0	0
	ICA	0	0	0	0	0
	HGICA	0	0	0	0	0
A4	GA	0	0	0	0	0
	ICA	0	0.154	0	0.105	0.064
	HGICA	0	0	0	0	0
A5	GA	0	0.03	0	0	0.007
	ICA	0.092	0.042	0	0	0.033
	HGICA	0	0	0	0	0
A6	GA	0.07	0.014	0	0.014	0.024
	ICA	0.052	0.038	0.112	0.076	0.084
	HGICA	0	0	0	0	0
A7	GA	0.024	0.046	0.165	0.039	0.068
	ICA	0.238	0.238	0.311	0.311	0.274
	HGICA	0	0	0	0	0
A8	GA	0.049	0.064	0.081	0	0.048
	ICA	0.049	0.122	0	0.076	0.061
	HGICA	0	0	0.052	0	0.013
A9	GA	0.144	0.07	0.018	0	0.058
	ICA	0.144	0.165	0.166	0.049	0.131
	HGICA	0	0	0	0	0
A10	GA	0.097	0.048	0.038	0.07	0.063
	ICA	0.096	0.098	0.098	0.074	0.091
	HGICA	0	0	0	0	0
problem	Method	Performance				$\overline{RPD}$
		(2,2)	(2,4)	(4,2)	(4,4)	
B15	GA	0.049	0.094	0.189	0	0.083
	ICA	0.137	0.129	0.268	0.078	0.153
	HGICA	0	0	0	0.103	0.025
B20	GA	0.017	0.069	0.204	0.05	0.085
	ICA	0.115	0.158	0.349	0.166	0.197
	HGICA	0	0	0	0	0
B25	GA	0.027	0.164	0.16	0.25	0.15
	ICA	0.03	0.199	0.244	0.522	0.248
	HGICA	0	0	0	0	0
B30	GA	0.07	0.09	0.132	0.087	0.094
	ICA	0.145	0.145	0.24	0.201	0.182
	HGICA	0	0	0	0	0

Analyzing the experiments results in Tables 6 and 7, it can be noticed that GA almost provides higher quality solutions and it provides lower mean of RPD than ICA (Fig. 7). Although GA outperforms ICA in quality of solutions, but in CPU time measurement, ICA works better and finds solutions in significantly lower amount of time (Fig. 8). Accordingly, these algorithms are hybridized in order to make high quality solutions in reasonable amount of time. Comparing the results of Table 7, it is shown that HGICA performs better than both GA and ICA, but it consumes more time. In order to confirm higher performance of the HGICA than GA, equal condition should be adjusted for both of them. So, the GA procedure was allowed to search the solution space for more computational time, but there wasn't any considerable improvement in the results. As it is illustrated in Fig. 9, HGICA finds much better solutions than GA in the same period of time.

Moreover, to calculate significant differences, the *LSD* (Least significance difference) method is used. The obtained results reveal that there is not any significant difference between GA, ICA and HGICA. The results of this experiment are shown in Fig. 10.

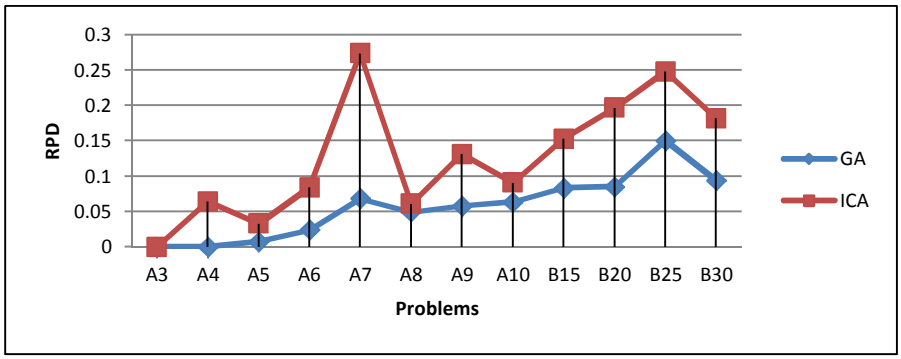

Fig. 7. Mean RPD plot for ICA and GA

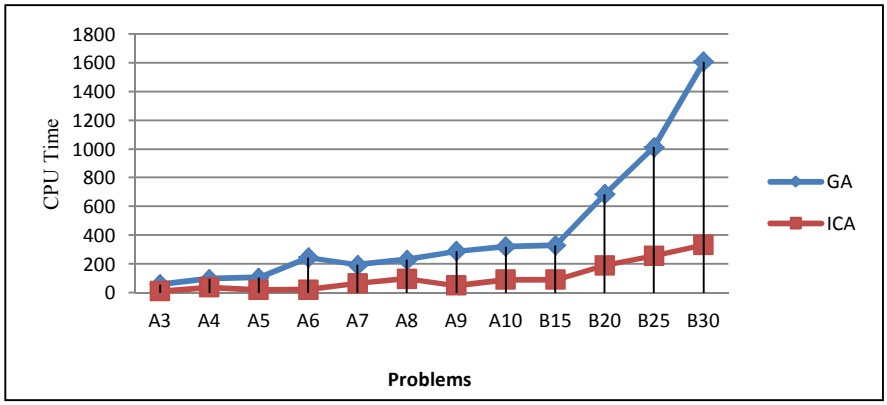

Fig. 8. CPU time of ICA and GA

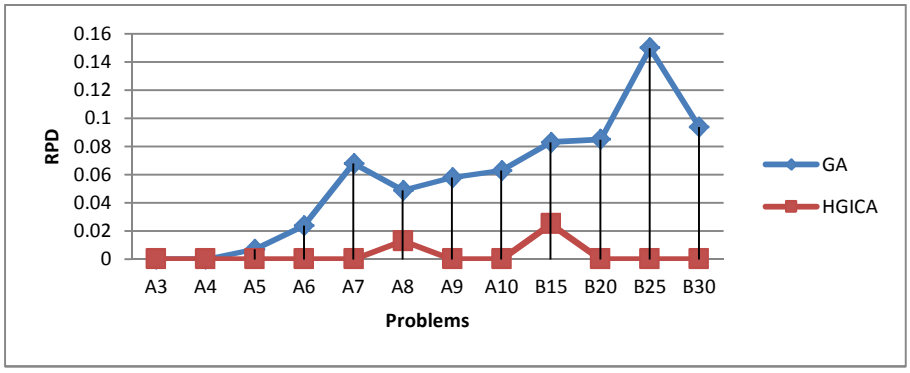

Fig. 9. Mean of RPD for GA and HGICA

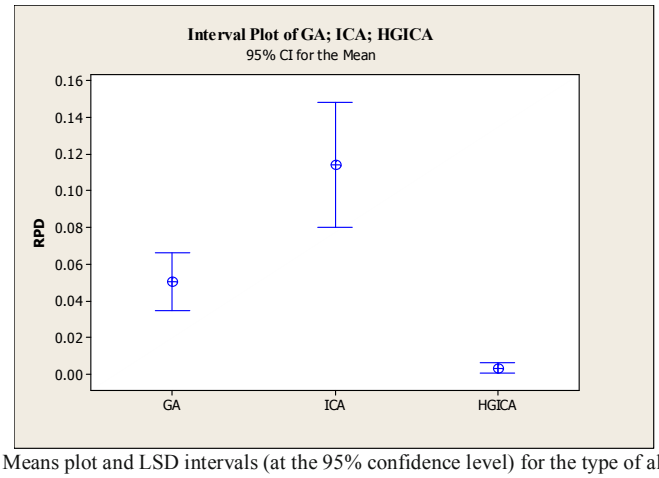

Fig. 10. Means plot and LSD intervals (at the 95% confidence level) for the type of algorithms

For testing the obtained results of proposed algorithms, the *t-test* is used as a method for statistical testing. The conducted hypothesis tests are as follows:

$$\boldsymbol{H_{0(1)} = the\ average\ error\ of\ GA <} \\ = the\ average\ error\ of\ ICA$$

$$\boldsymbol{H_{1(1)} = the\ average\ error\ of\ GA} \\ > the\ average\ error\ of\ ICA$$

In this statistical testing, statistic is $t_0 = \frac{X_{GA} - Y_{ICA}}{S_{p1}.\sqrt{\frac{1}{nX_{GA}} + \frac{1}{nY_{ICA}}}} = -2.31603$ Where ${S_{p1}}^2 = \frac{(nX_{GA}-1).{\delta_{GA}}^2 + (nY_{ICA}-1).{\delta_{ICA}}^2}{nX_{GA}+nY_{ICA}-1}$ and the range of acceptance is $(-\infty, t_{\alpha, nX_{GA}+nY_{ICA}-2}]$ .we cannot reject null hypothesis at 95% significance level because statistic is not greater than $t_{\alpha, nX_{GA}+nY_{ICA}-2}$.

$$\boldsymbol{H_{0(2)} = the\ average\ error\ of\ HGICA <} \\ = the\ average\ error\ of\ GA$$

$$\boldsymbol{H_{1(2)} = the\ average\ error\ of\ HGICA} \\ > the\ average\ error\ of\ GA$$

In this statistical testing, statistic is $t_0 = \frac{Z_{HGICA} - X_{GA}}{S_{p2}.\sqrt{\frac{1}{nX_{GA}} + \frac{1}{nZ_{HGICA}}}} = -4.71643$ where ${S_{p2}}^2 = \frac{(nX_{GA}-1).{\delta_{GA}}^2 + (nZ_{HGICA}-1).{\delta_{HGICA}}^2}{nX_{GA}+nZ_{HGICA}-1}$, and the range of acceptance is $(-\infty, t_{\alpha, nX_{GA}+nZ_{HGICA}-2}]$,because statistic is not greater than $t_{\alpha, nX_{GA}+nZ_{HGICA}-2}$,we cannot reject null hypothesis at 95% significance level.

These results of statistical testing imply that the average error of HGICA isn't statistically worse than other proposed algorithms.

Fig. 11. shows the *RPD* values for GA ,ICA and HGICA in different values of machines

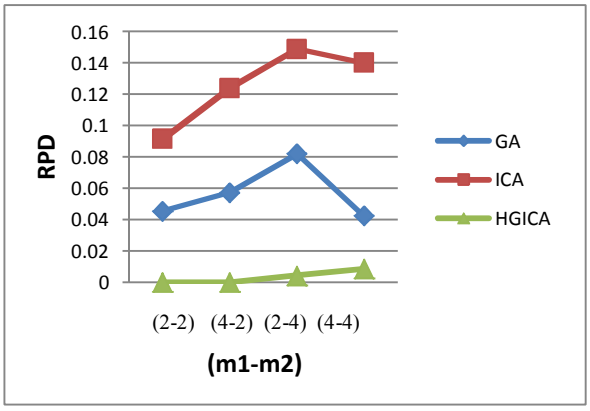

Fig. 11. RPD for different value of machine of problem

In this research a new mathematical model was presented for a hybrid flow shop scheduling problem with sequence dependent set up time while job preemption is allowed. Three meta-heuristics are proposed for solving the problem and many instances are applied to evaluate their performances. The parameters of the algorithms are calibrated by the RSM. The obtained results indicate that the hybrid algorithm produces best solutions based on RPD. For future research, it is worthwhile to develop the presented model by considering precedence constraint between jobs and also some other optimization algorithm may be investigated for the problem.

8. References

[1] Atashpaz-Gargari, E. and Lucas, C. (2007). Imperialist Competitive algorithm: an algorithm for optimization inspired by imperialist competition. Proceedings of the IEEE Congress on Evolutionary Computation, Singapore, 4661–4667.

[2] Allahverdi, A., Ng, C.T., Cheng, T.C.E. and Kovalyov, M.Y. (2008). A survey of scheduling problems with setup times or costs. European Journal of Operational Research, 187 (3), 985–1032.

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

[4] Arthanari, T.S. and Ramamurthy, K.G. (1971). An extension of two machines sequencing problem. Opsearch, 8, 10–22.

[5] Chou, F. (2013). Particle swarm optimization with cocktail decoding method for hybrid flow shop scheduling problems with multiprocessor tasks, International Journal of Production Economics, 141, 137–145.

[6] Engin, O., Ceran, G. and Yilmaz, M.K. (2011). An efficient genetic algorithm for hybrid flow shop scheduling with multiprocessor task problems. Applied Soft Computing, 11, 3056–3065.

[7] Johnson, S. M. (1954). Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly, 1, 61–68.

[8] Gen, M. and heng, R.C. (1997). Genetic algorithms and engineering design. John Wiley & Sons.

[9] Gupta, J. (1988). Two stages hybrid flow shop scheduling problem. Journal of Operation Research Soc, 39(4), 359-64.

[10] Holland, J.H. (1962). Outline for logical theory of adaptive systems, Journal of the association of computing machinery.

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

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

[13] Lahimer, A., Lopez, P. and Haouari, M., (2011). Climbing depth-bounded adjacent discrepancy search for solving hybrid flow shop scheduling problems with multiprocessor tasks. Lecture note on computer science, 6697, 117–130.

[14] Lin, S.W., Ying, K.C. and Huang, C.Y. (2013). Multiprocessor task scheduling in multistage hybrid flowshops: A hybrid artificial bee colony algorithm with bi-directional planning. Computers & Operations Research, 40, 1186–1195.

[15] Montgomery, D.C. (1991). Design and analysis of experiments. John Wiley & Sons.

[16] Naderi, B., Zandieh, Khaleghi-Ghoshe-Balagh, M.A. and Roshanaei, V. (2009). An improved simulated annealing for hybrid flow shops with sequence-dependent setup and transportation times to minimize total completion time and total Tardiness. Expert Systems with Applications, 36(6), 9625–9633.

[17] O˘guz, C., Ercan, M.F., Cheng, T.C.E. and Fung, Y.F. (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow shop. European Journal of Operational Research,149, 390–403.

[18] Rao, T.B.K. (1970). Sequencing in the order a, b, with multiplicity of machines for a single operation. Journal of the Operational Research Society of India, 7, 135–144.

[19] Ribas, I., Leisten, R. and Framin, 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.

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

[21] Shokrollahpour, E., Zandieh, M. and Dorri, B. (2010). A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. International Journal of Production Research, 1–17.

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

[23] Zandieh, M. and Rashidi, E. (2009). An effective hybrid genetic algorithm for hybrid flow shops with sequence dependent setup times and processor blocking. Journal of Industrial Engineering, 4, 51- 58.

[24] Wang, H.M., Chou, F.D. and Wu, F.C. (2011). A simulated annealing for hybrid flow shop scheduling with multiprocessor tasks to minimize makespan. International Journal of Advanced Manufacturing Technology, 53, 761–776.

[25] Ying, K.C. and Lin, S.W. (2009). Scheduling multistage hybrid flowshops with multi- processor tasks by an effective heuristic. International Journal of Production Research, 47, 3525–3538.