DIFFERENTIAL EVOLUTION ALGORITHM FOR FLEXIBLE JOB SHOP SCHEDULING PROBLEM

Bhaskara.P[1], Prof.G.Padmanabhan[2], B.Satheesh Kumar[3]
Sri Venkateswara University College of Engineering
India
E-mail:pbhaskar786@gmail.com,gpadmanabhan@svuniversity.ac.in

Abstract—This paper proposes differential evolution (DE) Algorithm for solving the Flexible job shop scheduling problem (FJSP) with the objective of minimization of makespan. Differential evolution algorithm is a latest population based evolutionary meta-heuristic, which was originally devised for solving continuous optimization problems, as stochastic real-parameter global optimizer. The DE employs simple mutation and cross- over to generate new candidate solutions, and applies one-to-one competition scheme to parsimoniously determine whether the new candidate or its present will survive in the next generation. The reckoning results and comparisons show that the proposed algorithm is very effective.

Keywords—Scheduling, Flexible Job Shop Scheduling, Differential Evolution Algorithm, Local Search, Makespan

I. Introduction

Scheduling problems have a vital role in recent years due to the growing consumer demand for variety, reduced product life cycles, changing markets with global competition and rapid development of new technologies. The Job Shop Scheduling Problem (JSSP) is one of the most popular scheduling models existing in practice, which is among the hardest combinatorial optimization problems [1]. Many approaches, such as, Simulated Annealing (SA) [2], Tabu Search (TS) [3], Genetic Algorithm (GA) [4], Ant Colony Optimization (ACO) [5], Neural Network (NN) [6], Evolutionary Algorithm (EA) [7] and other heuristic approach [8–10], have been successfully applied to JSSP.

In order to match today's market requirements, manufacturing systems need not only automated and flexible machines, but also flexible scheduling systems. The Flexible Job Shop Scheduling Problem extends JSSP by assuming that, for each given operation, it can be processed by any machine from a given set. Bruker and Schlie [11] were among the first to address this problem. The difficulties of FJSSP can be summarized as follows.

1) Assignment of an operation to an appropriate machine.
2) Sequencing the operations on each machine.
3) A job can visit a machine more than once (called recirculation).

These three features significantly increase the complexity of finding even approximately optimal solutions.

The job shop scheduling problem (JSSP) is one of the arduous combinatory optimization problems. Flexible job shop scheduling problem (FJSP) is a protraction of job shop scheduling problem that allows an operation to be processed by any machine from a given group, onward disparate routes. Scheduling optimization plays an important role in well designed and efficient manufacturing systems to fulfill global business needs. Intelligent utilization of resources to improve efficiency of the manufacturing systems is a convoluted combinatory job shop scheduling problem. Solving this kind of combinatory optimization problems with scholastic methods are almost impossible or takes considerable long time. Scheduling is defined in general as the process of Accrediting tasks to the available limited resources with goal of meeting the settled aspiration.

Scheduling in manufacturing industries is defined as the allocation of jobs to the available machines to reach the time based aspirations such as minimization of makespan, tardiness and due date etc. Scheduling is distressed with allocating limited resources to tasks to optimize certain objective functions.

The scholastic job shop scheduling problem can be stated as follows: A group of m machines and group of n jobs are given, each job contains of sequence of operations and a set of executed in a specified order. Each operation has to be performed on a given machine for a given time. In scheduling each operation sequence can be permuted independently [12]. The problem with n jobs and m machines can have a maximum of (n!)m different solutions. The completion of all operations of all jobs is known as makespan.

The objective is to find a feasible schedule with minimum makespan. Feasible schedules are obtained by permuting the processing order of operations on machines without violating the technological constraints.

The job shop scheduling problems are the most difficult problems as they are classified as non Deterministic polynomial (Np) hard type. The combinatory search space increases exponentially with increase in resources, and thus the generation consistently good.

II. Problem formulation

The Flexible job shop scheduling problem formulated as follows: There are a set of n number of independent jobs, j={j_1,$j_2$$j_3$,…,$j_n$} and a set of m machines m={m_1,m2,m3,…,m_n}. job is formed by a sequence of operations O, o={o_1,o_2,o_3,…,o_n} to be formed one after another according to the given sequence. Each operation i.e. jth operation of job ji, must be executed on one machine chosen from a given set of machines. The processing time of operation is machine dependent. The scheduling consists of two sub problems: The routing sub problem that assigns each operation to an appropriate machine and sequencing sub problem that determines a sequence of operations on all the machines. The objective is to find the perfect schedule that minimizes the makespan. The makespan means the time need to complete all the jobs, and can be denoted as F (C_{max}) [13].

The notations which are used to develop a mathematic model of the designing of Flexible job shop scheduling problem are defined as follows:

TABLE I. NOTATIONS FOR FJSP

notation	Representation
i	Part type index, i=1, 2, 3…n
j	Number of jobs. j=
m	Number of machines m =
O	sequence of operations O=
(C_{max})	Makespan(Maximum completion time)in minutes
($C_{i,j,k}$)	Partial makespan without predecessors.
($C_{i,j+1,k}$)	Enhanced makespan with predecessors.
d_{ij}	The duration(processing time) of operation i of job j.
$d_{i,j+1}$	The duration of operation i=1 of job j
$O_{i,k}$	Operation of job on corresponding machine.
$B_{i,k}$	Job processing on corresponding machine.
($f_{i,j,k}$)	Vector representing corresponding operation of job on specified machine.
R_{oij}	Most work remaining (MWKR) in minutes.
C_r	Correction Ratio.
P_r	Probability (Randomly taken).

III. Objective function

Minimize makespan F (C_{max})

Minimize F (C_{max}) = $C_{n,m}$

Conjunctive constraints;

$C_{i,j,k} \geq C_{i,j+1}-d_{i,j+1}$ for, i= 1, 2…n; j= 1, 2…p; k=1, 2…m.

$C_{i,j,k} \geq 0$ for, i= 1, 2…n; j= 1, 2…p; k=1, 2…m

Resource constraints

$O_{i,j,k}$ =1

If job j scheduled before the same job on specified machine=0 Otherwise

For, i= 1, 2…n;
j= 1, 2…p;
k= 1, 2…m.

The job shop scheduling represented by considering a three jobs and three machines problem (3X3 problem) as follows.

TABLE II. THE BASIC REPRESENTATION OF FJSP

Jobs	Operations-machine with processing time in min			
	Operations	*Machine1*	*Machine2*	*Machine3*
J_1	O_{11}	2	-	-
	O_{12}	-	3	-
	O_{13}	-	-	4
J_2	O_{21}	-	-	4
	O_{22}	-	3	-
	O_{23}	2	-	-
J_3	O_{31}	-	2	-
	O_{32}	-	-	2
	O_{33}	3	-	-

Representation of fixed id for the operations can be noted, and then the above problem can be written as follows;

TABLE III. REPRESENTATION OF FIXED ID FOR THE OPERATIONS

Jobs	**1**			**2**			**3**		
Operations	1	2	3	1	2	3	1	2	3
Machines	1	2	3	3	2	1	2	3	1
Work remaining	9	7	4	9	5	2	7	5	3
Processing time	2	3	4	4	3	2	2	2	3
Fixed ID	1	2	3	4	5	6	7	8	9

The selection process to calculate the makes span will be done by the following considerations;

a. Most work remaining (MWKR) .
b. SPT (shortest processing time), if MWRK is tie.
c. Random selection of operation if processing time is tie.

IV. Proposed algorithm

The Differential evolution (DE) algorithm proposed by Storn and Prince (1997) is one of the latest population based evolution meta-heuristic, which was originally devised for solving continuous optimization problems, as stochastic real-parameter global optimizer. The DE employs simple mutation and cross-over to generate new candidate solutions and applies one-to-one competition scheme to determine whether the new candidate or its present solution will survives in the next generation, since characterized by a novel mutation operator. The algorithm is very good at exploring the search space and locating the promising region [14].

A. *Description of Differential evolution algorithm*

a) Initialization

The initialization of the basic Differential evolution algorithm aims to set the control parameters and the initial population vector. The parameters include the size of population Np (Non deterministic polynomial). The mutation scale factor (F) and cross-over probability (C_r), It is obvious that the good set of parameters can be enhance the ability of the algorithm to search for the global optimization or near optimum region with a high convergence rate, as (DE) maintains a population on Np real valued. The dimensional vector denoted as X_{iG} where i denotes ith number of population and G denotes the generation to which the population belongs to.

$$X_{iG} = [\, X_{1,iG} + X_{2,iG} + X_{3,iG} + \ldots + X_{D,iG} \,] \quad (1)$$

Where; i=1, 2, 3… .

Which each parameter, taking values from user defined bounds.

$$X_{j,iG} \in [\, X_j^{low},\ X_j^{up} \,] \quad (2)$$

Where; j=1, 2, 3… D.

The size of the population doesn't change during the evaluation process, and it is one of the algorithm`s control parameters. Initially, the population is randomly created and may cover the entire parameter space with uniform probability [15]. Initialization process is followed by the process of evaluation, i.e. the cost of each vector is evaluated and stored for the feature reference.

b) Mutation

Mutation is an operation that adds a vector differential to the population vector for each target vector X_{iG}.

Where i=1, 2, 3…N_p.

The mutant vector is generated according to the relation;

$$V_{iG+1} = X_{aG} + F[\, X_{bG} - X_{cG} \,] \quad (3)$$

Where a, b, c are different integer numbers taken randomly for the mutation process over the range of population i.e.[1,N_p] and F is scaling factor(F > 0), which is a real constant in region[according to Storn and Price,1997].

Therefore, for each target vector X_{iG} DE generates a new mutant vector $V_{i,G+1}$ by adding the weighted difference between two (***randomly selected***) population vectors X_{bG} ,X_{cG} to a third X_{aG}, (***The best fitness vector***) with minimum makespan.

The mutant vector will be used as a donor vector for producing a trail vector, in the case where mutant vector is to shift outside the permitted interval(between the lower bound and upper bound shown in (2).

Then the parameters, can require by using the relation,

If $V_k < V_k^{low}$ then $V_k = V_k^{low}$ (4)
If $V_k > V_k^{up}$ then $V_k = V_k^{up}$
Where;
k=1, 2, 3… D.

c) Cross-over

The Cross-over represents as typical case of the information exchange between the individuals. In the basic DE, in order to increase diversity of the mutant vector, the operator of cross-over is applied on the population. This operator may partially suppress the effect of mutation by forming a trail vector (Generally the cross over probability lies in between 0 to 1).

Then the generation of trail vector is formulated as the following equation;

$U_{i,G+1}$ =[$U_{1,iG+1} + U_{2,iG+1} +…+ U_{D,iG+1}$] (5)

The components of trail vector are taken either from the mutant vector $V_{i,G+1}$ or from the original target vector X_{iG} according to the relation;

$V_{k,iG+1}$ If random number P_r
$U_{k,iG+1}$= (6)
$X_{k,iG}$ Otherwise

Where randomization is function that returns a uniform floating point number in range [0,1]P_c is cross-over probability (user defined parameter) and initially generated random number is a function which returns a random integer number in the range. This function ensures at least one parameter will be taken from the mutant vector.

d) Selection

Selection is the process of producing the offspring to the new generation 'G+1'.Unlike many other EA`s (Evolutionary algorithms) DE doesn't use ranking or propositional selection. Instead it decides whether the target or the trail vector is allowed to advance in the next generation by comparing their values. If the values of the target vector are lower than that of trail vector, then the target vector survives. It reproduces its structure to the next generation.

More finally, their greedy mechanism can be written as follows:

if cost (x_{iG} ≤ cost $Ui,_{G+1}$)
X_{iG+1}= (7)
$U_{i,G+1}$ Otherwise

V. Overview of Differential evolution algorithm

The basic steps and working principle of Differential evolution algorithm for a flexible job shop scheduling problem is as follows;

Step 1: Specify the population size (N_p), scaling factor (F), cross-over probability (C_r),
and maximum number of generations (G_{max}).
Step 2: Set initial population G=0
Step 3: Evaluate each sample population and specify
The base vector and denote it a X_{aG} or X_{bestG}.
Step 4: Mutation phase. Generate Noise vector, by using mutation operator (3).
Step 5 Cross- over operation. Generate N_p trail vectors using (6).
Step 6: For each trail or sample vector evaluate the Makespan and compare with makespan of Base vector X_{bestG}.
Step 7: Selection process. Evaluating the Target vector $X_{i,G+1}$ by the individual selection operator described in (7). Label it as next generation. Make G = G+1.
Step 8: Updating of X_{bestG}.
Step 9: If G <G_{max} then go to step 4. Otherwise stop the procedure.

VI. Implementation of Differential Evolution Algorithm

TABLE IV. IMPLEMENTATION OF DIFFERENTIAL EVOLUTION ALGORITHM FOR FJSP

	Sample vector		Noise vector				Target vector	
chr.	***Population***	***C max***	***Intermediate Population***	***C max***	***Cr***	***Pr***	***New population***	***Cmax***
1	123456789	17	318597426	11	0.5	0.45	318597426	11
2	142756839							
3	784159263							
4	451267839							
5	124536789							

In Differential evolution algorithm each sample pupation representation a chromosome and each chromosome represents a solution for the sequence. Each sequence is having a set of operation for jobs on different machines.

Then the process will continued to calculate a noise vector form two different randomly selected vector. Trail one we have taken chromosome 2and chromosome 3 and chromosome 1is considered as the base vector. The calculation of makespan for the each chromosome is required, then the calculation of makespan for the chromosome done for the 3X3 problem shown in table III.

a) Calculation of makespan for base vector

Base vector→	1	2	3	4	5	6	7	8	9
(J, O, M) →	(111)	(122)	(133)	(213)	(222)	(231)	(312)	(323)	(331)
R_{oij}	9	7	4	9	5	2	7	5	3
P.T	2	3	4	4	3	2	2	2	3

The Gantt chart is used to calculate the makespan by considering the assigning machines based on the most work reaming and shortest processing time. The makespan of the chromosome calculated and the time taken to complete all the operations for the entire job is 17 minutes.

b) Calculation of noise vector

Base vector→1 2 3 4 5 6 7 8 9

Let us choose two random vector chromosomes 2&3,

```
Chromosome 2 → 1  4  2  7  5  6  8  3  9

Chromosome 2 → 7  8  4  1  5  9  2  6  3
         ( - )_______________________________
               6  4  2  6  0  3  6  3  6   X 0.45
               ______________________________
               2  8  9  1  7  1  6  3  6
 Base vector→  1  2  3  4  5  6  7  8  9
         ( +)________________________________
               3  10  12 5  12  7  13  11 15
               ______________________________
     Converting the above values to `9', we get
               3  1  3  5  3  7  4  2  6
```

Here it is observed that vector doesn't containing all the operations and some operations are repeated, hence we replace the non existing operation to avoid repetition. Then the noise vector will be,

Noise vector → 3 1 8 5 9 7 4 2 6.

The makespan for the noise vector is calculated by using Gantt chart and obtained as 11 minutes.

c) Calculation of Target vector

Consider,

P_r = Random number (probability generated randomly).

C. R= Correction ratio ($0<C_r<1$).

Generally the correction ratio is defined or fixed by the user.

Then the condition for selection the target vector is as follows;

If, $P_r< C_r$. Then, Noise vector =Target vector.

If, $P_r > C_r$. Then, best of Noise vector = Target vector.

For the base vector, $P_r = 0.45$ and $C_r = 0.5$, here $P_r<C_r$ then, the noise vector is considered as Target vector. This procedure is to repeat for all the population individually to calculate make span.

VII. Experimental setup

The proposed DE algorithm was implemented in MATLAB (version 8.5 on windows 64 bit operation system) on an Intel 2.83 GHz Xeon processor with 4 GB of RAM and 2 GB GPU, To evaluate the performance of DE algorithms. The following four sets of standard benchmark in-stances in the FJSP literature are considered [14].

(1) Kacem data: The data set consists of five instances from Kacem et al. (2002b) with number of jobs ranging from 4 to15, number of machine ranging from 5 to 10, number of operations for each job ranging from 2 to 4, and number of operations for all jobs ranges from 12 to 56.

(2) BRdata: The data set consists of 10 instances from Brandim- arte (1993), which were generated randomly generated using uniform distribution between given limits. The number of jobs ranges from 10 to 20, number of machines ranges from 4 to 15, number of operations for each job ranging from 5 to 15, and number of operations for all jobs ranges from 55 to 240.

(3) BCdata: The data set consists of 21 instances from Barnes and Chambers (1996), which were obtained from three ever challenging classical JSP instances (mt10, la24, la40) (Fisher and Thompson, 1963; Lawrence, 1984). The number of jobs ranges from 10 to 15, number of machines ranges from 11 to 18, number of operations for each job ranging from 10 to 15, and number of operations for all jobs ranges from 100 to 225.

The proposed algorithms run 100 independent times for each in-stance from Kacem data, BRdata, and BCdata, and only run 10 independent times for each instance from HUdata due to the large number of instances in this data set. The results will involve four metrics including the best makespan (Best), the average makespan (AVG), the standard deviation of makespan (SD), and the average computational time in seconds (CPU_{av}) obtained by the related algorithms.

To show the superiority of our DE algorithm, we compare our computational results with competitive algorithms HGA [16] & Discrepancy algorithm [17] in the literature for each data set.

VIII. Conceptual results

a) Results of Kacem instances

The first data set under study is Kacem data. We compare ur DE with two recently proposed algorithms including HGA & Discrepancy algorithm. The detail results are listed in Table 5. The first column symbolizes the name for each instance; the second column shows the size of the instance, in which n stands for the number of jobs and m represents the number of machines; the third column lists the best known solution (BKS) ever reported in the literature for each instance; the remaining columns describe the computational results HGA & Discrepancy algorithm. The results from the table 5, It was observed that the conceptual results for HGA & Discrepancy algorithm were quite equal to Differential evolution algorithm.

b) Results of BRdata instances

The first data set under study is Kacem data. We compare our DE with two recently proposed algorithms including HGA & Discrepancy algorithm. The detail results are listed in Table 5. The first column symbolizes the name for each instance; the second column shows the size of the instance, in which n stands for the number of jobs and m represents the number of machines; the third column lists the best known solution (BKS) ever reported in the literature for each instance; the remaining columns describe the computational results HGA & Discrepancy algorithm. The results from the table 5, it was observed that the conceptual results for HGA & Discrepancy algorithm. The makespan for the instances are comparative and the results obtained by DE algorithm were shown to be effective and efficient.

c) Results of BCdata instances

The BCdata is one of the largest data set for the FJSP in the literature. Recent important work on this data set can be referred the total 21 instances were shown in the table 7 and the results obtained by DE algorithm were compared along with average makespan for all the population. The comparison of makespan of HGA & Discrepancy algorithm to DE algorithm was done and it is observed that the conceptual results for BC data instances were efficient and accurate.

d) Effect of hybridizing DE and local search algorithms

To investigate the effectiveness of DE algorithm based global search and local search algorithms, the experiments and comparisons are carried out between the DE algorithm with Hybrid Genetic and Discrepancy algorithms.

TABLE V. RESULTS OF KACEM INSTANCES.

Instance	n x m (jobs/machines)	BKS	HGA (Jie Gao 2008)	Discrepancy search(Abir Ben 2010)	Proposed DE algorithm			
			C_{max}	C_{max}	Population size	Best	AVG	CPU_{av}
Case 1	4 x 5	11	11	11	100	11	11.05	0.15
Case 2	8 x 8	14	14	14	100	14	14.00	0.21
Case 3	10 x 7	11	11	11	100	11	11.02	0.29
Case 4	10 x 10	7	7	7	100	7	7.00	0.36
Case 5	15 x 15	11	11	11	100	11	11.05	0.41

TABLE VI. RESULTS OF BRDATA INSTANCES.

Instance	n x m (jobs/machines)	BKS	HGA (Jie Gao 2008)	Discrepancy search(Abir Ben 2010)	Proposed DE algorithm			
			Cmax	Cmax	Population size	Best	AVG	CPUav
MK01	10 x 6	40	40	40	100	40	40.00	1.78
MK02	10 x 6	26	26	26	100	26	26.01	2.84
MK03	15 x 8	204	204	204	100	204	203.97	11.47
MK04	15 x 8	60	60	60	100	60	60.02	5.07
MK05	15 x 4	172	173	175	100	173	172.91	23.21
MK06	10 x 15	58	58	60	100	61	61.20	7.89
MK07	20 x 5	139	144	139	100	141	141.01	13.21
MK08	20 x 10	523	523	523	100	521	520.97	31.25
MK09	20 x 10	307	307	307	100	308	308.25	29.12
MK10	20 x 15	197	198	198	100	197	197.02	24.55

Table VII. RESULTS OF BCDATA INSTANCES.

Instance	n x m (jobs/machines)	BKS	HGA (Jie Gao 2008)	Discrepancy search(Abir Ben 2010)	Proposed DE algorithm			
			Cmax	Cmax	Population size	Best	AVG	CPUav
Mt10x	10 x 11	918	927	918	100	918	918.02	31.43
Mt10xx	10 x 12	918	910	918	100	918	918.06	31.70
Mt10xxx	10 x 13	918	918	918	100	918	918.23	35.43
Mt10xy	10 x 12	905	918	906	100	905	905.21	34.51
Mt10xyz	10 x 13	847	918	849	100	847	847.49	35.79
Mt10c1	10 x 11	927	905	928	100	928	928.21	31.07
Mt10cc	10 x 12	908	849	910	100	906	905.98	34.00
Setb4x	15 x 11	925	914	925	100	925	925.42	34.04
Setb4xx	15 x 12	925	914	925	100	925	925.02	31.76
Setb4xxx	15 x 13	925	925	925	100	924	924.05	31.89
Setb4xy	15 x 12	910	925	916	100	910	910.07	31.13

Setb3xyz	15 x 13	903	925	905	100	905	905.21	30.39
Setb4c9	15 x 11	914	916	919	100	914	914.01	32.19
Setb4cc	15 x 12	907	905	909	100	909	909.21	32.00
Seti5x	15 x 16	1198	1175	1201	100	1204	1204.02	78.20
Seti5xx	15 x 17	1197	1138	1199	100	1202	1202.08	76.52
Seti5xxx	15 x 18	1197	1204	1197	100	1202	1202.10	75.07
Seti5xy	15 x 17	1136	1202	1136	100	1138	1138.23	79.98
Seti5xyz	15 x 18	1125	1204	1125	100	1130	1130.41	81.85
Sei5c12	15 x 16	1174	1136	1174	100	1175	1175.56	65.06
Seti5cc	15 x 17	1136	1126	1136	100	1137	1137.21	74.83

IX. Conclusions

This paper can be summarized, that the Differential evolution (DE) algorithm is the effective tool to solve the Flexible job shop scheduling problem. The performance of DE was largely improved by means of local search and some available knowledge was learned from the optimization of FJSP, at the same time, the existing knowledge was applied to guide the current heuristic searching of DE. Final experimental results indicate that the proposed DE algorithm outperforms some published methods in the quality of schedules. The efficiency of the DE algorithm also helps to solve the extension to multi-objective FJSSP will be investigated in the near future.

References

[1] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flow shop and job-shop scheduling, Mathematics of Operations Research 1 (2) (1996) 117–129.

[2] M. Kolonko, Some new results on Simulated Annealing applied to the Job Shop Scheduling Problem, European Journal of Operational Research 113 (1) (1999) 123–136.

[3] F. Pezzella, E. Merelli, A Tabu Search method guided by shifting bottleneck for the Job Shop Scheduling Problem, European Journal of Operational Research 120 (2) (2000) 297–310.

[4] J.F. Goncalves, et al., A hybrid genetic algorithm for the Job Shop Scheduling Problem, European Journal of Operational Research 167 (1) (2005) 77–95.

[5] K.L. Huang, C.J. Liao, Ant colony optimization combined with taboo search for the job shop scheduling problem, Computers and Operations Research 35 (4) (2008) 1030–1046.

[6] D.J. Fonseca, D. Navaresse, Artificial neural networks for job shop simulation, Advanced Engineering Informatics 16 (4) (2002) 241–246.

[7] I.T. Tanev, T. Uozumi, Y. Morotome, Hybrid evolutionary algorithm-based real-world Flexible Job Shop Scheduling Problem: application service provider approach, Applied Soft Computing 5 (1) (2004) 87–100.

[8] H. Chen, P.B. Luh, An alternative framework to Lagrangian relaxation approach for Job Shop Scheduling, European Journal of Operational Research 149 (3) (2003) 499–512.

[9] W.Q. Huanh, A.H. Yin, An improved shifting bottleneck procedure for the Job Shop Scheduling Problem, Computers & Operations Research 31 (12) (2004) 2093–2110.

[10] K. Jansen, M. Mastrolilli, R. Solis-Oba, Approximation schemes for Job Shop Scheduling Problems with controllable processing times, European Journal of Operational Research 167 (2) (2005) 297–319. [11]P. Bruker, R. Schlie, Job-shop scheduling with multi-purpose machines, Com- puting 45 (4) (1990) 369–375.

[12] Arit Thammano,Ajchara phu-ang, A

hybrid artificial bee colony algorithm with local search for Flexible job shop scheduling problem (2013).

[13]Rui Zhang, A Differential evolution algorithm for Job shop scheduling problems involving due date determination decision.

[14]Yuan Yuan,Hua Xu, Flexible job shop scheduling problem using hybrid differential evolution algorithms (2013).

[15]Riza A.Rahman,Budi Santosa, Hybrid Differential evolution and bottle neck Heuristic algorithm to solve Bi-objective Hybrid flow shop scheduling unrelated parallel machines problems (2014).

[16] Gao, J., Sun, L., & Gen, M. (2008). A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers & Operations Research, 35(9), 2892–2907.

[17] Ben Hmida, A., Haouari, M., Huguet, M., & Lopez, P. (2010). Discrepancy search for the flexible job shop scheduling problem. Computers & Operations Research, 37(12), 2192–2201.