World Academy of Science, Engineering and Technology
International Journal of Industrial and Manufacturing Engineering
Vol:7, No:1, 2013

A Dual Fitness Function Genetic Algorithm: Application on Deterministic Identical Machine Scheduling

Saleem Z. Ramadan, Gürsel A. Süer

***Abstract*—**In this paper a genetic algorithm (GA) with dual-fitness function is proposed and applied to solve the deterministic identical machine scheduling problem. The mating fitness function value was used to determine the mating for chromosomes, while the selection fitness function value was used to determine their survivals. The performance of this algorithm was tested on deterministic identical machine scheduling using simulated data. The results obtained from the proposed GA were compared with classical GA and integer programming (IP). Results showed that dual-fitness function GA outperformed the classical single-fitness function GA with statistical significance for large problems and was competitive to IP, particularly when large size problems were used.

***Keywords*—**Machine scheduling, Genetic algorithms, Due dates, Number of tardy jobs, Number of early jobs, Integer programming, Dual Fitness functions.

I. Introduction

IN classical genetic algorithm, a single-fitness function value determines the chance for a chromosome to mate and survive. In some biological systems the set of traits that allows the species to survive are different from the set of traits that allows them to find a suitable mate and thus reproduce. Traits like physical strength, speed, and strength of vision may be essential for some species to survive in their surroundings (avoiding danger and increasing ability to hunt) while these traits may have minor or no importance when it comes to reproduce (attracting a partner). Traits like color, size, and strength of smell can be essential in attracting a partner while these traits may have minor or no importance for surviving. To imitate this biological system in genetic algorithms, two different fitness functions need to be used; one for mating and another one for selection.

The purpose of the mating fitness function (MFF) is to select chromosomes for mating, while the purpose of the selection fitness function (SFF) is to select chromosomes for the next generation. This means that the MFF considers the relation between the genes among the perspective mating couple to have a "successful" crossover while the SFF considers the genes of only one chromosome to determine its chances for survival.

In this study we claim that using two different fitness functions enhances the quality of the solution found by genetic algorithms that adapt this strategy, hence, a dual-fitness function genetic algorithm is proposed. This algorithm is applied to deterministic identical parallel machine scheduling problem to test its effectiveness. The problem is solved by the proposed algorithm and is compared to solutions generated by both classical GA and IP.

The rest of the paper will be organized as follows: Section II presents the literature review, Section III presents the problem statement, Section IV presents the proposed genetic algorithm, Section V presents an IP formulation for the problem in hand, Section VI presents illustrative example, Section VII presents the experimentations and results, and Section VIII presents the discussion and conclusions.

Saleem Z. Ramadan is with the Mechanical and Industrial Engineering Department, Applied Science Private University, Amman-Jordan-11931, P.O. Box 166 (e-mail: s_ramadan@asu.edu.jo, suer@ohio.edu).

Süer A. Gürsel is with the Industrial and Systems Engineering, Ohio University, Athens, OH 45701, USA (e-mail: suer@ohio.edu).

II. Literature Review

A comprehensive review of research in parallel machine scheduling problems and solution methodologies is provided by Cheng and Sin [1]. Different approaches to the problem include heuristics, exact methods and a number of approximate algorithms. Bedworth and Bailey [2] suggested a simple heuristic in their book. Suer et al. [3] presented three heuristics (SBC-1, SBC-2, SBC-3) to minimize the number of tardy jobs in identical machine scheduling. A decomposition-based branch and bound approach for minimizing the number of tardy jobs, total weighted earliness and tardiness criteria for problems with common due date was proposed by Chen and Powell [4].

Evolutionary Computation (EC) techniques have also found application in this domain. Cheng and Gen [5] proposed a genetic algorithm combined with a local optimizer to minimize the maximum weighted absolute lateness in parallel machine scheduling. An evolutionary computation approach for minimizing the number of tardy jobs was presented by Liu and Wu [6]. The solution was encoded as a segmented digit string with the list of jobs in the first segments and the machine order in the second segment. A Gaussian mutation operator and neighborhood-based local search mechanism were employed in the algorithm. Genetic algorithms have also been applied to non due date related objectives. A hybrid approach which combines a greedy heuristic with a pure genetic algorithm through an insertion method was proposed by Luu et al. [7] for the parallel machine batch sequencing problem. Min and Cheng [8] presented a genetic algorithm approach for minimizing the make span in scheduling identical machines. Integer Programming Model for parallel

machine scheduling is given in Suer et al. [9]. Fariborz et al. [10] used GA to optimize bi-criteria for a single machine scheduling. The two criteria used were maximizing the earliness and minimizing the tardiness. Seeded initial population was used in this GA. The fitness function used was the weighted average of the two criterions. Roulette-wheel strategy was used for selection and three different crossover strategies were used, one point, two points and cycle crossover. Cycle crossover showed the best results. Johnny and Long [11] proposed job-focused and machine-focused approaches to minimize number of tardy jobs in *m* parallel machine scheduling. Hui-Yuan et al., [12] proposed a dual fitness function genetic algorithm to distinguish the local optima basin from the global optima basin. The results showed its effectiveness.

III. Problem Statement

Deterministic identical machine scheduling problem is defined when n deterministic jobs have to be scheduled on m machines with identical capabilities, i.e., when the processing time of a job is the same on all machines. However, processing times vary from one job to another. All ready times are assumed to be zero and a job becomes tardy when its completion time exceeds its due date. The objective for this problem is to schedule the jobs such that the number of tardy jobs (nT) is minimized.

The assumptions made in this problem are as follows:

- *m* identical machines are used, i.e., the processing time for each job doesn't differ from one machine to the other.
- Each machine can only process one job at a time before another job can be processed, i.e., no multiple processing in any machine.
- No preemption, i.e., each job must be finished totally before another job can be started.
- Processing times, due dates, and completion dates are all deterministic.
- Set-up times are sequence-independent and can be included in the process time.
- Number of machines, *m*, is fixed and number of jobs, *n*, is also fixed such that $n>m$.
- All jobs and all machines are available and ready at time zero.
- Jobs are independent and equally important.

The following notations are used throughout this study:

n number of jobs to be scheduled.
m number of machines used.
p_i processing time of job i.
d_i due date of job i.
C_i completion time of job i.
nT_j number of tardy jobs on machine j.

IV. The Proposed Genetic Algorithm

Evolutionary algorithms (EAs), as metaheuristic optimization techniques, are characterized with a population that evolves over time. The underlying principle for EAs can be traced back to Darwin's theory of evolution. The principle is easy; given a population of individuals, the environmental pressure will dictate which individual will survive by a process called natural selection. In short, the strong individual will adapt to the surrounding and survives and the weak one will not adapt to the surrounding and eventually dies out. GAs have certain traits that make them the most popular among different EAs. GAs use both crossover and mutation operators which make their population more diverse and consequently more immune to be trapped in a local optima. In theory, the diversity helps the algorithm to be faster in reaching the global optima as it allows the algorithm to explore the solution space faster.

A. Overview of the Algorithm

The following strategies were adopted in the proposed genetic algorithm:

1. Chromosome Representation

The chromosome representation for this GA consists of a row vector of N genes, where N is computed as given in (1):

$$N = n + (1 - (\frac{n}{m} - \text{floor}(\frac{n}{m}))) \times m \quad (1)$$

where $\text{floor}(\frac{n}{m})$ is the smallest integer of the fraction $\frac{n}{m}$. From these N genes there are $(N - n)$ dummy jobs with zero processing times and infinite due dates. The first $\frac{N}{m}$ genes are for the first machine, the second $\frac{N}{m}$ genes are for the second machine, and the jth $\frac{N}{m}$ genes are for the jth machine.

2. Initial Population

A seeded initial population strategy is used in which S chromosomes are constructed as follows: jobs are sorted according to their processing times descending order. After that, jobs are assigned to machines from the top of the list in a systematic way such that all of the machines are covered with almost the same working hours. Of course this does not guarantee to have the minimum total number of tardy jobs, but it is believed that it will balance the load on the machines better than random assignment, consequently it will form a better initial solutions. This process is repeated S times with different random starting points to form S different initial chromosomes.

3. Mating

The mating strategy used in this GA is based on the average variance of the processing times such that the higher the average variance the fitter the parents to mate. The MFF is given as in (2):

$$MFF = \frac{\sum_{j=1}^{m} \text{Var}_j(P)}{m} \quad (2)$$

where P is the set of distinct jobs in machine j in both chromosomes under consideration without considering any dummy jobs. In this mating strategy $\sum_{j=1}^{m}(m - j)$ distinct couples are mate. These couples are ranked, descending, based on their MFF values such that the higher the fitness value the fitter the couple is. The top β of these couples are chosen for possible crossover.

The reasoning behind this mating strategy can be explained as follows: it is expected that as the variance of the processing times for a group of jobs scheduled on a certain machine increases, the variance of the due dates for this group of jobs will also increase. A high variance of the due dates for a group of jobs increases the probability of low number of tardy jobs on the group's machine. Consequently, the total number of tardy jobs for the whole schedule is expected to decrease.

4. Selection

Enlarged sample space strategy $(s+\lambda)$ is available selection in this GA. In this strategy λ offspring and S parents will compete for their survivals using elitist selection strategy based on their SFF value.

The SFF value is calculated by adding the number of tardy jobs on each machine for the entire schedule as given in (3) where nTj denotes the number of tardy jobs on machine j:

$$SFF = \sum_{j=1}^{m} nT_j \quad (3)$$

5. Crossover

Mating fitness values will determine the best couples for crossover. Let P1Mj be the genes for machine j in parent 1 and P2Mj be the genes for machine j in parent 2. Furthermore, assume that the mating fitness value for parent 1 and parent 2 permits them to have a crossover. Crossover will be done between P1Mj and P2Mj for all values of j = 1, 2, … m. The total number of crossovers that will be done for any pair of parents are m. The crossover will take place on the positions of the common-set of genes between the pair of machines under consideration such that the positions of the common-set genes for P1Mj will be replaced by the positions of the common-set of P2Mj and vice versa. The remaining genes will be assigned randomly. This crossover strategy guarantees that the generated offspring is feasible. This procedure is explained in section VI in details. A 70% crossover rate is used. The reason for using such a high crossover rate is to increase the diversity in the population to explore the solution space adequately and to reduce the chances of premature convergence problem.

6. Mutation

Random mutation strategy will be adopted in which two distinct gene's locations are randomly selected and the values of the two genes are switched. This guarantees that the mutated chromosome is feasible. This procedure is explained in details in section VI. A mutation rate of 10% is used.

7. Termination Criterion

The GA will terminate when N_g generations are reached.

V. Integer Programming

For the sake of comparison, the deterministic identical machine scheduling problem will be solved using IP model. It should be noticed that Minimizing number of tardy jobs (nT) is equivalent to maximizing number of early jobs (nE). This IP model was presented in Suer et al. [3].

VI. Illustration Example for the GA Strategies Used

In this section, the different strategies used in the proposed GA are illustrated. Suppose we have 13 jobs that need to be scheduled on three identical machines, i.e., n =13 and m = 3. Moreover, suppose that the processing times and the due dates for these jobs are as given in Table I:

TABLE I
DATA FOR THE ILLUSTRATION EXAMPLE

Job #	Processing Time (Pi)	Due date (di)
1	8.3325	53.3263
2	9.1521	49.3816
3	2.1429	15.9718
4	9.2204	59.0218
5	6.6912	11.6783
6	1.8779	29.2215
7	3.5065	23.6932
8	5.9216	10.8038
9	9.6176	43.4556
10	9.6840	32.5532
11	2.4185	3.9672
12	9.7353	53.0977
13	9.6145	32.5989

Using (1), the total number of genes, N, used in any chromosome is:

$$N = 13 + (1 - (\frac{13}{3} - \text{floor}(\frac{13}{3}))) \times 3 = 15$$

and the number of dummy jobs are

$$(N - n) = 15 \quad 13 = 2.$$

This means that each of the three machines will have 5 jobs.

The chromosome representation for this problem is represented as shown in Fig. 1.

Machine 1					Machine 2					Machine 3				
job 1	job 2	job 3	job 4	job 5	job 6	job 7	job 8	job 9	job 10	job 11	job 12	job 13	job 14	job 15

Fig. 1 Chromosome representation used in the proposed GA

For a population size of 3, the initial population is generated as follows:

- jobs are sorted according to their processing times descending as shown in Table II.
- jobs are assigned to machines starting from a random job according to the following trend { machine 1 machine 2 machine 3 machine 3 machine 2 machine 1 machine 1 ...}. Starting from job 12, the assignment results are shown in Table III.

TABLE II
SORTED JOBS BASED ON PROCESSING TIMES

Job #	Processing Time (P_i)	Due date (d_i)
12	9.7353	53.0977
10	9.6840	32.5532
9	9.6176	43.4556
13	9.6145	32.5989
4	9.2204	59.0218
2	9.1521	49.3816
1	8.3325	53.3263
5	6.6912	11.6783
8	5.9216	10.8038
7	3.5065	23.6932
11	2.4185	3.96720
3	2.1429	15.9718
6	1.8779	29.2215
14	0	Inf
15	0	Inf

TABLE III
ASSIGNING JOBS TO MACHINE

Machine 1		Machine 2		Machine 3	
Job #	**(P_i)**	Job #	**(P_i)**	Job #	**(P_i)**
12	9.7353	10	9.6840	9	9.6176
2	9.1521	4	9.2204	13	9.6145
1	8.3325	5	6.6912	8	5.9219
3	2.1429	11	2.4185	7	3.5065
6	1.8779	14	0	15	0
Total time	31.2407		28.0141		28.6605

The corresponding chromosome is shown in Fig. 2.

This process is repeated 2 more times with same assignment trend and different random starting points to form two more different chromosomes as shown in Fig. 3. The starting point for chromosome 2 is job 4 and for chromosome 3 is job 3.

Chromosome 1

Machine 1					Machine 2					Machine 3				
job 12	job 2	job 1	job 3	job 6	job 10	job 4	job 5	job 11	job 14	job 9	job 13	job 8	job 7	job 15

Fig. 2 First chromosome in the initial population

Chromosome 2

Machine 1					Machine 2					Machine 3				
job 4	job 7	job 11	job 12	job 10	job 2	job 8	job 3	job 15	job 9	job 1	job 5	job 6	job 14	job 13

Chromosome 3

Machine 1					Machine 2					Machine 3				
job 3	job 10	job 9	job 1	job 11	job 6	job 12	job 13	job 5	job 8	job 14	job 15	job 4	job 2	job 7

Fig. 3 Second and third chromosomes in the initial population

For the mating strategy, $\sum_{j=1}^{m}(m \quad j) = (3-1)+(3-2)+(3-3) = 3$ different pairs of parents are available for crossover namely, Ch1/Ch2, Ch1/Ch3, Ch2/Ch3. The MFF value calculations for one pair only will be shown, say Chromosome 1 with Chromosome 2 (Ch1/Ch2).

In this scenario, the P set for machine 1 is { job 12, job 2, job 1, job 3, job 6, job 4, job 7, job 11, job 10 }, the variance for processing times for these jobs can be calculated as:

$$Var_1(P_{12}, P_2, P_1, P_3, P_6, P_4, P_7, P_{11}, P_{10})$$
$$= Var_1\left(\left(\left(\begin{matrix}(9.7353,\ 9.1521,\ 8.3325,\ 2.1429,\ 1.8779,\\ 9.2204,\ 3.5065,\ 2.4185,\ 9.864)\end{matrix}\right)\right)\right) = 13.1227$$

Calculations for machine 2 and machine 3 are as follows:

$$Var_2(P_{10}, P_4, P_5, P_{11}, P_2, P_8, P_3, P_9) = 9.8988$$

$$Var_3(P_9, P_{13}, P_8, P_7, P_1, P_5, P_6) = 10.0217$$

Note that for machine 2 and machine 3 jobs 14 and 15 were ignored from the calculations as they are dummy jobs.

The mating fitness value for this pair of parents is:

Ch1/Ch2:

$$\text{MFF} = \frac{\sum_{j=1}^{3} Var_j(P_i)}{3} = \frac{13.1227 + 9.8988 + 10.0217}{3} = 11.0144$$

The mating fitness values for the other two pairs are as follows:

Ch1/Ch3:

$$\text{MFF} = \frac{\sum_{j=1}^{3} Var_j(P_i)}{3} = \frac{11.4049 + 10.69263 + 6.479059}{3} = 9.5255$$

Ch2/Ch3:

$$\text{MFF} = \frac{\sum_{j=1}^{3} Var_j(P_i)}{3} = \frac{12.11488 + 10.95602 + 9.44273}{3} = 10.83788$$

If we choose $(\beta = 1)$ then Ch1/Ch2 is the only two chromosomes that will mate according to the mating strategy as they have the highest mating fitness value.

The crossover will take place on the positions of the common-set of genes between the pair of machines under consideration such that the positions of the common-set genes for P1Mj will be replaced by the positions of the common-set of P2Mj and vice versa. The remaining jobs will be assigned randomly for the remaining positions.

From Figs. 1 and 2, the common-set of genes between the different pairs of machines are as follows:

M1/M1: {job12}, the positions for this set is as follows: Ch1: {1} and for Ch2: {4}, for M2/M2 and M3/M3 there are no common jobs between them in this pair of chromosomes.

Applying the crossover operator on Ch1/Ch2 couple will generate two offspring as shown in Figs. 4 and 5.

Chromosome 1

Machine 1					Machine 2					Machine 3				
job 12	job 2	job 1	job 3	job 6	job 10	job 4	job 5	job 11	job 14	job 9	job 13	job 8	job 7	job 15

Offspring1

Machine 1					Machine 2					Machine 3				
job 2	job 6	job 3	job 12	job 1	job 4	job 5	job 10	job 14	job 11	job 13	job 8	job 7	job 9	job 15

Chromosome 2

Machine 1					Machine 2					Machine 3				
job 4	job 7	job 11	job 12	job 10	job 2	job 8	job 3	job 15	job 9	job 1	job 5	job 6	job 14	job 13

Fig. 4 First offspring of Ch1/Ch2 couple

Chromosome 1

Machine 1					Machine 2					Machine 3				
job 12	job 2	job 1	job 3	job 6	job 10	job 4	job 5	job 11	job 14	job 9	job 13	job 8	job 7	job 15

Offspring2

Machine 1					Machine 2					Machine 3				
job12	job 10	job 11	job 4	job 7	job 3	job 8	job 2	job 9	job15	job 6	job 14	job 1	job 5	job 13

Chromosome 2

Machine 1					Machine 2					Machine 3				
job 4	job 7	job 11	job 12	job 10	job 2	job 8	job 3	job 15	job 9	job 1	job 5	job 6	job 14	job 13

Fig. 5 Second offspring of Ch1/Ch2 couple

To mutate the offspring, two distinct locations will be randomly determined for each offspring and the values of these two genes will be switched. Assume location 10 and location 5 were randomly chosen for Offspring 1 and locations 7 and 12 were randomly chosen for Offspring 2, the mutated offspring is given as in Fig. 6.

Mutated Offspring1

Machine 1					Machine 2					Machine 3				
job 2	job 6	job 3	job 12	job 11	job 4	job 5	job 10	job 14	job 1	job 13	job 8	job 7	job 9	job 15

Mutated Offspring2

Machine 1					Machine 2					Machine 3				
job12	job 10	job 11	job 4	job 7	job 3	job 14	job 2	job 9	job15	job 6	job 8	job 1	job 5	job 13

Fig. 6 The mutated offspring

In this case $\lambda = 2$ corresponds to the two offspring, therefore, the enlarged sample space will consist of 5 chromosomes available for selection $(m + \lambda = 3 + 2 = 5)$.

Number of tardy jobs is computed by applying (3). Table IV shows the selection fitness value for each chromosome.

TABLE IV
SFF VALUES FOR THE ENLARGED POPULATION POOL

Chromosome	SFF value
Ch1	6
Ch2	6
Ch3	3
Off1	3
Off2	3

Ranking these chromosomes ascending, according to their fitness values, allows using elitist selection strategy for the next generation parents. The resulting parents for the next generation are: Off2, Off1, and Ch3.

To evaluate the effectiveness of using dual-fitness function in GA, the same genetic algorithm described above will be used except that the SFF will be used for both mating selections and survival selections. This GA will be called classical GA to distinguish it from the proposed dual-fitness function GA.

VII. EXPERIMENTATION AND RESULTS

In this section, the experimentation performed is explained. Five problems were generated randomly and the relevant data is summarized in Table V. The five problems were solved using the proposed GA, the classical GA, and the IP.

TABLE V
CHARACTERISTICS OF THE PROBLEMS USED IN THE EXPERIMENTATION

	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
n	10	20	30	50	100
m	3	2	3	4	5
Population size	50	100	150	200	500
# generations	100	200	300	1000	2000

The processing times are generated from the uniform distribution U[1,10] and the due date for job i is computed by equation (4) where k is distributed uniformly according to U[1,7]. All of the jobs have zero ready times and 50 replications were made for each problem.

$$d_i = p_i \times k \tag{4}$$

The experimentation results are summarized in Tables VI, VII and VIII. Table VI shows the average number of tardy jobs for the 50 replications obtained from the classical GA, the proposed GA, and the IP. The results indicate that the proposed algorithm with dual-fitness function outperforms the classical single-fitness function algorithm in all of the problems. In addition, the results obtained by the proposed GA were also compared with IP model results. The results showed that the maximum deviation between the average number of tardy jobs obtained by the proposed GA and the optimal number of tardy jobs obtained by the IP model is 8.0% where this value was 24% for the classical GA.

TABLE VI
RESULTS OF EXPERIMENTATION

Method	Problem 1	Problem 2	Problem 3	Problem 4	Problem 5
Classical GA $(\overline{nT}_{trad})$	4.22	7.44	10.30	14.64	30.14
S.D. Classical	0.764	1.473	1.360	1.382	4.150
Proposed GA $(\overline{nT}_{prop})$	4.10	6.48	9.64	13.92	28.04
S.D. Proposed	0.505	1.092	1.005	1.338	2.740
Improvement% between proposed and classical	2.8%	12.9%	6.4%	7.2%	6.7%
Integer Programming model (nT_{IP})	4	6	9	13	26
Deviation % (Dual/IP)	2.5%	8.0%	7.1%	7.1%	7.9%
Deviation % (Class./IP)	5. 5%	24%	14.4%	12.6%	15.9%

The number of tardy jobs for the 50 replications was used to test the hypotheses that the average number of tardy jobs for the proposed GA (μProp.) is significantly lower than the average number of tardy jobs for the classical GA (μClass.) at significant level of 0.05. Table VII shows the results for the following hypotheses:

TABLE VII
TEST OF HYPOTHESES FOR THE DIFFERENCE BETWEEN THE MEAN NUMBER OF TARDY JOBS FOR CLASSICAL AND PROPOSED GA

Problem	t-Value	Degrees of Freedom	P-value
1	-0.927	85	0.1785
2	-3.701	91	0.0002
3	-2.761	91	0.0035
4	-2.648	98	0.0047
5	-2.986	85	0.0018

At significance level of 0.05, Table VII shows that the mean number of tardy jobs for the proposed GA is significantly lower than the mean number of tardy jobs for the classical GA except for problem 1. This means that the dual-GA is better than the classical GA in terms of quality of solution for large problems.

Table VIII compares the number of replications in which the optimal solution was found for the classical GA and the proposed GA. The results showed that the proposed GA outperformed the classical GA in all problems.

TABLE VIII
FREQUENCY OF OPTIMAL SOLUTION FOUND

Method	Problem 1 1	Problem 2	Problem 3	Problem 4	Problem 5
Classical GA	45	23	24	13	10
Percentage found	90%	46%	48%	26%	20%
Proposed GA	48	41	29	29	22
Percentage found	96%	82%	58%	58%	44%

Table IX compares the average execution time for those runs that reached the optimal solution for the proposed and the classical GA. The results showed that the proposed GA outperformed the classical GA in all problems. In addition, the proposed GA outperformed the IP model in problems 2, 3, 4, and 5.

TABLE IX
AVERAGE EXECUTION TIME IN SECONDS NEEDED TO REACH THE OPTIMAL SOLUTION

IP time	Prop. GA time	Class. GA time
2.3	2.85	2.87
6.19	5.77	6.53
622	537.27	640.50
85561	63210.00	75893.34
181440	101993.728	156414.20

The hypotheses that the average execution time to reach the optimal solution for the proposed GA (time Prop.) is significantly lower than the average execution time to reach the optimal solution for the classical GA (time Class.) at significant level of 0.05 will be tested next. Table X shows the results for the following hypotheses:

TABLE X
TEST OF HYPOTHESES FOR THE EXECUTION TIME

Problem	t-value	Degrees of Freedom	P-value
1	-0.428	88	0.3348
2	-5.347	40	<0.0001
3	-7.370	45	<0.0001
4	-64.460	20	<0.0001
5	-111.384	12	<0.0001

Table X shows that at significance level of 0.05, the average execution time to reach the optimal solution for the proposed GA is lower than the average execution time to reach the optimal solution for the classical GA except for problem 1. Again this shows that at large problems the dual-GA is better than classical GA in terms of execution time.

VIII. DISCUSSION AND CONCLUSIONS

Tables VII and X show that the proposed dual-fitness function genetic algorithm is significantly better than the classical single-fitness function genetic algorithms for large problems.

Because the only difference between the classical single-fitness function genetic algorithm and the dual-fitness function genetic algorithm is in using mating fitness function, we can conclude that using a dual-fitness function can help in enhancing the quality of the solution found, particularly, for large problem sizes as the improvement percentage showed in Table VI. The same result was confirmed using test of hypotheses in Table VII.

Moreover, from Table VI, it is obvious that the deviations between the dual-fitness function genetic algorithm and the integer programming are less compared to the deviations between the classical single-fitness function and the integer programming.

The performance of the classical single-fitness function and the dual-fitness function GA was also measured using the number of trials that the optimal solution was found for each GA. Table VIII shows that the dual-fitness function GA outperformed the single-fitness function GA in all of the problems. In problem 5 the single-fitness function GA was able to find the optimal solution in 10 occasions out of the 50 replications while the proposed GA found it in 22 occasions out of the 50 replications.

In addition, the performance for the proposed GA was measured using the execution time needed to reach the optimal solution. Table IX shows that the average execution time for the dual-fitness function GA outperformed the average execution time for the traditional GA in all problems and the deviation in the execution time increases as the problem size increases. The same result was found using Table X. The execution time for the dual-fitness function GA outperformed the IP model in problems 2, 3, 4, and 5 and the deviation percentage increases as the problem size increased. As for problem 1, the IP model did better than the dual-fitness

function GA. The reason for this was the problem size. The problem size for problem1 was small and thus IP model was able to reach the optimal solution in few sub problems.

In conclusion, the results showed that using dual-fitness function, one for selection and a different one for mating, can improve the quality of the solution and the execution time for the GA especially when the problem size is large. It is believed that this result needs more experimentation on different domains to generalize it. This paper comes as one step on this road and we recommend to apply this idea on different domain problems to see if this result still holds and thus to see if this result can be generalized. The early experimentation results using deterministic identical machine scheduling problem supports this expectation.

REFERENCES

[1] Cheng, T. and Sin, C. 1990. A State-of-the-Art Review of Parallel-Machine Scheduling Research. European Journal of Operation Research, 47: 271-292.
[2] Bedworth, D. and Bailey, J., Integrated Production Control Systems, John Wiley, 1987.
[3] Suer, G., Baez, E. and Czajkiewicz, Z. (1993). Minimizing the number of tardy jobs in identical machine scheduling. Computers and Industrial Engineering. 25(4): 243–246.
[4] Chen, Z and Powell, W. 1999. A column generation based decomposition algorithm for a parallel machine just-in-time scheduling problem. European Journal of Operation Research, 116 (1): 220-232.
[5] Cheng, R. and Gen, M. (1996). Parallel machine scheduling problems using memetic algorithms. IEEE International Conference on Systems, Man, and Cybernetics. 4: 2665-2670.
[6] Liu, M. and Wu, C. (2003). Scheduling algorithm based on evolutionary computing in identical parallel machine production line. Robotics and Computer Integrated Manufacturing. 19: 401–407.
[7] Luu, D.T., Bohez E. L. J., and Techanitisawad A. (2002). A hybrid genetic algorithm for the batch sequencing problem on identical parallel machines. Production Planning and Control. 13(3):43-252.
[8] Min L. and Cheng W. (1999). A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines. Artificial Intelligence in Engineering. 13(4): 399-403.
[9] Suer, G., Pico, F., and Santiago, A. (1997). Identical machine scheduling to minimize the number of tardy jobs when lot-splitting is allowed. Computers and Industrial Engineering. Vol. 33, Nos 1-2, pp. 277-280.
[10] Fariborz J., Rabbani M., Amalnick S., Dabaghi S., Dehghan M., and Yazadn M. 2007. Genetic algorithm for bi-criteria single machine scheduling problem of minimizing maximum earliness and number of tardy jobs. Applied Mathematics and Computation, 194: 552–560.
[11] Johnny C., and Yih-Long H. (1995). Minimizing the number of tardy jobs for m parallel machines. European Journal of Operational Research 84: 343-355
[12] Hui-Yuan F. Guang X., and Shang-Jin W (2000). A dual fitness function genetic algorithm and application in aerodynamic inverse design. Inverse Problems in Engineering 8, 4, pp. 325-344

Saleem Z. Ramadan is an Assistant Professor in the Department of Mechanical and Industrial Engineering at Applied Science Private University, Shafa Badran, Jordan. He received his BE in Industrial Engineering from University of Jordan, Amman; BS in CIS from DeVry University, Chicago, ILL; MS degree in MIS from Keller Graduate School of Management, Chicago, ILL and the PhD degree from the Department of Industrial and Systems Engineering at Ohio University. His research interests are in Bayesian statistics, reliability modeling and prediction, optimization using genetic algorithms, and operations research. He is a member of Jordan Engineering Association since 1998.

Gürsel A. Süer is a Professor in the Industrial and Systems Engineering Department at Ohio University. He has obtained his BSIE and MSIE degrees from Middle East Technical University, Ankara, Turkey and PhD in IE from Wichita State University. He is on the editorial board of various journals. Currently, he serves as the manufacturing area editor of the Computers and Industrial Engineering Journal. He has co-chaired two computers and industrial engineering conferences (1997-Puerto Rico, 2005-Istanbul). He also initiated Group Technology/Cellular Manufacturing Conferences which were held in 2000-Puerto Rico, 2003-Ohio, 2006-Netherlands, 2009-Japan. He has consulted various companies and carried out several funded projects. Most of his research has been motivated by his experiences and observations in industrial settings. His main interests are applied scheduling, manufacturing system design, decision making, genetic algorithms, intelligent systems with human component, supply chain, modeling competitive business environment and vehicle routing. He has edited six conference proceedings and three special issues with different journals. He has published over 110 papers in journals, edited books, conference proceedings and made over 100 technical presentations.