Int J Adv Manuf Technol (2014) 72:439–446
DOI 10.1007/s00170-014-5686-x

ORIGINAL ARTICLE

Simultaneous minimization of total tardiness and waiting time variance on a single machine by genetic algorithms

Maghsoud Amiri · Laya Olfat · Mehdi Keshavarz Ghorabaee

Received: 29 October 2013 / Accepted: 31 January 2014 / Published online: 16 February 2014
© Springer-Verlag London 2014

Abstract This paper considers a single machine scheduling problem, with the objective of minimizing a linear combination of total tardiness and waiting time variance in which the idle time is not allowed. Minimizing total tardiness is always regarded as one of the most significant performance criteria in practical systems to avoid penalty costs of tardiness, and waiting time variance is an important criterion in establishing quality of service (QoS) in many systems. Each of these criteria is known to be non-deterministic polynomial-time hard (NP-hard); therefore, the linear combination of them is NP-hard too. For this problem, we developed a genetic algorithm (GA) by applying its general structure that further improves the initial population, utilizing some of heuristic algorithms. The GA is shown experimentally to perform well by testing on various instances.

Keywords Bicriteria scheduling · Single machine · Genetic algorithms · Total tardiness · Waiting time variance

1 Introduction

In this study, we consider the problem of scheduling a set of n jobs on a single machine in order to minimize total tardiness and waiting time variance simultaneously. In this problem, each of n jobs has a processing time (p_i) and a due date (d_i); furthermore, it is assumed that the machine can only process one job at a time, and the idle time is not allowed.

Scheduling problems in many practical systems are generally considered according to due dates [1]. Managers are regularly faced with the problem of satisfying customer requirements such as delivery dates. Minimizing total tardiness is used as one of the most frequent performance criteria to avoid penalties of late deliveries and increase the responsibility of the system to the customers [2]. Morton et al. [3] and Yoon and Lee [4] proposed constructive heuristics for total tardiness problem. Panneerselvam [5] also developed a simple heuristic for total tardiness problem. Variance minimization criterion was proposed by Merten and Muller [6], and it has received wide attention during the past four decades. Minimizing waiting time variance (WTV) is a well-known problem, which is substantial in providing quality of service (QoS) in many industries that lead to stable and predictable performance [7]. This criterion is often desirable to provide a uniform response to process requests [6]. Eilon and Chowdhury [8] and Ye et al. [7] proposed heuristic algorithms for WTV problem.

Actual scheduling problems may necessitate the decision maker to consider a variety of criteria prior to make any decision. Consequently, many studies have been conducted in multi-criteria scheduling problems. Koksalan et al. [9] considered the bicriteria scheduling problem of minimizing flowtime and maximum earliness on a single machine. Jolai et al. [10] focused on bicriteria scheduling to minimize maximum earliness and number of tardy jobs, and they presented a genetic algorithm in order to solve it. Koksalan and Burak Keha [11] studied two bicriteria scheduling problems: minimizing flowtime and maximum earliness, and minimizing flowtime and number of tardy jobs, and they recommended

M. Amiri · L. Olfat · M. Keshavarz Ghorabaee (✉)
Management and Accounting Faculty, Department of Industrial Management, Allame Tabataba'i University, Dehkadeh-ye-Olympic, Tehran, Iran
e-mail: m.keshavarz_gh@yahoo.com

M. Amiri
e-mail: Amiri@atu.ac.ir

L. Olfat
e-mail: layaolfat@gmail.com

genetic algorithms (GAs) for both of these problems. Hallah [12] used a hybrid of SA and GA to solve the problem of minimization of earliness and tardiness on a single machine and showed the performance of the presented algorithm. Bagchi [13] studied a bicriteria scheduling problem involving completion times and waiting times and presented an efficient algorithm for it. Molaee et al. [14] researched on simultaneous minimization of maximum earliness and number of tardy jobs on a single machine, and developed a heuristic algorithm for it. Hoogeveen [15] provided a comprehensive survey of the multiple-criteria scheduling problems.

This research focuses on minimizing the linear combination of total tardiness and waiting time variance criteria. Du and Leung [16] showed that the total tardiness scheduling problem on a single machine is non-deterministic polynomial-time hard (NP-hard). Moreover, Wieslaw [17] demonstrated that the waiting time variance scheduling problem is NP-hard. It is concluded that the problem of linear combination of total tardiness and waiting time variance is NP-hard. Optimal algorithms for the NP-hard problems would require a computational time that increases exponentially with the size of the problem. Therefore, we developed a genetic algorithm for the problem in this research. We present computational experiments that show the performance of the developed GA.

In section 2, we discuss the bicriteria problem of minimizing total tardiness and waiting time variance. Then in section 3, we delineate the developed GAs and utilized methods and finally in section 4, the performance of GAs is illustrated by computational experiments.

2 Problem description

The problem is to schedule n jobs ($i=1, 2,..., n$) on a single machine with the aim of minimizing total tardiness and waiting time variance. Each job has a processing time (p_i) and a due date (d_i). The following assumptions are considered in the problem:

- The jobs are available at time zero.
- The jobs are independent of each others.
- Each job is processed only once on the machine.
- Job's preemption is not allowed.
- Job's processing times and due dates are known at time zero.

We use the following notations to express the problems:

n Number of jobs
p_i Processing time of the job i
$p_{[j]}$ Processing time of the job located at jth position in sequence and $j= 1,2,...,n$
d_i Due date of the job i
$d_{[j]}$ Due date of the job located at jth position in sequence and $j= 1,2,...,n$
$C_{[j]}$ Completion time of the job located at jth position in sequence and $j= 1,2,...,n$ $C_{[j]}=\sum_{k=1}^{j} p_{[k]}$
$W_{[j]}$ Waiting time of the job located at jth position in sequence and $j= 1,2,...,n$ $W_{[j]}=C_{[j]}-p_{[j]}$ and $W_1=0$
$T_{[j]}$ Tardiness of the job located at jth position in sequence and $j= 1,2,...,n$ $T_{[j]}=\max\{C_{[j]}-d_{[j]},0\}$
$\overline{W}$ Average of total waiting time of sequence $\overline{W} = \left(1/n\right)\sum_{k=1}^{n} W_{[k]}$

We address two criteria of this problem as Z_1 and Z_2. As can be seen in the following, Z_1 denotes the total tardiness criterion and Z_2 denotes the waiting time variance criterion. The purpose of this research is to minimize Z_1 and Z_2 simultaneously.

Z_1 Total tardiness of sequence $Z_1=\sum_{k=1}^{n} T_{[j]}$
Z_2 Waiting time variance of sequence $Z_2 = \left(1/n\right)\sum_{k=1}^{n}\left(W_{[k]}-\overline{W}\right)^2$

3 The genetic algorithm application

The GA is an optimization and search technique based upon the rules of genetics and natural selection. A GA enables a population made up of many individuals to evolve under particular selection rules to a state that maximizes (minimizes) the "fitness" (i.e., minimizes the cost function). The most common form of genetic algorithm involves three types of operators: selection, crossover, and mutation. There are several preferences to be developed concerning the GA application. In this research, we define two levels to present our developed GA: "primary GA" and "secondary GA," and these are described as follows:

3.1 Primary GAs

This level of the developed genetic algorithm includes the main GAs for solving the problem, in which following features and operators are used.

3.1.1 The solution encoding

For the single machine problem which is considered throughout this paper, the natural permutation representation of a solution is a permutation of the integers $1,...,n$, which defines the processing order of n jobs. Each chromosome is represented by such a scheduling solution, i.e., the natural permutation representation of a solution, so as to simplify the solution encoding. For example, for a 10-job problem:

A scheduling solution :	2	1	4	3	6	10	8	9	7	5
A chromosome :	2	1	4	3	6	10	8	9	7	5

3.1.2 The fitness function

The fitness function of the primary GAs in this research is a linear combination of total tardiness and waiting time variance. We define the fitness function as follows:

$$fitness = w\left[\left|Z_1(S)-Z_1^*\right|/\left(1+Z_1^*\right)\right] + (1-w)\left[\left|Z_2(S)-Z_2^*\right|/\left(1+Z_2^*\right)\right], \quad (1)$$

where S represents a chromosome (schedule) in the population, and Z_1^* and Z_2^* are explained in coming sections. We utilize this fitness function in the two following ways:

Dynamic fitness function In this way, large values are assigned to Z_1^* and Z_2^* in the first iteration. When better value is found in the next iterations, the old value is replaced by the better value. Thus, values of Z_1^* and Z_2^* can be updated several times throughout generations.

Static fitness function In this way, we obtain Z_1^* and Z_2^* values by the secondary GAs (S1-GA and S2-GA which are described in section 3.2) before starting primary GA iterations. Next, primary GA begins and values of Z_1^* and Z_2^* do not alter during generations.

3.1.3 The initial population

We use two types of the initial population with the population size 30 (POP=30). The first type is a completely random initial population, and the second type is a heuristic initial population. In the first type, we generate random permutations of jobs to produce the random initial population. In the second type, the population is divided into seven parts (part 1 to part 7) to produce the heuristic initial population. The first six parts (part1 to part 6) are the heuristic parts, and each of them has the size PS=⌊POP/7⌋. We use one heuristic for each part; therefore, we use six heuristics in total. The first and second heuristics are the methods that Eilon and Chowdhury [8] presented for the waiting time variance scheduling problem, and they called "Method 1.1" and "Method 1.2." The third heuristic is the EDD (earliest due date) method that minimizes total tardiness on a condition that, at most, one job has positive tardiness [18]. The fourth heuristic is a constructive method that Yoon and Lee [4] presented for the total tardiness problem and called "Heuristic H1." The fifth heuristic is AU rule that Morton et al. [3] proposed, and the sixth heuristic is a simple method that Panneerselvam [5] presented for minimizing total tardiness. Letting C denotes the counter of chromosomes in a part. The following steps are used to produce chromosomes for each of the first six parts:

1. Obtain the schedule with the heuristic and put it as the first chromosome, C←1
2. Select two positions in the first chromosome
3. Exchange the genes in these positions
4. Put the obtained chromosome at the next place in the part, C←C+1
5. If PS-C=0 then stop, else go to step 2

The last part (part 7) is a random part with the size RS= POP−6PS. We generate random permutations of jobs for part 7.

3.1.4 Parent selection

Selection rate of the developed algorithm is 0.5 (50 %). It could be said, in each iteration, half of the population that has better fitness value, survive for the next iteration (mating pool), and others are discarded and replaced by offsprings. In order to bear an offspring, we should select two chromosomes as parents from the mating pool. We determine both first and second parents by using tournament selection. In the tournament selection, T chromosomes are chosen randomly and the chromosome that has the best fitness value among those becomes the parent. We determined $T=4$ as a good tournament size with better CPU time.

3.1.5 The crossover operator

We utilize two-point crossover with one offspring. For example, we have parents 1 and 2 as (1, 2 , 3, 4, 5, 6, 7, 8, 9, 10) and (5, 2, 10 , 3, 7, 1, 8, 4, 6, 9), respectively, and the crossover points are 3 and 8. The offspring will be born as shown in Fig. 1.

Points "x" and "y" in Fig. 1 represent the crossover points. Crossover points in this research are the function of the number of jobs in problems (n). We use a parameter that controls the crossover points, and k is the symbol of it. It should be noted that k is greater than 2 ($k>2$). We define "x" and "y" as follows:

$$x = \lfloor n/k \rfloor + 1, \quad (2)$$

$$y = n-x, \quad (3)$$

If k increases and moves close to n, then the number of inherited genes of the offspring from parent 1 will be decreased. Similarly, making k close to 2 reduces the number

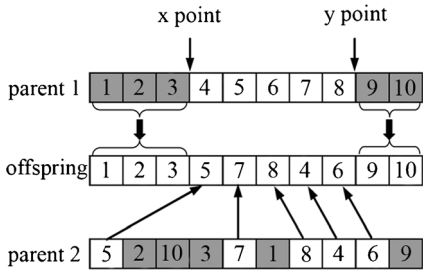

Fig. 1 The crossover operator

of inherited genes from parent 2. We choose k=6 as the best value for k in experimental computations.

3.1.6 The mutation operator

The mutation rate of this algorithm is μ=0.1 that means 10 % of the genes from the population in each iteration are mutated. We randomly select chromosome that we want to be mutated, then two positions are selected randomly along the selected chromosome, and the genes in these positions are exchanged. It should be noted that the best chromosome (chromosome that has better fitness) in each iteration is not mutated due to elitism.

3.1.7 Stopping conditions

In this section, we define two parameters to describe stopping conditions:

a Number of consecutive iterations for which the best chromosome of the population does not change
b Number of total iterations

According to a few basic analyses, we terminate the genetic algorithm either the best chromosome of the population does not change for "*a= 40*" consecutive iterations or after "*b= 100*" iterations in total.

3.1.8 The genetic algorithm

Based on mentioned features and operators, we developed four genetic algorithms as follows:

1. GA-A: This genetic algorithm uses the dynamic fitness function and completely random initial population.
2. GA-B: This genetic algorithm uses the dynamic fitness function and heuristic initial population.
3. GA-C: This genetic algorithm uses the static fitness function and completely random initial population.
4. GA-D: This genetic algorithm uses the static fitness function and heuristic initial population.

Letting ITR and NC denote the iteration number and the number of consecutive iterations for which the best chromosome does not change, respectively. Steps of GA-A and GA-B can be summarized as follows:

1. Create initial population. Let ITR←1, NC←0.
2. Assign large values to Z_1^* and Z_2^*
3. Evaluate Z_1 for each chromosome of the population and, if $Z_1<Z_1^*$ then $Z_1^*\leftarrow Z_1$
4. Evaluate Z_2 for each chromosome of the population and, if $Z_2<Z_2^*$ then $Z_2^*\leftarrow Z_2$
5. Evaluate fitness for each chromosome of the population.
6. Select two parents by tournament selection.
7. Apply crossover and replace discarded chromosomes of the population with the generated offsprings.
8. Apply mutation.
9. If the best fitness of iteration ITR is equal to that of iteration ITR-1, then NC←NC+1, else NC←0.
10. ITR←ITR+1
11. If ITR>b or NC≥a, then stop, else go to step 3.

And following steps are used for GA-C and GA-D:

1. Create initial population. Let ITR←1, NC←0.
2. Obtain values of Z_1^* and Z_2^* by secondary GAs (S1-GA and S2-GA which are described in section 3.2).
3. Evaluate fitness for each chromosome of the population.
4. Select two parents by tournament selection.
5. Apply crossover and replace discarded chromosomes of the population with the generated offsprings.
6. Apply mutation.
7. If the best fitness of iteration ITR is equal to that of iteration ITR-1, then NC←NC+1, else NC←0.
8. ITR←ITR+1
9. If ITR>b or NC≥a, then stop, else go to step 3.

3.2 Secondary GAs

This level includes S1-GA and S2-GA which are used to obtain Z_1^* and Z_2^*, respectively, in static fitness function mode. These genetic algorithms are very similar to the primary GAs. However, differences between them are listed below:

1. The initial population: the population size of secondary GAs is half of the primary GA population size (POP= 15). We divide the population into two parts (part 1 and part 2) to produce the initial population. The first part (part 1) is the heuristic part and has the size PS=⌊POP/7⌋. The second part (part 2) is the random part and has the size RS= POP−PS. We use the same steps of section 3.1.3 to produce chromosomes of the first part (part 1). "Heuristic H1" and "Method 1.2" are used for S1-GA and S2-GA, respectively. We produce random permutations of jobs for part 2.

Table 1 Due date ranges of problem sets

Problem set	Due date range
I	[0.0*P*, 0.4*P*]
II	[0.1*P*, 0.3*P*]
III	[0.25*P*, 0.45*P*]
IV	[0.3*P*, 1.3*P*]

2. The fitness function: total tardiness is the fitness function of S1-GA and waiting time variance is the fitness function of S2-GA.
3. Stopping condition: we terminate S1-GA and S2-GA if the best chromosome of the population does not change for "*a*= *20*" consecutive iterations or after "*b*= *50*" iterations in total.

Selection, crossover and mutation operators of these GAs are the same as primary GAs. The algorithm which we used for S1-GA and S2-GA is as follows:

1. Create initial population. Let ITR←1, NC←0.
2. Evaluate fitness for each chromosome of the population.
3. Select two parents by tournament selection.
4. Apply crossover and replace discarded chromosomes of the population with the generated offsprings.
5. Apply mutation.
6. If the best fitness of iteration ITR is equal to that of iteration ITR-1, then NC←NC+1, else NC←0.
7. ITR←ITR+1
8. If ITR>b or NC≥a, then stop, else go to step 2.

4 Computational results

We carried out experiments using randomly generated problems to assess the performance of the developed GAs. Processing times of problems were generated from the discrete

Table 2 Average deviation for 10 replications of CAT-1

n	Problem set	*w*=0.3				*w*=0.5				*w*=0.7			
		GA-A	GA-B	GA-C	GA-D	GA-A	GA-B	GA-C	GA-D	GA-A	GA-B	GA-C	GA-D
	Low processing time												
5	I	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0607	0.0000	0.0000	0.0639	0.0000
	II	0.0289	0.0289	0.0537	0.0537	0.0000	0.0000	0.0247	0.0000	0.0001	0.0001	0.0205	0.0929
	III	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0943	0.0200	0.0707	0.0000	0.0707	0.0000
	IV	0.0000	0.0000	0.0529	0.0972	0.0000	0.0000	0.1047	0.0190	0.0001	0.0001	0.0795	0.0878
10	I	0.4934	0.3007	0.1273	0.1090	0.0693	0.3698	0.0426	0.1638	0.0534	0.1010	0.1840	0.3076
	II	0.0124	0.0137	0.0829	0.3161	0.0202	0.0000	0.4998	0.0219	0.1060	0.0000	0.3775	0.0000
	III	1.2113	0.2449	0.1494	0.0252	0.3921	0.5306	0.8494	0.7854	0.1070	0.2517	0.5981	0.6096
	IV	0.0668	0.5000	0.6542	0.6283	0.3571	0.5000	0.5985	0.5003	0.3459	0.7470	0.7295	0.5000
15	I	0.4359	0.1930	0.6707	0.1408	5.0416	0.0295	7.2318	0.8612	0.5186	0.5543	1.6046	0.5220
	II	0.4528	0.1123	0.5443	0.1909	0.8262	0.2812	0.6323	0.4387	1.3996	0.3643	1.1824	0.4504
	III	0.2112	0.0904	0.3201	0.0755	0.3489	0.0955	0.8579	0.3085	0.6585	0.4468	1.8587	0.7992
	IV	0.6943	0.5624	1.2228	0.3398	0.7859	0.2152	1.1692	0.4910	0.6731	0.4300	1.0427	0.5477
Average		0.3006	0.1705	0.3232	0.1647	0.6534	0.1685	1.0088	0.3059	0.3278	0.2413	0.6510	0.3264
	High processing time												
5	I	0.0000	0.0000	0.0283	0.0135	0.0000	0.0000	0.0540	0.0346	0.0000	0.0000	0.0000	0.0000
	II	0.0550	0.0550	0.0648	0.1155	0.0107	0.0107	0.0107	0.0107	0.0000	0.0000	0.0001	0.0000
	III	0.0225	0.0225	0.0308	0.0279	0.0000	0.0000	0.0169	0.0062	0.0000	0.0000	0.0000	0.0000
	IV	0.0001	0.0001	0.0248	0.0001	0.0002	0.0002	0.0075	0.0001	0.0001	0.0001	0.0001	0.0001
10	I	0.0508	0.0835	0.3136	0.2226	0.0569	0.0709	0.3099	0.2977	0.4548	0.0148	0.2689	0.0148
	II	0.6452	0.6435	0.0000	0.0167	0.0872	0.0324	0.4102	0.0324	0.3102	0.3598	1.1127	0.6772
	III	0.1224	0.0000	0.1751	0.1365	0.1836	0.1996	0.9364	0.4130	0.1861	0.4506	0.6884	0.4979
	IV	0.2100	0.0000	0.0988	0.0292	0.5542	0.0002	0.6946	0.0203	0.2662	0.0274	0.1946	0.0000
15	I	0.7258	0.4033	0.7230	0.3227	0.3341	0.1610	0.6456	0.2002	0.9438	0.1300	0.7887	0.0768
	II	0.6209	0.3955	1.3122	0.3580	0.8426	0.2115	0.7514	0.2403	1.2046	0.3556	1.7619	0.9021
	III	0.3342	0.0982	0.2125	0.0813	0.3045	0.0306	0.5778	0.1150	0.5195	0.2336	1.0585	0.4470
	IV	1.0172	0.4228	0.9327	0.1697	0.6179	0.3287	1.2176	0.5296	0.6896	0.6331	1.1719	0.4458
Average		0.3170	0.1770	0.3264	0.1245	0.2493	0.0872	0.4694	0.1583	0.3812	0.1838	0.5872	0.2551

Table 3 Average deviation for 10 replications of CAT-2

n	Problem set	w=0.3				w=0.5				w=0.7			
		GA-A	GA-B	GA-C	GA-D	GA-A	GA-B	GA-C	GA-D	GA-A	GA-B	GA-C	GA-D
	Low processing time												
20	I	1.3396	0.2683	1.1070	0.2129	0.8208	0.2700	0.9633	0.4108	1.6908	0.1787	1.7597	0.1695
	II	1.0262	0.3825	1.1634	0.3209	0.6965	0.1290	0.7829	0.2994	1.7881	0.3488	1.3915	0.9997
	III	0.2581	0.0916	0.3317	0.0847	0.4583	0.1354	0.3765	0.0794	0.7567	0.0060	0.8850	0.3401
	IV	0.6697	0.8238	0.9823	0.5778	1.1486	0.0936	1.0660	0.4423	0.4955	0.2819	1.1586	0.2359
50	I	2.8374	0.0857	2.6100	0.2016	4.0312	0.1785	3.7418	0.0878	9.0609	0.0541	8.3800	0.0031
	II	3.0802	0.1956	3.3615	0.0807	3.1351	0.1770	3.6849	0.3226	7.7713	0.1822	7.5583	1.0536
	III	8.3372	0.1833	7.4454	0.8784	9.3234	0.1710	8.7701	1.6697	10.7723	0.2011	12.4626	0.7186
	IV	9.2752	0.9521	9.3519	1.1074	7.2650	0.7793	9.1899	0.7281	6.6366	0.7600	9.1246	1.0642
100	I	6.7527	0.1306	6.0638	0.0826	7.5046	0.0853	8.2023	0.0775	15.0430	0.0028	14.8075	0.0602
	II	6.9867	0.0823	6.9647	0.0882	7.1315	0.1349	6.6497	0.0794	13.4371	0.0409	14.6170	0.1014
	III	8.8226	0.3163	9.5934	0.2008	12.6793	0.2449	12.7736	0.7433	15.1369	0.1742	16.3595	0.8028
	IV	13.3649	0.5571	11.5751	0.4501	12.8504	0.2907	11.6826	0.9631	12.1645	0.3751	15.9035	0.3339
Average		5.2292	0.3391	5.0459	0.3572	5.5871	0.2241	5.6570	0.4920	7.8961	0.2172	8.7007	0.4903
	High processing time												
20	I	1.1400	0.5184	0.5441	0.2973	0.8701	0.2196	1.0522	0.0690	2.9972	0.2126	1.8686	0.0376
	II	0.7563	0.2604	1.3478	0.1755	0.8246	0.1894	0.6922	0.3561	1.6736	0.3309	2.3991	0.5152
	III	0.3249	0.1250	0.3677	0.1155	0.6600	0.0217	0.5459	0.1376	0.5634	0.1275	1.1383	0.4329
	IV	3.3498	0.5309	1.2739	1.0959	1.1241	0.4613	0.9249	0.6245	1.4878	0.3916	2.3641	0.5899
50	I	3.6334	0.1727	2.8457	0.2517	4.0599	0.1803	3.3548	0.1507	9.0815	0.0293	7.5107	0.0103
	II	3.3457	0.1314	2.7266	0.1547	4.0495	0.1865	4.3928	0.1423	8.2248	0.0110	7.8472	0.2532
	III	6.6457	0.8500	6.3789	0.0654	9.5799	0.2802	9.6658	0.7189	11.9338	0.3035	11.7440	1.3162
	IV	11.1039	0.1369	5.9775	1.1401	9.2860	0.9786	7.5011	1.0030	10.6825	0.6302	9.5386	1.9013
100	I	6.3481	0.1578	5.9670	0.1652	7.2581	0.3000	7.2450	0.0313	12.6918	0.0134	12.1979	0.0047
	II	6.5732	0.0618	6.9566	0.1488	6.4570	0.1841	6.4790	0.0474	10.9099	0.0664	12.1113	0.1234
	III	10.9300	0.1855	11.3794	0.2595	13.4912	0.2824	12.6521	0.8499	17.6231	0.3687	16.4503	0.9553
	IV	10.7974	0.3181	10.1538	0.1124	9.4985	0.7334	8.9271	0.7158	11.9430	0.4078	12.2828	0.1679
Average		5.4124	0.2874	4.6599	0.3318	5.5966	0.3348	5.2861	0.4039	8.3177	0.2411	8.1211	0.5257

uniform distributions in two ranges of low and high processing time as in Koksalan et al. [9]. The range [1–25] represents low processing time variability, and the range [1–100] shows high processing time variability. Due dates were generated using discrete uniform distributions in four sets given in Table 1 as in Koktener and Koksalan [19]. In this table, each set represents a range and P denotes the sum of processing times of the jobs.

We tried six problem sizes n=5, 10, 15, 20, 50, and 100 with three weights w=0.3, 0.5, and 0.7. These problem sizes were divided into two categories. First category (CAT-1) represents problems relating to sizes n=5, 10, and 15. We compared GAs result deviations from global optimal results of LINGO 8 in the first category. Second category (CAT-2) shows problems relating to sizes n=20, 50, and 100. We compared GAs result deviations from the best results in the second category. With regard to different levels of factors such as processing times (high and low), number of jobs, different GAs, different weights (w=0.3, 0.5, and 0.7), and due date ranges, respectively, we tried $2\times3\times4\times3\times4$ class of problems. We randomly made 10 problems for each class. Therefore, 2,880 problems were solved in each category. Note that all GAs were coded in MATLAB, and all computations were carried out on a computer having a 2.1-GHz processor.

4.1 CAT-1

In this category, we considered problem sizes n=5, 10, and 15 and we compared deviations from global optimal results which were obtained by the LINGO 8 global solver. For this purpose, we minimized the following function in the LINGO:

$$f(Z_1,Z_2)=w\left[\left|Z_1-Z^*_{1\mathrm{opt}}\right|/\left(1+Z^*_{1\mathrm{opt}}\right)\right]+(1-w)\left[\left|Z_2-Z^*_{2\mathrm{opt}}\right|/\left(1+Z^*_{2\mathrm{opt}}\right)\right], \quad (4)$$

where Z^*_{1opt} and Z^*_{2opt} represent optimal values for total tardiness and waiting time variance problems, respectively, and these were obtained by LINGO 8 for each replication, separately.

In order to be able to assess the deviation of GA results from optimal results of LINGO, we scaled the results. For this aim, we put Z_1 and Z_2, which were obtained from GAs (GA-A, GA-B, GA-C, and GA-D) for each problem in the objective function of LINGO that was mentioned above, and we obtained values of $f(Z_1,Z_2)$ for each GA. Next, we obtained deviations from the global optimal value of LINGO with the following deviation measure:

$$\%\ \text{deviation} = 100 \times \left[\frac{\left(f - f_{\text{opt}}\right)}{f_{\text{opt}}}\right], \tag{5}$$

where f denotes the scaled result of the GAs, and f_{opt} represents the global optimum which were obtained by LINGO for each replication.

The average percent of deviation is shown in Table 2. From this table, it can be seen that the performance of GA-D for w= 0.3 is better than other algorithms both in low and high processing time. However, for w=0.5 and w=0.7, GA-B performed better than other algorithms.

4.2 CAT-2

This category includes problem sizes n=20, 50, and 100, and we compared deviations from the best results of each replication. In order to be able to evaluate the deviation of GA results from best results, we scaled the results. For this purpose, we put Z_1 and Z_2, which were obtained from GAs (GA-A, GA-B, GA-C, and GA-D) for each problem in the following function, and we obtained values of $f(Z_1,Z_2)$ for each GA:

$$f(Z_1,Z_2) = w\left[\left|Z_1 - Z^*_{1\text{GA}}\right| / \left(1 + Z^*_{1\text{GA}}\right)\right] + (1-w)\left[\left|Z_2 - Z^*_{2\text{GA}}\right| / \left(1 + Z^*_{2\text{GA}}\right)\right], \tag{6}$$

where $Z^*_{1\text{GA}}$ and $Z^*_{2\text{GA}}$ represent near optimal values for total tardiness and waiting time variance problems, respectively, and these were obtained by using genetic algorithms like the secondary GAs which are illustrated in section 3.2 with the stopping condition parameters: a= *80* and b= *200*. We compared the performance of the GAs using the following deviation measure:

$$\%\ \text{deviation} = 100 \times \left[\left(f - f_{\text{best}}\right) / f_{\text{best}}\right], \tag{7}$$

where f represents the scaled result of the GAs, and f_{best} shows best scaled result of each replication.

The average percent of deviation is shown in Table 3. With respect to Table 3, we can say that the performance of GA-B is relatively better than other GAs in both low and high processing time.

5 Conclusion

In this research, we concentrated on a single machine bicriteria scheduling problem. We considered the problem of minimizing a linear combination of total tardiness and waiting time variance criteria. Then, we developed genetic algorithms for solving this bicriteria problem since it was an NP-hard problem. Two types of heuristic and random initial population and two distinct fitness functions were applied to genetic algorithms. Consequently, four genetic algorithms presented with regard to initial population and fitness function. In order to show the efficiency of proposed GAs, we solved different problems in two categories. We concluded that heuristic initial population and dynamic fitness function increase the performance of genetic algorithm for this bicriteria scheduling problem.

References

1. Omar MK, Teo SC (2006) Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: an improved MIP approach. Appl Math Comput 181(2):1008–1017
2. Holsenback JE, Russell RM (1992) A heuristic algorithm for sequencing on one machine to minimize total tardiness. J Oper Res Soc 43(1):53–62
3. Morton TE, Rachamadugu RM, Vepsalainen A (1984) Accurate myopic heuristics for tardiness scheduling. GSIA Working Paper, Carnegie-Mellon University, Pittsburgh, PA.
4. Yoon SH, Lee IS (2011) New constructive heuristics for the total weighted tardiness problem. J Oper Res Soc 62(1):232–237
5. Panneerselvam R (2006) Simple heuristic to minimize total tardiness in a single machine scheduling problem. Int J Adv Manuf Technol 30(7):722–726
6. Merten AG, Muller ME (1972) Variance minimization in single machine sequencing problems. Manag Sci 18(9):518–528
7. Ye N, Li X, Farley T, Xu X (2007) Job scheduling methods for reducing waiting time variance. Comput Oper Res 34(10):3069–3083
8. Eilon S, Chowdhury IG (1977) Minimising waiting time variance in the single machine problem. Manag Sci 23(6):567–575
9. Koksalan M, Azizoglu M, Kondakci SK (1998) Minimizing flowtime and maximum earliness on a single machine. IIE Trans 30(2):192–200
10. Jolai F, Rabbani M, Amalnick S, Dabaghi A, Dehghan M, Parast MY (2007) Genetic algorithm for bi-criteria single machine scheduling problem of minimizing maximum earliness and number of tardy jobs. Appl Math Comput 194(2):552–560
11. Köksalan M, Burak Keha A (2003) Using genetic algorithms for single-machine bicriteria scheduling problems. Eur J Oper Res 145(3):543–556
12. M'Hallah R (2007) Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. Comput Oper Res 34(10): 3126–3142
13. Bagchi U (1989) Simultaneous minimization of mean and variation of flow time and waiting time in single machine systems. Oper Res 37(1):118–125

14. Molaee E, Moslehi G, Reisi M (2010) Minimizing maximum earliness and number of tardy jobs in the single machine scheduling problem. Comput Math Appl 60(11):2909–2919
15. Han H (2005) Multicriteria scheduling. Eur J Oper Res 167(3):592–623
16. Du J, Leung JYT (1990) Minimizing total tardiness on one machine is NP-hard. Math Oper Res 15(3):483–495
17. Wieslaw K (1993) Completion time variance minimization on a single machine is difficult. Oper Res Lett 14(1):49–59
18. Hamilton E (1969) One-machine sequencing to minimize certain functions of job tardiness. Oper Res 17(4):701–715
19. Karasakal EK, Köksalan M (2000) A simulated annealing approach to bicriteria scheduling problems on a single machine. J Heuristics 6(3):311–327