ISSN 1392–1207. MECHANIKA. 2017 Volume 23(4): 594–603

An effective genetic algorithm for flow shop scheduling problems to minimize makespan

R.B. Jeen Robert*, R. Rajkumar**
**Department of Mechanical Engineering, AAA College of Engineering and Technology, Sivakasi- 626005, India, E-mail: jeenrb_robert@yahoo.com*
***Department of Mechanical Engineering, Mepco Schlenk Engineering College, Sivakasi-626005, India*

crossref http://dx.doi.org/10.5755/j01.mech.23.4.15053

1. Introduction

In past 5 decades flow shop scheduling may be a challenging area for researchers. Its main aim is to work out the job sequence of processing jobs on a given set of machines. In manufacturing environment flow shop scheduling is taken into account as most category of problem. A general flow shop scheduling during which n jobs are to be processed through m machines is to be considered. Fixed and non-negative processing times are thought of here. the foremost important assumptions created are that every job will be processed on only one machine at a time, the operations don't seem to be preventive, the jobs are out there for processing at time zero and set-up times are sequence independent. Here we have a tendency to think about the permutation flow shop problem, identical job sequence is taken into account on each machine for manufacturing. In flow shop scheduling minimizing makespan and total flow time may be a difficult task for several of the researchers. Therefore we thought of minimizing makespan as objective for my present work using meta-heuristic approach by improvising the Genetic algorithmic program. During this effective genetic algorithm (EGA), worst solutions are aloof from that algorithm by adding robust factor concept.

The first research involved to the flow shop scheduling problem was planned by Johnson [1]. Johnson represented a certain algorithm to reduce makespan for the n-jobs and 2-machines flow shop scheduling problem. Once the flow shop scheduling problem enlarges as well as additional jobs and machines, it becomes a combinatorial optimization problem. It's clear that combinatorial optimization problems are in NP-hard problem class, and close to optimum solution techniques are most popular for such problems. In recent years, metaheuristic approaches like Tabu Search, Genetic algorithms, simulated annealing, differential evolution, and artificial immune systems are very fascinating to solve combinatorial optimization problems relating to their computational performance. The recent studies for the flow shop scheduling problem with makespan criteria, it's obvious that the solution methods supported metaheuristic approaches are often planned. Mainstream of studies for the flow shop scheduling problem focuses to reduce makespan. For instance, the flow time, the machine idle times are main measures in minimizing total scheduling cost. Whereas makespan decrease results in total production run utilization, flow time decrease results in stable consumption of resources, fast turn-around of jobs and work-in-process inventory minimization. So as to reduce the production cost, it's desired to attain each these two objectives at the same time. Rajendran [2, 3] given one branch-and-bound algorithm and two heuristic algorithms aimed at two machine flow shop scheduling problem through makespan because the primary criterion and total flow time. Neppalli, Chen, and Gupta [4] planned two genetic algorithms for this problem. T'kindt, Gupta, and Billaut [5] presented mathematical programming designs, a branch-and-bound process, and a heuristic algorithm. Later, Jeen Robert et al. [6] have given a hybrid algorithm for Minimizing Makespan in the flow shop Environment. Yeh [7] created a memetic algorithm to solve this problem.

Recently, ant colony optimization (ACO) approach has become additional preferred to solve combinatorial optimization problems. This heuristic algorithm combines simulated annealing search and a local search algorithm. This study is that the initial application of ACO metaheuristic to multiobjective m-machine flow shop scheduling problem with esteem to the both objectives of makespan and total flow time. The performance of planned algorithm was related with two heuristic algorithms developed by Rajendran [8] and Ravindran [9] for these category problems and Genetic algorithm. Computational studies were showed on the yardstick problems from Taillard [10] because the test problem so as to verify the algorithm's performance. During this literature survey we tend to found several algorithm are accustomed solve flow shop scheduling problem in manufacturing field by many of the researchers, however we might found that genetic algorithm is very old algorithm however very powerful algorithm to solve flowshop scheduling problem. At the same time Muthiah [18] was described an opposite genetic algorithm and it was applied in Job shop scheduling problem. Thus we used genetic algorithm to solve flowshop scheduling however it's in effective approach by modifying its method routes strategies. Here we tend to addressed effective genetic algorithm (EGA) to solve flowshop benchmark problem, and also the results obtained by EGA are compared with earlier reportable results by using ant colony algorithm ACO by Betul Yagmahan [11] and Andrea Rossi [12].

2. Problem description and notations

Consider an m-machine flow shop problem where there are n jobs to be processed on the m machines among an equivalent order. We've got a tendency to completely take under attention of the permutation schedules, i.e., the same job order on each machine. The objective of this paper is to improve a developed Real Coded Genetic Algorithm and Efficient Genetic Algorithm to hunt out the most effective or near best sequence in flowshop environment by minimizing makespan. Meanwhile, the sequence among which

each machine processes all jobs is identical on all machines.

Makespan is one among the foremost necessary criteria in every production systems; it's up to the whole completion time of all the activities. Minimizing this criterion caused higher usage of the resources specially machinery and manpower. In each simple and hybrid flow shop, the method is to transform the flow shop into a network form, then a linear programming model with the objective of minimizing the entire completion time of all the actions are constructed. Minimizing total completion time of all the activities is similar to minimizing makespan among the production system. The result is that the sequencing and scheduling of all the activities are determined. Besides, the following assumptions are thought of throughout this study:

- Each machine can process at the foremost one job at a time and each job is processed on just one machine at any given time.
- The schedule is non-preemptive; which suggests once a job starts to be processed on a machine, the process cannot be interrupted before completion.
- The number of jobs and their process times on each machine are known ahead and are non-negative integers.
- The individual operation setup times are little compared with their processing times, and are enclosed within the processing times.
- The prepared time of all jobs is zero, which suggests that everyone jobs are available for processing at the start.

The handling time $t(i,j)$ is specified for any job i on any machine j. Given a permutation of the jobs, the accomplishment times of jobs on the machines, makespan are calculated as follows:

$$C(J_1,1) = t(J_1,1), \tag{1}$$

$$C(J_i,1) = C(J_{i-1},1) + t(J_i,1), i = 2,\cdots\cdots\cdots, n, \tag{2}$$

$$C(J_1,1) = C(J_1, J-1) + t(J_1, j),\ j = 2,\cdots\cdots\cdots, m, \tag{3}$$

$$C(J_1,1) = max\left\{C(J_{i-j}, j), C(J_i, j-1)\right\} + t(J_i, j),$$
$$i = 2,\cdots\cdots\cdots, n;\ j = 2,\cdots\cdots\cdots, m. \tag{4}$$

(i) Makespan C_{max} is the length of time required to finish processing all jobs, i.e.:

$$C_{max} = max\{C_1, C_2, \cdots\cdots, C_n\}, \tag{5}$$

where C_n is the completion time of job n.

The problem is to seek out a permutation of jobs thus on minimize the makespan of jobs. Since flow-shop scheduling problem has been shown to be NP−complete problem, for practical purposes, it has always a lot of acceptable to use an approximation technique that generates a close to best solution effectively. Throughout this paper an effort is made to boost the prevailing genetic algorithm procedures to use to permutation flowshop scheduling problems to provide higher results. The subsequent notations are utilized during this paper:

n	number of jobs
m	number of machines
p_{ij}	processing time of job i on machine j
p_m	probability of mutation
P_{to}	probability for Roulette wheel selection
p_{sc}	probability for sub chromosomal crossover
P_{sm}	probability for sub chromosomal mutation
$P(k)$	population at *kth* generation
X* , X**	best and second best chromosomes
R	real random number is 0 and 1
Ng	maximum generation

3. Simple genetic algorithms (SGA)

Genetic Algorithms may be a world class optimization algorithm that look for improved performance by sampling areas of the solution space that have high probability of resulting in a good solution. They reproduce the natural evolutionary process in this, in every generation, the fittest individuals have a superior chance to produce offsprings by collaborating features of the parents or by altering one or a lot of the parent's characteristics whereas, the worst individuals are possibly to die. Decisions that need to be created for applying GA include individual or chromosome illustration, technique of crossover, probability of crossover, technique of mutation, probability of mutation, and population size. GA is of course parallel and exhibits implicit parallelism, that doesn't evaluate and improve a single answer, however analyses and modifies a set of solutions at the same time (Goldberg, [15]). The potential of a GA to control on many solutions instantaneously and collect information from all current solutions to direct search reduces the possibility of being trapped in a very local optimum.

In general, simple Genetic algorithm (SGA) consists of the subsequent steps:

Step 1: Initialize a population of chromosomes.
Step 2: Evaluate the fitness operate of each body.
Step 3: create new chromosomes by applying genetic operators like crossover and mutation to current chromosomes.
Step 4: evaluate the fitness worth of the new population of chromosomes.
Step 5: If the termination condition is satisfied, stop and come the simplest chromosome; otherwise, go to Step 3.

4. Real coded genetic algorithms (RCGA)

To compensate for the excessive computation time needed by the SGA, the important genetic algorithm (IGA) emphasizing on the coding of the chromosomes with floating purpose illustration was introduced and established to possess important improvements on the computation speed and exactness (Wei-Der Chang [14]; Goldberg [15]). Real Coded Genetic algorithm (RGGA) is same as that of straightforward genetic algorithm provided instead of taking binary values it'll take real values as [0, 1].

In real coded genetic algorithm the subsequent steps are to be followed:

Step 1: randomly generate N chromosomes on initial population within the search.
Step 2: Calculate the fitness for every chromosome.
Step 3: Perform reproduction, i.e. choose the higher chromosomes with probabilities supported their fitness values.
Step 4: Perform crossover on chromosomes selected in higher than step by crossover probability.
Step 5: Perform mutation on chromosomes generated in higher than step by mutation probability.

Step 6: If reach the stop condition or get the best solution, one could stop the process, else repeat Steps 2-6 until the stop condition is achieved.
Step 7: Get the best solution.4.

5. Effective genetic algorithm (EGA)

To enhance the performance of straightforward genetic algorithm, an efficient genetic algorithm has been developed to induce higher results. It's just a modification of straightforward genetic algorithm. GA is impressed by Darwin's Theory concerning evolution - "survival of the fittest". GA represents an intelligent exploitation of a random search wont to solve optimization problems. Within the simple GA-based approach, the varied stages like evaluation, selection, crossover and mutation are repeatedly executed when initialization till a stopping criterion is met. The algorithm works on multiple solutions at the same time. In this proposed EGA tool the initial population is processed by Nawaz, Enscore, and Ham (NEH) heuristic technique. During this a general purpose schedule optimizer for manufacturing flow shops planning using genetic algorithms. A performance measures for planning is make-span C_{max} that has been used most optimum utilization of resources to extend productivity and stated as maximum completion time of last job to exit from the system:

$$C_{max} = max\{C_1, C_2, \ldots\ldots\ldots, C_n\},$$

where C_{max} is the makespan that needs to be decreased. Effective Genetic algorithm consists of the subsequent steps:
Step 1: Generate Initial Population using NEH principle.
Step 2: evaluate fitness function value of chromosomes.
Step 3: selection procedure is done by roulette wheel technique.
Step 4: Use sub chromosomal level crossover.
Step 5: Use sub chromosomal level mutation (inverse technique, single purpose mutation).
Step 6: Robust- Replace heuristic technique applied.
Step 7: chromosomal level mutation.

The elaborated flow chart of EGA is given in Fig. 1. The planned algorithm can work with these steps that are within the flow chart. The subsequent parameters were described before the program works:

Initial Population: 1000.
Number of generation: 250.
Crossover probability: 0.9.
Mutation probability: 0.05.

5.1. Generation of initial population

A set of initial population are indiscriminately generated in step with the problem size and one initial seed is incorporated by NEH heuristic (Nawaz et al. [16]) for the initial population.

5.2. Evaluate fitness function

In order to mimic the natural process of the survival of the fittest, the fitness analysis function assigns to every member of the population a value reflective their relative superiority (or inferiority). Every chromosome has an evaluation criterion supported the objective function. Since IGA is used for maximization problems, a minimization problem may be appropriately converted into a maximization problem using a fitness function. The fitness function is:

$$f(x) = 1/C_{max} \tag{6}$$

5.3. Selection procedure

In EGA only classical roulette wheel selection technique is incorporated for improving results (Goldberg [15]) is taken into account. In roulette wheel selection, parents are selected according to their fitness value. The higher the fitness the additional chances to be selected. The subsequent procedure is used for selection.

5.4. Sub chromosomal level crossover

In this step the initial chromosomes are reformed by subjecting sub chromosomal crossover. The whole chromosome string length is splited into sub chromosomes of equal length and these sub chromosomes are moved indiscriminately within the string to create a new chromosome.

Then the makespan values of corresponding new chromosome (child) are calculated, and it's compared with parent chromosome. Finally that chromosome is producing less makespan value that chromosome are maintained.

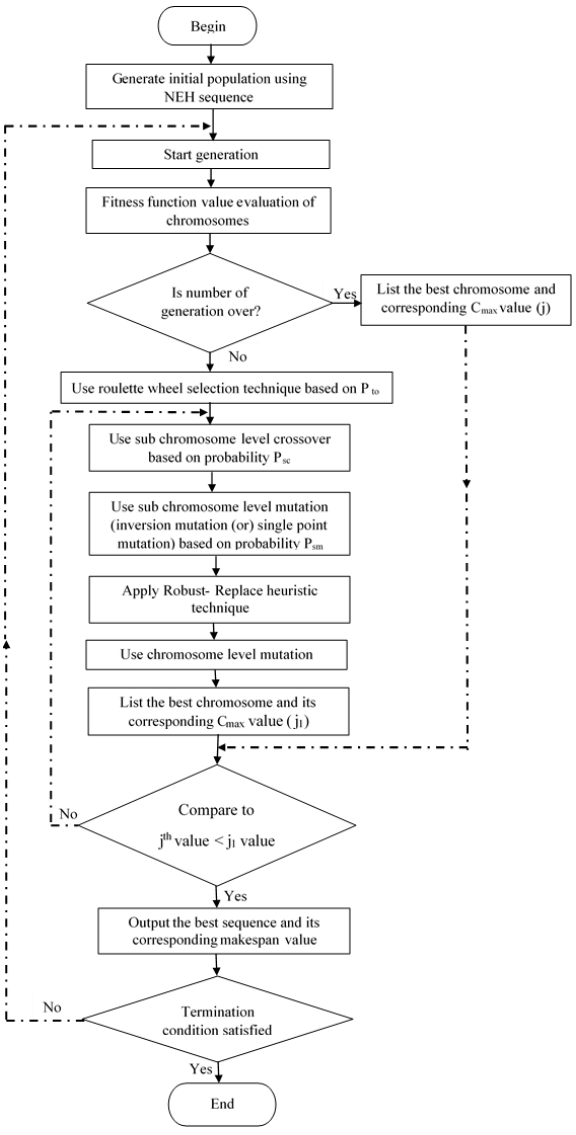

Fig. 1 Flow chart of EGA

Parent chromosome

1	2	3	4	5	6	7	8	9	10
1			2			3			4

Sub-chromosome length =3

After sub-chromosomal crossover (child)

7	8	9	10	1	2	3	4	5	6
3			4			1			2

5.5. Sub-chromosomal level mutation

Mutation generates an offspring solution by altering the parent's feature indiscriminately. The most effective chromosome obtained from sub-chromosomal crossover is subjected for mutation operation. During this work, two differing types of mutation operators are used specifically inverse mutation and single point mutation.

5.5.1. Inverse mutation

In this step, indiscriminately choose two positions of i and j from a sequence. The portion of the sequence between these two positions is reversed to get a brand new mutated sequence. The new sequence represents the sequence of operations after mutation. Then the makespan values of corresponding new sequence are calculated, and it's compared with old sequence. Finally that sequence is producing less makespan value that sequence is sustained.

Original Sequence

10	8	9	7	6	4	5	3	2	1

Mutated Sequence

10	**2**	**4**	**3**	**8**	**7**	**9**	**6**	**5**	1

Mutation between positions **2** and **9**.

5.5.2. Single point mutation

In this mutation technique, a random operation is chosen within the sequence and moved random position within the sequence. Then the makespan values of the resulting sequence are calculated and its value is compared with previous sequence. At last, the sequence which produces lees makespan that might be continued for following step.

Before mutation

10	8	9	7	**6**	4	5	3	2	1

After mutation

10	8	9	7	4	5	3	2	1	**6**

5.6. Robust- replace heuristic

In this step, indiscriminately generate ten sequences for implementation supported mutation and their corresponding makespan value and robust factors are calculated as shown in Table 1. Value of robust factor is calculated using the formula Robust Factor:

$$R = 1/C_{max} \ . \tag{7}$$

Then calculate the average value of the robust factor for the ten set of sequences. The sequences having very less robust factor compared with the typical robust value are selected for making new schedule. The maintained schedules are given in Table 2. For better understanding of this process, here we have a tendency to thought of benchmark problem (IPTA001) proposed by Taillard [10] is chosen. The problem size is 20 jobs and 5 machines

Table 1

Initial sequences

Sequence																				Makespan	Robust Factor
3	17	9	8	15	14	11	16	13	19	6	4	5	18	1	2	10	7	20	12	1286	0.000778
3	17	19	8	15	14	11	16	13	9	6	4	5	18	1	2	10	7	20	12	1322	0.000756
3	13	4	14	5	15	6	16	7	17	8	18	9	19	20	2	1	10	11	12	1519	0.000658
3	17	9	8	1	5	15	14	11	16	13	19	6	4	12	2	10	7	20	18	1377	0.000726
3	12	17	20	9	7	8	10	15	2	14	1	11	18	16	5	13	4	19	6	1485	0.000673
17	9	8	3	15	14	11	16	13	19	20	12	10	7	2	1	18	6	5	4	1497	0.000668
6	15	3	9	7	11	1	2	17	13	4	8	19	16	5	18	12	14	10	20	1311	0.000763
6	17	7	8	9	1	11	12	15	13	4	3	19	16	5	18	2	14	10	20	1377	0.000726
17	6	3	9	8	15	14	11	16	13	19	4	5	18	1	2	10	7	20	12	1323	0.000756
6	17	3	9	8	15	14	11	16	13	19	4	5	18	1	2	10	7	20	12	1345	0.000743
Average Robust factor= 0.000725																					

Table 2

Sequences retained

Sequence																				Makespan	Robust Factor
3	17	9	8	15	14	11	16	13	19	6	4	5	18	1	2	10	7	20	12	1286	0.000778
3	17	19	8	15	14	11	16	13	9	6	4	5	18	1	2	10	7	20	12	1322	0.000756
3	17	9	8	1	5	15	14	11	16	13	19	6	4	12	2	10	7	20	18	1377	0.000726
6	15	3	9	7	11	1	2	17	13	4	8	19	16	5	18	12	14	10	20	1311	0.000763
6	17	7	8	9	1	11	12	15	13	4	3	19	16	5	18	2	14	10	20	1377	0.000726
17	6	3	9	8	15	14	11	16	13	19	4	5	18	1	2	10	7	20	12	1323	0.000756
6	17	3	9	8	15	14	11	16	13	19	4	5	18	1	2	10	7	20	12	1345	0.000743

5.7. Chromosomal level mutation

The two mutation processes are allotted among the chromosomes within the population whereas the sub chromosomal level mutation is restricted within a single chromosome. The mutation process includes inverse mutation and single point mutation as given in sec 5.5.

6. Result analysis and discussion

The bench mark issues planned by Taillard [17] are tested against the meta-heuristic technique by Effective genetic algorithmic rule (EGA). Numerous sizes of the issues with twenty jobs and five, 10, twenty machines were tested and therefore the same issues were tested exploitation the EGA methodology and therefore the results are shown in Table 3.

From Table 4, it's found that the planned new meta-heuristic algorithmic rule that's EGA is giving higher results in comparison to any or all different algorithms. It's conjointly found that the developed EGA is incredibly abundant appropriate for twenty jobs five machines drawback, as a result of out of ten samples (IPTA001… IPTA010) EGA is giving seven higher results. Conjointly for twenty jobs ten machines drawback EGA is giving seven higher results in comparison different results. Then the program is dead for twenty jobs twenty machines drawback out of eight samples EGA has created seven higher results. From this Table 4 results, it's terminated that the developed EGA is giving higher results in comparison to any or all different algorithms conjointly it's giving best for big sized issues. The secret writing of the each RCGA & EGA were developed in Matlab atmosphere and therefore the programming was tested on associate Intel Core i5 processor 1.6 gigacycle per second system with 4GB RAM.

For the Taillard [10] benchmark drawback the obtained results by planned new meta-heuristic methodology (EGA) is compared with existing algorithms (Rajendran [8] and Ravindran at al [9] and Yagmahan, B., & Yenisey, M. M. [11] and Andrea Rossi [12]). Table 3 Consolidated results of the benchmark issues by EGA (Best makespan sequence) and Table 4 displays the comparison of results of Taillard Benchmark drawback with totally different heuristics. Conjointly the proportion of improvement is calculated are given in Table 5 and Table 6.

$$Percentage\ Improvement = \left(C_{yy} - C_{RCGA}\right) / C_{yy} \qquad (8)$$

where C_{yy} is erformance measure reported by Yagmahan, B., & Yenisey, M. M. [11]; C_{RCGA} is performance measure obtained by RCGA algorithms.

Percentage improvement of RCGA algorithmic rule over the past literature results is delineated in Table 5. From that it is ascertained that RCGA provides a pair of.94% average improvement in makespan with relevance (Rajendran [8]), conjointly it created three.02%, 5.61% and 5.53% average improvement in makespan over HAMC1, HAMC2 and HAMC3 Ravindran et al. [9] according values. Once more the algorithmic rule is tested with Yagmahan, B., & Yenisey, M. M. [11] however RCGA has created solely zero.16% of average improvement in makespan. Then the performance of RCGA is tested with NPFA-ACO developed by Andrea Rossi [12] and it's out performed by -1.55%. Then we've got an inspiration that to boost the performance RCGA by Effective genetic algorithmic rule (EGA) by modifying its parameters.

Percentage improvement of EGA algorithmic rule over the past literature results is delineated in Table 6. From that it is ascertained that RGA provides a 5.77% average improvement in makespan with relevance Rajendran [8], conjointly it created 5.83%, 8.35% and 8.26% average improvement in makespan over HAMC1, HAMC2 and HAMC3 Ravindran et al. [9] according values. Once more the algorithmic rule is tested with Yagmahan, B & Yenisey, M. M. [11] however EGA has created 3.08% of average improvement in makespan. Then the performance of EGA is tested with NPFA-ACO developed by Andrea Rossi [12] and it's performed 1.42% of average improvement in makespan. Any have EGA has created 21 best makespan sequences out of 28 benchmark Taillard problems.

Figs. 2 to 4 represents diagrammatically the performance of the important coded genetic algorithmic and Effective Genetic algorithmic over Rajendran [8], Y&Y [11] and NPFS-ACO [12] algorithms. Every bar in Figs. 2 to 4 shows the effectiveness of RCGA and EGA over all different algorithms within the chart.

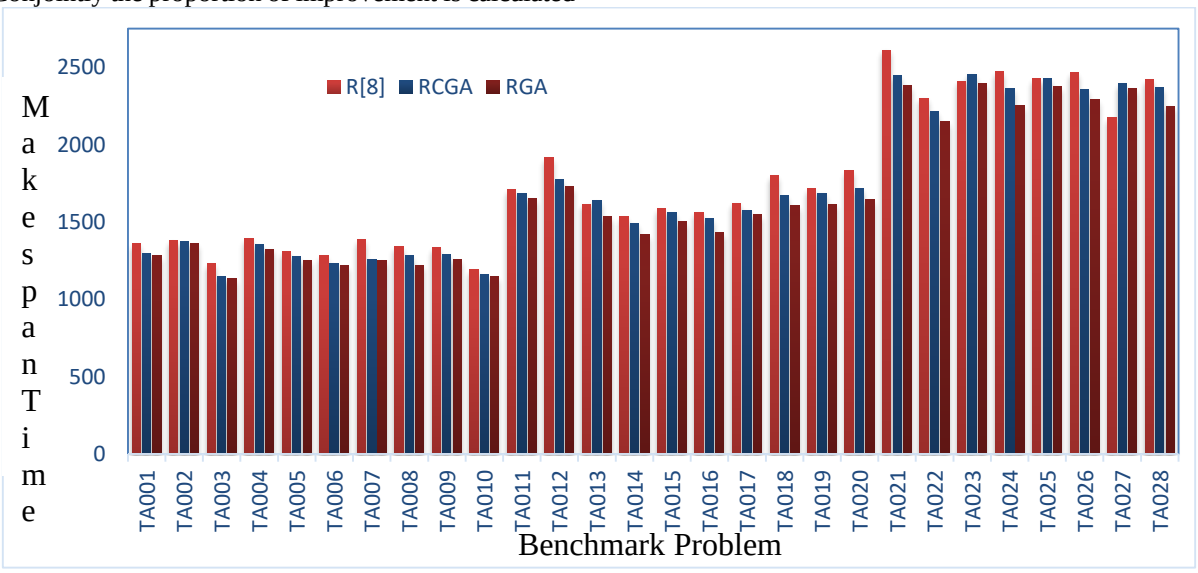

Fig. 2 Chart showing the comparison of EGA & RCGA algorithm with Rajendran [8]

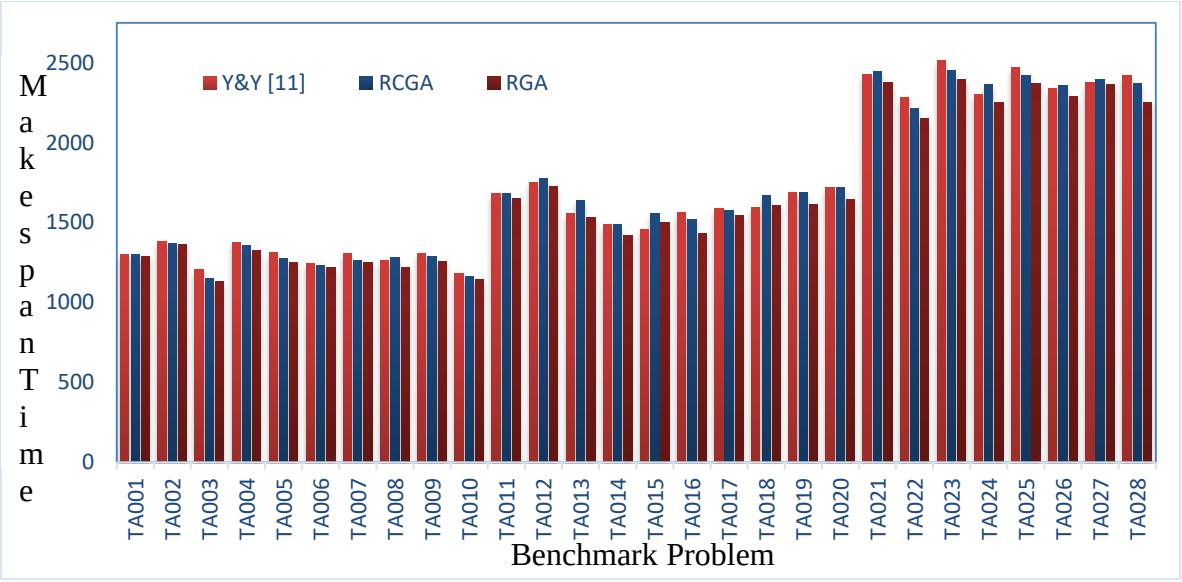

Fig. 3 Chart showing the comparison of EGA & RCGA algorithm with Y&Y [11]

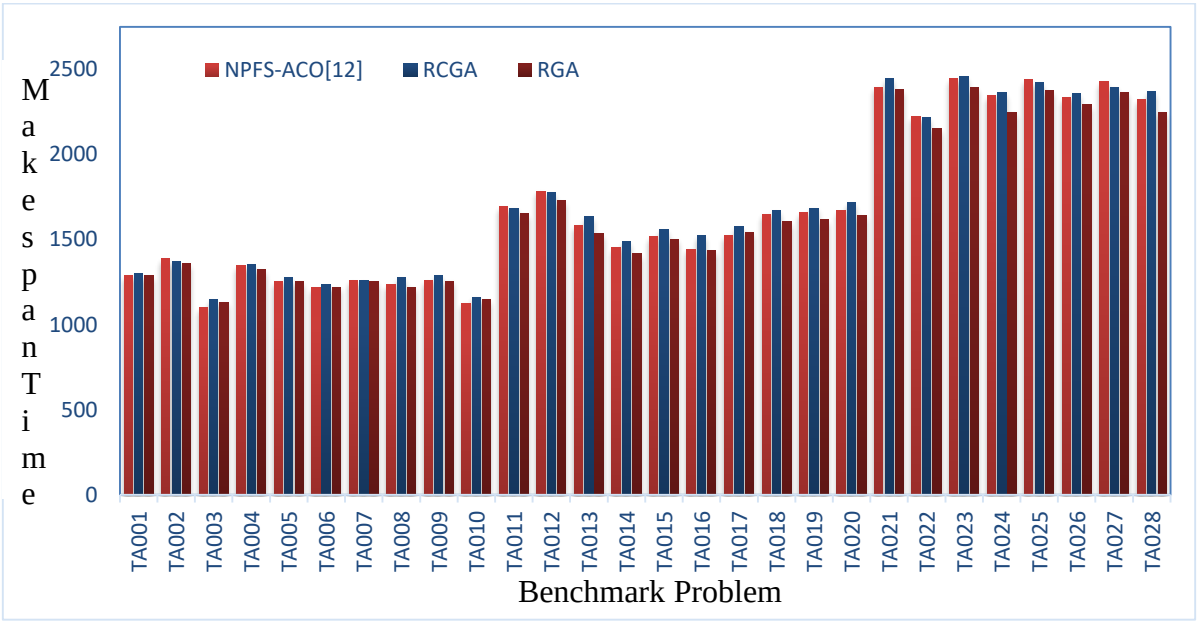

Fig. 4 Chart showing the comparison of EGA & RCGA algorithm with NPFS-ACO [12]

Table 3

Consolidated results of the benchmark problems by EGA (Best makespan sequence)

Benchmark Problem	Best makespan sequence																				Objective function
No.of jobs=20; No.of machines=5																					Makespan
TA001	3	17	9	8	15	14	11	16	13	19	6	4	5	18	1	2	10	7	20	12	1286
TA002	6	10	17	7	19	14	20	3	9	18	12	15	1	13	16	5	4	11	2	8	1359
TA003	16	14	19	11	3	20	18	7	1	12	10	5	2	9	4	17	6	8	13	15	1132
TA004	13	9	16	17	11	19	10	6	7	15	1	12	5	20	2	3	8	14	4	18	1325
TA005	3	5	12	10	19	9	18	17	15	13	4	16	6	2	14	11	1	7	8	20	1250
TA006	11	5	8	17	20	13	6	16	1	7	12	14	18	10	15	9	4	19	3	2	1220
TA007	5	2	15	11	6	20	13	19	1	17	7	9	12	3	8	4	16	14	18	10	1251
TA008	17	12	9	2	14	10	18	4	16	19	7	8	6	5	20	15	1	3	13	11	1221
TA009	4	2	10	12	1	18	17	6	16	3	13	11	9	5	20	14	7	15	8	19	1257
TA010	7	19	11	12	16	6	1	13	10	2	18	17	5	20	3	14	8	4	15	9	1145

Benchmark Problem	Best makespan sequence																			Objective function
No.of jobs=20; No.of machines=10																				
TA011	4	5	9	10	15	18	2	17	3	6	12	20	13	8	14	19	11	1	7	16	1652
TA012	12	17	5	13	15	7	20	9	11	19	10	1	6	2	3	4	18	16	8	14	1729
TA013	4	7	9	2	16	5	12	13	11	15	1	20	6	14	17	10	3	18	19	8	1534
TA014	18	20	3	11	9	13	4	16	15	1	10	2	7	8	6	19	12	14	17	5	1419
TA015	16	8	4	20	18	14	15	13	9	6	1	7	3	17	2	5	19	12	11	10	1502
TA016	19	8	20	3	18	16	11	14	6	15	13	4	5	7	12	17	10	9	2	1	1433
TA017	19	6	7	10	17	1	4	8	20	18	9	2	5	16	14	15	13	11	12	3	1545
TA018	17	8	20	4	7	18	14	2	5	9	19	3	6	11	1	13	15	10	16	12	1604
TA019	20	11	16	14	12	8	17	4	2	1	19	3	13	18	7	15	10	5	6	9	1617
TA020	5	12	14	13	17	9	19	4	7	8	16	6	20	2	10	3	18	1	15	11	1644
No.of jobs=20; No.of machines=20																				
TA021	8	9	10	12	13	15	16	17	11	5	1	20	14	2	18	6	7	3	4	19	2380
TA022	18	3	11	4	5	13	1	12	16	19	15	6	14	10	20	17	7	9	8	2	2150
TA023	19	4	1	13	5	20	11	9	16	8	15	17	18	3	12	2	10	14	6	7	2393
TA024	14	3	18	5	2	8	12	4	6	20	15	13	1	7	19	16	10	17	9	11	2250
TA025	10	2	5	19	9	11	15	13	3	18	17	4	20	12	14	1	16	8	7	6	2373
TA026	18	6	11	2	8	20	16	9	17	4	13	15	10	14	5	1	3	7	12	19	2290
TA027	10	12	16	14	5	19	18	6	7	17	4	2	11	15	20	8	9	3	1	13	2362
TA028	4	2	16	10	20	5	1	14	6	7	3	11	17	19	13	12	8	18	15	9	2249

Table 4

Comparison of results of Taillard benchmark problems with different heuristics

Benchmark Problem	UB/ Best Known	Rajendran [8]	Ravindran et al. [9]			Yagmahan and Yenisey [11]	NPFS-ACO [12]	RCGA	EGA
			HAMC1	HAMC2	HAMC3				
	No.of jobs=20; No.of machines=5								
TA001	1278	1359	1297	1324	1307	1297	1290	1297	**1286**
TA002	1358	1378	1373	1409	1409	1383	1389	1371	**1359**
TA003	1073	1230	1206	1210	1210	1203	1100	1149	1132
TA004	1292	1393	1402	1423	1418	1377	1344	1355	**1325**
TA005	1231	1307	1334	1387	1387	1311	1250	1276	**1250**
TA006	1193	1282	1238	1281	1281	1245	1217	1234	1220
TA007	1234	1387	1322	1359	1332	1303	1258	1259	**1251**
TA008	1199	1344	1287	1404	1404	1265	1235	1280	**1221**
TA009	1210	1335	1307	1382	1382	1303	1258	1290	**1257**
TA010	1103	1191	1195	1298	1221	1179	1127	1162	1145
	No.of jobs=20; No.of machines=10								
TA011	1560	1711	1774	1812	1787	1681	1693	1685	**1652**
TA012	1644	1916	1791	1817	1832	1749	1785	1775	**1729**
TA013	1486	1617	1643	1784	1783	1554	1583	1636	**1534**
TA014	1368	1533	1531	1595	1584	1490	1452	1490	**1419**
TA015	1413	1588	1722	1557	1586	1455	1516	1558	1502
TA016	1369	1565	1612	1674	1667	1564	1445	1522	**1433**
TA017	1428	1622	1594	1624	1628	1590	1524	1576	1545
TA018	1527	1800	1631	1659	1659	1595	1650	1671	1604
TA019	1586	1717	1769	1842	1823	1689	1659	1686	**1617**
TA020	1559	1831	1744	1831	1793	1719	1670	1718	**1644**
	No.of jobs=20; No.of machines=20								
TA021	2293	2610	2491	2539	2546	2428	2396	2446	**2380**
TA022	2092	2301	2491	2491	2586	2281	2225	2215	**2150**
TA023	2313	2411	2422	2433	2506	2515	2446	2455	**2393**
TA024	2223	2471	2567	2693	2722	2299	2346	2363	**2250**

Benchmark Problem	UB/ Best Known	Rajendran [8]	Ravindran et al. [9]			Yagmahan and Yenisey [11]	NPFS-ACO [12]	RCGA	EGA
			HAMC1	HAMC2	HAMC3				
TA025	2291	2427	2420	2453	2493	2473	2439	2424	**2373**
TA026	2221	2466	2557	2641	2663	2339	2331	2356	**2290**
TA027	2267	2174	2448	2528	2515	2378	2428	2396	2362
TA028	2183	2418	2464	2473	2472	2418	2321	2368	**2249**

Table 5

Percentage improvement of RCGA algorithm over the past literature

Benchmark Problem	Imp.% of RCGA over [8]	Imp.% of RCGA over HAMC1	Imp.% of RCGA over HAMC2	Imp.% of RCGA over HAMC3	Imp.% of RCGA over Y&Y [11]	Imp.% of RCGA over NPFS-ACO [12]
No.of jobs=20; No.of machines=5						
TA001	4.56	0.00	2.04	0.77	0.00	-0.54
TA002	0.51	0.15	2.70	2.70	0.87	1.30
TA003	6.59	4.73	5.04	5.04	4.49	-4.45
TA004	2.73	3.35	4.78	4.44	1.60	-0.82
TA005	2.37	4.35	8.00	8.00	2.67	-2.08
TA006	3.74	0.32	3.67	3.67	0.88	-1.40
TA007	9.23	4.77	7.36	5.48	3.38	-0.08
TA008	4.76	0.54	8.83	8.83	-1.19	-3.64
TA009	3.37	1.30	6.66	6.66	1.00	-2.54
TA010	2.43	2.76	10.48	4.83	1.44	-3.11
No.of jobs=20; No.of machines=10						
TA011	1.52	5.02	7.01	5.71	-0.24	0.47
TA012	7.36	0.89	2.31	3.11	-1.49	0.56
TA013	-1.18	0.43	8.30	8.24	-5.28	-3.35
TA014	2.80	2.68	6.58	5.93	0.00	-2.62
TA015	1.89	9.52	-0.06	1.77	-7.08	-2.77
TA016	2.75	5.58	9.08	8.70	2.69	-5.33
TA017	2.84	1.13	2.96	3.19	0.88	-3.41
TA018	7.17	-2.45	-0.72	-0.72	-4.76	-1.27
TA019	1.81	4.69	8.47	7.52	0.18	-1.63
TA020	6.17	1.49	6.17	4.18	0.06	-2.87
No.of jobs=20; No.of machines=20						
TA021	6.28	1.81	3.66	3.93	-0.74	-2.09
TA022	3.74	11.08	11.08	14.35	2.89	0.45
TA023	-1.82	-1.36	-0.90	2.04	2.39	-0.37
TA024	4.37	7.95	12.25	13.19	-2.78	-0.72
TA025	0.12	-0.17	1.18	2.77	1.98	0.62
TA026	4.46	7.86	10.79	11.53	-0.73	-1.07
TA027	-10.21	2.12	5.22	4.73	-0.76	1.32
TA028	2.07	3.90	4.25	4.21	2.07	-2.02

Table 6

Percentage improvement of EGA algorithm over the past literature

Benchmark Problem	Imp.% of RCGA over [8]	Imp.% of RCGA over HAMC1	Imp.% of RCGA over HAMC2	Imp.% of RCGA over HAMC3	Imp.% of RCGA over Y&Y [11]	Imp.% of RCGA over NPFS-ACO [12]
No.of jobs=20; No.of machines=5						
TA001	5.37	0.85	2.87	1.61	0.85	0.31
TA002	1.38	1.02	3.55	3.55	1.74	2.16
TA003	7.97	6.14	6.45	6.45	5.90	-2.91
TA004	4.88	5.49	6.89	6.56	3.78	1.41
TA005	4.36	6.30	9.88	9.88	4.65	0.00
TA006	4.84	1.45	4.76	4.76	2.01	-0.25

Benchmark Problem	Imp.% of RCGA over [8]	Imp.% of RCGA over HAMC1	Imp.% of RCGA over HAMC2	Imp.% of RCGA over HAMC3	Imp.% of RCGA over Y&Y [11]	Imp.% of RCGA over NPFS-ACO [12]
TA007	9.81	5.37	7.95	6.08	3.99	0.56
TA008	9.15	5.13	13.03	13.03	3.48	1.13
TA009	5.84	3.83	9.04	9.04	3.53	0.08
TA010	3.86	4.18	11.79	6.22	2.88	-1.60
No.of jobs=20; No.of machines=10						
TA011	3.45	6.88	8.83	7.55	1.73	2.42
TA012	9.76	3.46	4.84	5.62	1.14	3.14
TA013	5.13	6.63	14.01	13.97	1.29	3.10
TA014	7.44	7.32	11.03	10.42	4.77	2.27
TA015	5.42	12.78	3.53	5.30	-3.23	0.92
TA016	8.43	11.10	14.40	14.04	8.38	0.83
TA017	4.75	3.07	4.86	5.10	2.83	-1.38
TA018	10.89	1.66	3.32	3.32	-0.56	2.79
TA019	5.82	8.59	12.21	11.30	4.26	2.53
TA020	10.21	5.73	10.21	8.31	4.36	1.56
No.of jobs=20; No.of machines=20						
TA021	8.81	4.46	6.26	6.52	1.98	0.67
TA022	6.56	13.69	13.69	16.86	5.74	3.37
TA023	0.75	1.20	1.64	4.51	4.85	2.17
TA024	8.94	12.35	16.45	17.34	2.13	4.09
TA025	2.22	1.94	3.26	4.81	4.04	2.71
TA026	7.14	10.44	13.29	14.01	2.09	1.76
TA027	-8.65	3.51	6.57	6.08	0.67	2.72
TA028	6.99	8.73	9.06	9.02	6.99	3.10

Based on the higher than discussion show that the planned EGA and RCGA works higher than SGA, HAMC's, ACO by Y& Y [11], hymenopteran colony letter by Andrea Rossi [12] for all the benchmark issues and it giver for higher results. It is ascertained that the planned EGA algorithmic rule is additional economical than the present one for resolution flowshop programming issues with minimizing makespan as criteria, and this work is extended to solve bi-objective or different appropriate objectives either combined or individually.

7. Conclusions

In this paper, a new Effective Genetic algorithm (EGA) and Real Coded Genetic algorithm (RCGA) is planned to solve flow shop scheduling problems to reduce makespan. To verify the computational performance of the EGA and RCGA, an enormous experiment were conducted with Tailard benchmark problem for the various sizes of the problems with 20, 50 & 100 jobs through 5, 10 & 20 machines. The obtained results by planned EGA are one step ahead of all the algorithms for the flow shop scheduling problem. The planned algorithm will be used for single or combinational problems considering completely different conditions like total flow time, total tardiness and maximum tardiness. Furthermore the proposed algorithm will be implemented for scheduling problems in numerous manufacturing systems such as job shops, flexible flow shops, and cellular manufacturing and flexible manufacturing systems with respect to completely different objectives. The future work includes application EGA to combinational optimization problems

References

1. **Johnson, S.M.** 1954. Optimal two- and three-stage production schedules with setup times included. Naval Research Logistics Quarterly 1(1): 61–68. http://dx.doi.org/10.1002/nav.3800010110.
2. **Rajendran, C.** 1992. Two-stage flowshop scheduling problem with bicriteria. Journal of the Operational Research Society 43(9): 871–884. http://dx.doi.org/10.1057/jors.1992.126.
3. **Anderson, Rajendran, C.; Ziegler, H.** 2004. Ant-colony algorithms for flowshop scheduling to minimize makespan/total flowtime of jobs. European Journal of Operational Research 155(2): 426–438. http://dx.doi.org/10.1016/S0377-2217(02)00908-6.
4. **Neppalli, V.R.; Chen, C.L.; Gupta, J.N.D.** 1996. Genetic algorithms for the twostage bicriteria flowshop problem. European Journal of Operational Research 95(2): 356–373. http://dx.doi.org/ 10.1016/0377-2217(95)00275-8.
5. **T'kindt, V.; Gupta, J.N.D.; Billaut, J.C**. 2003. Two-machine flowshop scheduling with a secondary criterion. Computers and Operations Research 30(4): 505–526. http://dx.doi.org/10.1016/S0305-0548(02)00021-7.
6. **Jeen Robert, R.B.; Rajkumar, R.** 2016. A Hybrid Algorithm for Minimizing Makespan in the Permutation Flow Shop Scheduling Environment. Asian Journal of Research in Social Sciences and Humanities Vol. 6, No. 9, September 2016: 1239-1255. http://dx.doi.org/10.5958/2249-7315.2016.00867.4.

7. **Yeh, W.C.** 2002. A Memetic Algorithm for the n/2/Flowshop/αF + βCmax Scheduling Problem. The International Journal of Advanced Manufacturing Technology 20(6): 464–473. http://dx.doi.org/10.1007/s001700200179.
8. **Rajendran, C.** 1995. Heuristics for scheduling in flowshop with multiple objectives. European Journal of Operational Research 82: 540–555. http://dx.doi.org/10.1016/0377-2217(93)E0212-G.
9. **Ravindran, D.; Noorul Haq, A.; Selvakuar, S.J.; Sivaraman, R.** 2005. Flow shop scheduling with multiple objective of minimizing makespan and total flow time. International Journal of Advanced Manufacturing Technology 25, 1007–1012. http://dx.doi.org/10.1007/s00170-003-1926-1.
10. **Taillard, E.** 1993. Benchmarks for basic scheduling problems. European Journal of Operational Research 64: 278–285. http://dx.doi.org/ 10.1016/0377-2217(93)90182-M.
11. **Betul Yagmahan; Mehmet Mutlu Yenisey.** 2010. A multi-objective ant colony system algorithm for flow shop scheduling problem. Expert Systems with Applications 37 (2010): 1361–1368. http://dx.doi.org/10.1016/j.eswa.2009.06.105.
12. **Andrea Rossi; Michele Lanzetta.** 2013. Scheduling flow lines with buffers by ant colony digraph. Expert Systems with Applications 40 (2013): 3328–3340. http://dx.doi.org/10.1016/j.eswa.2012.12.041.
13. **David E. Goldberg.** 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. ADDISON-WESLEY PUBLISHING COMPANY, INC. 1-405. http://dx.doi.org/10.1023/A:1022602019183.
14. **Wei-Der Chang.** 2007. Non-linear system identification and control using a real-coded genetic algorithm, Applied Mathematical Modelling 31(3): 541–550. http://dx.doi.org/10.1016/j.apm.2005.11.024.
15. **Goldberg, D.E.** 1991. Real-coded genetic algorithms, virtual alphabets and blocking. Complex Systems 5: 139-167. http://dx.doi.org/ 10.1.1.52.9880.
16. **Nawaz, M.; Enscore, E.; Ham, I.** 1983. A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. OMEGA, 11(1): 91–95. http://dx.doi.org/10.1016/0305-0483(83)90088-9.
17. **Taillard, E.** 1990. Some efficient heuristic methods for the flowshop-sequencing problem. European Journal of Operational Research 47(1): 65–74. http://dx.doi.org/10.1016/0377-2217(90)90090-X.
18. **Muthiah, A.R.; Rajkumar, B. Muthukumar**, 2015. Minimizing Makespan in Job Shop Scheduling Problem Using Genetic Algorithm, Applied Mechanics and Materials 813-814: 1183-1187. http://dx.doi.org/10.4028/www.scietific.net/AMM.813-814.1183.

R.B. Jeen Robert, R. Rajkumar

AN EFFECTIVE GENETIC ALGORITHM FOR FLOW SHOP SCHEDULING PROBLEMS TO MINIMIZE MAKESPAN

S u m m a r y

In this paper, the flowshop scheduling problem with the objective of minimizing the makespan has important applications in an exceedingly type of industrial systems. The main concern of flow shop scheduling is to get the most effective sequence, that minimizes the makespan, time of flow, time of idle, delay, etc. the objective of minimizing makespan is planned for finding the flowshop scheduling problem with Effective Genetic algorithm (EGA). EGA could be an easy and efficient algorithm that is employed to resolve for each single and multi-objective problem in flow shop environment. This algorithm can works simply for our real life applications. The planned algorithm is tested with well-known problems in literature. EGA's resolution performance has been compared with the present results reported by researchers. The obtained results show that the planned EGA performs higher than NPFS-ACO algorithms in finding the flowshop scheduling problem with the makespan criterion as average percentage improvement of 1.42%. This improvement ends up in two completely different meta-heuristic algorithms for finding flow shop planning problems specifically real coded Genetic algorithm (RCGA) and EGA. However EGA is performed well once comparing with RCGA.

Keywords: Flow shop scheduling, Makespan, Genetic Algorithm, Matlab.

Received May 24, 2016
Accepted August 04, 2017