APPLICATION OF GENETIC ALGORITHM TO A BICRITERIA SINGLE MACHINE SCHEDULING PROBLEM OF MINIMIZING MAXIMUM EARLINESS AND NUMBER OF TARDY JOBS

Maryam Dehghanbaghi, Azadeh Dabbaghi, Fariborz Jolai, Mohammad Yazdani

Department Of Industrial Engineering, University Of Tehran,
P.O. Box 11365-4563, Iran

dehghanbaghi@yahoo.com (Maryam Dehghanbaghi)

Abstract

Today's importance of on time deliveries according to JIT concept makes managers to consider a desired set of extra "due date related " criteria in the workshop scheduling. These criteria can be stated as tardiness, number of tardy jobs, and earliness. In bicriteria scheduling problems one criterion can stands for the manufacturer concerns and the other can represent customer concerns. In this paper we focus on the bicriteria scheduling problem of minimizing the number of tardy jobs and maximum earliness for single machine, in which the idle time is not allowed. The problem is known to be NP-hard. This problem has not been solved by any meta heuristic. Thus, we developed a genetic algorithm (GA) by exploiting its general structure that further improves the initial population, utilizing a heuristic algorithm on the initial population. We present a computational experiment, considering the real conditions of industrial systems based on generating stochastic processing times and due dates in different intervals to test the algorithm for both high and low processing time that shows the out performance of the improved GA by comparing the results with a pair wise interchange method, applied on the two famous sequences (Moore and MST). Our effort is also to minimize both two criteria simultaneously.

Keywords: Genetic algorithm, Bicriteria scheduling.

Presenting Author's biography

Maryam Dehghanbaghi. Received her BSc degree in textile engineering in 2002. She is currently a MSc. student in industrial Engineering Department at University of Tehran and a research Assistant and PhD candidate. Her research interests contain, decision making, performance measurement, data mining, simulation, scheduling and outsourcing.

1 Introduction

In this paper we consider a bicriteria scheduling problem on single machine: minimizing number of tardy jobs and maximum earliness. We developed a genetic algorithm to solve the problem. Single machine scheduling problem can be generally described as follows:

A set of n jobs has to be scheduled on one machine and each job has a processing time (pi), and due date (di) in addition, the machine can only process one job at a time [2].

Although, single machine scheduling is one of the simplest problems, there are several reasons that the majority of the literature on earliness and tardiness scheduling deals with problems with only a single machine: ***First***, this problem can be regarded as a base for learning scheduling concepts which leads to facilitate modeling of complex systems. ***Second***, analysis and determination of the specifications of bottlenecks in multi stage production lines can be conducted through single machine concepts.

According to just in time (JIT) approach production managers should consider more than one criterion in scheduling problems .Just in time requires only the necessary units to be provided with the necessary quantities at the necessary times. Producing one extra unit is as bad as one unit short [7].

While both early and tardy completions are not desirable, one criterion can represent manufacturer concerns and the other can represent customer concerns. Minimizing maximum earliness-the first criterion-leads to reduce finished goods inventory for manufacturer; n_T (number of tardy jobs) can affect on customers dissatisfactory regarding the tardiness.

Many researchers have been working on single machine bicriteria scheduling problems. Hoogveenm, J. A. [6] studied a number of bicriteria scheduling problems.

GA has been applied to various forms of earliness and tardiness criteria in the past.

Koksalan, M. and Burak Keha, A. [8] used a GA to solve two bicriteria scheduling problems: minimizing flow time and maximum earliness, and minimizing flow time and number of tardy jobs.

Hallah, R.M. [5] applied hybrid of SA and GA to solve minimize total earliness and tardiness on single machine and tested the capability of the proposed algorithm.

Azizoglu, M. et al. [1] proposed a heuristic algorithm to solve the bicriteria scheduling problem of minimizing the maximum earliness and number of tardy jobs on single machine by developing a general procedure for generating all efficient schedules and finding the best one. Since this problem has not been solved by any meta heuristic, we developed a GA to solve this problem and compared the results with a constructive heuristic, applied on this problem, obtained by [3].

The organization of this paper is as follows: in section 2 we define the problem then in section 3 we describe the utilized method. The computational results are presented in section 4.

2 Problem description

We consider the problem of scheduling a set of n jobs on a single machine with the following assumption:

- Set up times are considered as a part of its processing times.
- The jobs are ready at time zero in a static scheduling environment
- There are no precedence relationships between jobs and preemption is not allowed.
- Machine idle time is not allowed.
- The maximum earliness criterion is a non Regular performance measure, that is, it may be beneficial to insert machine idle times between the processing of some jobs. It is of course, possible to force E_{max} to zero by inserting sufficient machine idle time. However, there could be strong reasons against keeping the machine idle [1].

The following notations are used to describe the scheduling problems:

n number of jobs;
P_i processing time of job i;
d_i due date of job i ;
J_i job located in i_{th} position in sequence;
S_i slack time of i_{th} job. $S_i = d_i - p_i$;
C_i completion time of job i;
E_i earliness of i_{th} job. $E_i = d_i - C_i$;

Given a schedule S, our criteria is:

$E_{max} = \max \{E_i\}$

N_T: number of tardy jobs. $N_T = \sum \alpha_i(S)$

Where $\alpha_i(S) = \begin{cases} 1 & \text{if } C_i(S) > d_i \\ 0 & \text{if } C_i(S) <= d_i \end{cases}$

Definition: A schedule S is set to be efficient if there does not exist another schedule S′ satisfying:

$$E_{max}(S') <= E_{max}(S) \quad \text{and} \quad n_T(S') <= n_T(S).$$

Lee, C. Y. and Vairaktarakis, G. L.[9] showed that solving this problem is NP-hard. Minimizing both E_{max} and $_{nT}$ problems can be solved optimally in polynomial time when taken individually. it is evident that N_T is minimized by using Moore's algorithm and E_{max} is minimized by MST (minimum slack time) order [10].

However NP-hard problems can be solved by several heuristic and Meta-Heuristic methods (GA, SA…), we develop a genetic algorithm because there were not any paper to solve this specific problem with GA or any other Meta-Heuristic methods.

3 GA application

GA is a flexible, intelligent probabilistic search algorithm. Genetic algorithm operates by maintaining and modifying the characteristics of a set of trial solutions over a number of solutions. Each individual solution is represented by a string include a set of random numbers. GA is capable of retaining certain desirable characteristics that may be ignored by completely random searches.

GA applies genetic operations on the individuals (chromosomes or parents) chromosomes are reproduced by exchanging genes utilizing cross over .an offspring inherit some characteristics from each of its parents .the offspring face mutation in which some of genes in random selected chromosomes, alters to make it adaptable to the environment. The offspring can be replaced with some or all the previous population through calculating fitness function for each new generated chromosome. The general steps of GA are as follows:

1. Generate random numbers for initial population. (chromosomes)
2. Evaluate fitness function of each chromosome.
3. Repeat POP times:
   - Select two parents from population
   - Apply cross over to them to produce new off springs
   - Enter the new offspring to the population utilizing replacement strategy
   - select a chromosome from the population
   - apply mutation to the random selected chromosomes
   - enter the mutated chromosomes to the population using replacement strategy.
   - Mutation and crossover apply with the probability of p_m and p_c.

For scheduling problem a chromosome includes genes that are number of jobs (in single machine) that should be applied on one machine with permutation.
There are several ways to be made in all the stages of GA. in this article the applied GA is as follows:

3.1 Initial population

In this problem the population size is 30. We first generate random schedules for the first population; we then utilize a heuristic algorithm on each chromosome of the initial population. The procedure of the applied heuristic includes 7 steps:

1. Search the whole random schedule and find the Place of E_{max}. (P_{Emax})
2. Search the whole random schedule and find the Place of fist tardy job (P_{nT}).
3. If P_{Emax} is greater than P_{nT} go to step 5
4. If P_{Emax} is lower than P_{nT} go to step 6
5. Assign the jobs from the place of the first tardy job to the end of the schedule by ascending due dates order.
6. Assign the jobs from the place of the job, to the place of the first tardy job by descending due dates order. go to the step 7
7. Enter the acquired schedule to the initial population.

3.2 The fitness function

In different bicriteria scheduling problems different fitness functions are used to evaluate the solutions. As in Azizoglu,M., et al.[1] , we utilize the following linear fitness function:

$$F = w\,[E_{max}(S) - E_{max}(MST)] / [E_{max}(moore) - E_{max}(MST)] + (1-w)\,[n_T(S) - n_T(moore)] / [n_T(MST) - n_T(moore)]$$

Where w is the weight representing the relative importance of the criteria. Note that the fitness function is to be minimized in comparison with traditional maximization fitness functions.

3.3 Parent selection

The selection strategy is a procedure to choose the individuals in the current population for creating offspring of subsequent generation. For this mean, there are a number of efficient procedures in the literature.

In the most selection strategies, the solutions with better fitness have more chance to be selected as parents for creating offspring of subsequent generation. Rolette wheel sampling (RWS) is one of the most common strategies in which each individual is assigned a slice of a circular ‘‘rolette wheel,’’ the size of the slice being proportional to the individual’s fitness. The wheel is spun Pop-Size (population size) times, where Pop-Size is the number of individuals in the population. On each spin, the individual under the wheel’s marker is selected to be in the pool of parents for the next generation. This method can be implemented as follows:

1. Let F be the sum of the fitness values of all solutions in current population as follows:

$$F = \sum_{i=1}^{pop-size} (f_i)$$

Where f_i is the fitness values of solution i and Pop-Size is equal to the number of chromosomes in the population.

2. Let Pi be the relative probability related to chromosome i as follows:

$$Pi = \frac{fi}{F} \qquad k = 1,2,\ldots,\text{pop-size}$$

3. Let qi be the cumulative probability related to chromosome i as follows:

$$q_k = \sum_{j=1}^{k} Pj \qquad k = 1,2,\ldots,\text{pop-size}$$

4. Generate a random number such as r in the range of [0, 1]. If r < q1, then the first chromosome selected.

Otherwise, the i^{th} chromosome is selected where $q_{i-1} < r < q_i$. ($2 \le i \le pop - size$)

The fitness of each solution is obtained by the objective function directly. The initial population is randomly created in terms of a continuous uniform distribution [11].

3.4 The crossover operator

We test three types of crossover: one point, two point and cycle crossover. The best one recognized as cycle crossover. The probability of cross over is 0.6 (p_c=0.6) Cycle crossover method is discussed by Goldberg [4]. In this method, the information exchange begins at the left of the spring and the first two digits are exchanged. We illustrate this method by one example.

Cycle Crossover (Initial Parents)

$Offspring_{x1}$ [3 4 6 2 1 5]
$Offspring_{x2}$ [4 1 5 3 2 6]

By exchanging 3 and 4 we have:

Cycle Crossover (First Step)

$Offspring_{x1}$ [4 4 6 2 1 5]
$Offspring_{x2}$ [3 1 5 3 2 6]

There are two 4s in the first off spring. We exchange the second positions in each parent:

Cycle Crossover (Second Step)

$Offspring_{x1}$ [4 **1** 6 2 1 5]
$Offspring_{x2}$ [3 **4** 5 3 2 6]

Now there are two 1s in the first off spring, so we exchange position 5 with second offspring.

Cycle Crossover (Third Step)

$Offspring_{y1}$ [4 **1** 6 2 **2 5**]
$Offspring_{y2}$ [3 **4** 5 3 **1** 6]

The next position to exchange is position 4 where there is a repeated 2.

Cycle Crossover (Fourth Step)

$Offspring_{z1}$ [4 **1** 6 **3 2** 5]
$Offsprin_{z2}$ [3 **4** 5 **2 1** 6]

At this point, we have exchanged the 2 for the 3 and are back to our starting point, so the crossover is complete .We see that each offspring has exactly one of each digit, and it is in the position of one of the parents.

3.5 The mutation operator

We randomly choose a chromosome and two of its genes as a candidate to be mutated. We change the value of chosen genes with each other, with probability of 0.4(p_m).

3.6 Replacement strategy

The elitist strategy is applied in this problem. In each generation we select the best chromosome as one of the chromosomes in the next population, in addition to the chromosomes (parents and off springs) which are ranked after cross over and mutation according to their fitness values. Among these ranked chromosomes, 29 of the best chromosomes select as the next population.

3.7 Stopping conditions

GA is terminated if the best chromosome of the population does not change for 50 consecutive generations or after 100 generations in total, which ever comes first.

4 Computational experiments

We conduct experiments utilizing randomly generated problems to evaluate the performance of the proposed algorithm. We generate the processing time of each job from a discrete uniform distribution in the range of [1-30] for high processing time and in the range of [1-10] for low processing time. Then we generate due dates from a discrete uniform distribution in the range of [0- $\rho \sum p_i$] .we use three different values for ρ to create three set of problems that are as :0.4,0.6 and 0.8.

We tried three problem sizes n= 50,100and 150. thus we have 2*3*4*3*3 class of problems corresponding to different levels of factors such as problem sets(high and low processing time), number of jobs, different GAs, different weights and due dates, respectively. And also 2*3*3*3 classes to make the solutions of Hmst and Hmoore (table1).

For each class we randomly generate 10 problems therefore 2700 problems are solved in total.

In this paper we concentrate on improvement of GA solutions through utilizing the described heuristic in section 3.1. In order to achieve it, four different genetic algorithms are presented as follows, in which the only difference is in their initial population, the results of these GAs are then compared with a constructive heuristic method applied on Moore and MST sequences (Hmoore & Hmst).

1. GA1: The GA described in section 3 with randomly generated chromosomes.

2. GA2: The GA1 that the heuristic algorithm is applied on its initial population.

3. GA3: The GA1 that contains Moore and MST schedules in its initial population.(28 random schedule + Moore + MST)

4. GA4: The GA2 that contains Moore and MST schedules in its initial population. (28 random schedule that the heuristic procedure in section 3.1 is applied on + Moore + MST)

5. Hmoore & Hmst: a pair wise interchange method* is applied on Moore and MST sequences.

Tab. 1 Average deviation over 10 replication.

		w=0.1						w=0.5						w=0.9					
n	ρ	GA1	GA2	GA3	GA4	Hmoore	Hmst	GA1	GA2	GA3	GA4	Hmoore	Hmst	GA1	GA2	GA3	GA4	Hmoore	Hmst
a. Low processing time																			
	0.4	5.96	5.23	0.103	0.089	0.079	10.13	0.8842	0.4233	0.0752	0.0805	0.3052	0.4263	3.97	0.21	0.045	0.009	7.22	0.066
50	0.6	5.74	5.95	0.105	0.128	0.052	12.55	1.21	0.77	0.070	0.060	0.282	0.410	5.26	3.20	1.115	0.031	7.60	0.050
	0.8	9.03	8.00	0.286	0.155	0.057	11.56	3.95	2.22	0.262	0.019	0.218	0.390	30.46	13.87	0.092	0.028	8.30	0.209
	0.4	9.10	9.54	0.057	0.041	0.014	10.53	3.84	0.56	0.192	0.213	0.118	0.251	20.31	0.97	0.245	0.011	7.24	0.036
100	0.6	8.44	7.91	0.059	0.097	0.034	8.48	5.28	2.56	0.091	0.043	0.170	0.232	37.06	17.92	0.037	0.022	7.68	0.083
	0.8	10.97	8.92	0.182	0.193	0.001	9.25	5.39	2.18	0.068	0.041	0.162	0.147	47.04	29.32	0.050	0.042	7.94	0.043
	0.4	11.95	11.75	0.314	0.092	0.328	12.76	3.97	1.47	0.124	0.094	0.191	0.298	34.59	9.75	4.873	0.025	7.44	0.026
150	0.6	11.68	9.08	0.077	0.063	0.067	9.13	5.48	3.96	0.085	0.020	0.161	0.236	43.86	28.19	0.007	0.023	7.57	0.039
	0.8	16.34	14.85	0.168	0.075	0.115	9.35	10.93	5.98	0.219	0.049	0.078	0.155	112.97	65.88	0.340	0.018	9.30	0.240
average		**9.91**	**9.03**	**0.150**	**0.104**	**0.083**	**10.41**	**4.54**	**2.23**	**0.132**	**0.069**	**0.187**	**0.283**	**37.28**	**18.81**	**0.756**	**0.023**	**7.81**	**0.088**
b. High processing time																			
	0.4	6.73	6.40	2.779	0.268	0.164	12.02	0.78	0.23	0.066	0.056	0.052	0.287	3.29	0.08	1.098	0.005	5.89	0.023
50	0.6	5.19	5.08	0.078	0.007	0.161	9.49	1.51	0.72	0.111	0.075	0.245	0.261	13.98	4.35	0.245	0.179	8.39	0.047
	0.8	7.29	6.25	0.150	0.118	0.022	8.02	4.08	1.46	0.298	0.168	0.289	0.343	27.93	8.35	0.015	0.034	7.45	0.103
	0.4	7.12	7.12	0.048	0.019	0.019	8.74	2.65	1.27	0.066	0.021	0.130	0.251	18.38	7.21	0.023	0.008	6.81	0.025
100	0.6	8.08	8.46	0.098	0.088	0.017	9.19	5.19	1.88	0.079	0.015	0.084	0.250	34.84	9.05	0.283	0.009	6.66	0.014
	0.8	12.12	9.19	0.091	0.077	0.009	8.39	6.90	3.10	0.043	0.083	0.036	0.157	59.64	23.48	0.066	0.066	7.54	0.053
	0.4	14.55	8.33	0.528	0.123	0.129	9.66	3.57	1.62	0.048	0.046	0.197	0.292	30.63	8.81	0.003	0.000	7.27	0.000
150	0.6	10.91	9.42	0.088	0.024	0.076	9.04	5.53	1.77	0.089	0.038	0.138	0.229	38.73	12.62	3.946	0.000	7.30	0.008
	0.8	14.51	10.52	0.104	0.078	0.106	8.28	6.11	4.70	0.103	0.032	0.104	0.136	60.28	36.64	0.084	0.063	8.15	0.061
average		**9.61**	**7.86**	**0.440**	**0.089**	**0.078**	**9.20**	**4.04**	**1.86**	**0.100**	**0.059**	**0.141**	**0.245**	**31.97**	**12.29**	**0.640**	**0.040**	**7.27**	**0.037**

**note that the efficiency of the heuristic applied on initial population can easily be seen through comparing GA1 and GA2.th best in each weight has the minimum average value both in high and low processing time.

.

ISBN 978-3-901608-32-2

Tab. 2 Maximum deviation over 10 replication.

		w=0.1						w=0.5						w=0.9					
n	ρ	GA1	GA2	GA3	GA4	Hmoore	Hmst	GA1	GA2	GA3	GA4	Hmoore	Hmst	GA1	GA2	GA3	GA4	Hmoore	Hmst
a. Low processing time																			
	0.4	8.88	8.43	0.587	0.587	0.418	13.29	1.63	1.200	0.400	0.200	0.760	1.000	8.11	1.08	0.12	0.042	8.11	0.124
50	0.6	8.09	8.82	0.364	0.364	0.400	16.60	3.16	2.84	0.237	0.360	0.600	1.000	14.93	9.44	8.67	0.087	8.45	0.099
	0.8	14.25	14.00	0.538	0.700	0.231	20.50	9.73	5.24	1.091	0.152	0.333	0.643	49.81	28.19	0.43	0.222	11.35	0.471
	0.4	13.71	11.58	0.155	0.155	0.060	14.52	7.88	2.81	1.071	1.308	0.216	0.389	30.53	4.94	1.80	0.043	8.47	0.064
100	0.6	11.59	10.01	0.176	0.176	0.138	9.59	11.05	4.72	0.256	0.278	0.344	0.389	54.32	45.38	0.09	0.082	8.13	0.176
	0.8	23.50	17.72	0.936	1.128	0.010	16.66	9.24	5.95	0.143	0.116	0.371	0.400	76.11	69.20	0.25	0.293	10.47	0.205
	0.4	16.02	17.72	0.724	0.507	0.830	18.15	6.17	5.51	0.257	0.471	0.324	0.471	41.10	32.72	8.74	0.200	9.64	0.111
150	0.6	31.17	15.96	0.211	0.263	0.362	11.32	12.47	8.72	0.351	0.064	0.333	0.389	75.40	58.50	0.05	0.183	8.15	0.220
	0.8	30.46	30.01	0.464	0.266	0.679	14.36	18.93	16.50	0.515	0.136	0.303	0.394	341.89	207.68	1.63	0.075	20.58	1.289
average		**17.52**	**14.92**	**0.462**	**0.461**	**0.347**	**15.00**	**8.92**	**5.95**	**0.480**	**0.343**	**0.398**	**0.564**	**76.91**	**50.79**	**2.42**	**0.136**	**10.37**	**0.306**
b. High processing time																			
	0.4	13.75	11.50	13.41	1.500	0.471	21.50	1.34	0.33	0.316	0.154	0.263	0.429	9.82	0.56	8.60	0.042	7.75	0.042
50	0.6	8.26	7.55	0.23	0.054	0.613	13.52	1.94	1.88	0.235	0.250	0.353	0.471	41.16	12.90	1.62	1.432	20.62	0.190
	0.8	11.00	10.00	0.56	0.563	0.113	12.75	6.79	3.75	0.923	0.923	0.731	0.857	44.95	28.50	0.12	0.149	9.00	0.351
	0.4	8.75	8.67	0.23	0.101	0.148	9.86	3.70	3.24	0.400	0.071	0.257	0.429	25.20	17.48	0.09	0.042	8.13	0.087
100	0.6	11.92	10.25	0.25	0.200	0.115	10.13	11.03	7.20	0.257	0.073	0.263	0.371	68.49	53.08	2.16	0.031	7.87	0.031
	0.8	24.68	16.42	0.19	0.163	0.075	9.47	11.32	10.97	0.184	0.136	0.103	0.289	101.89	80.10	0.39	0.250	9.28	0.163
	0.4	51.44	10.82	3.61	0.155	0.299	11.00	6.70	4.59	0.130	0.220	0.471	0.588	37.37	32.70	0.02	0.000	8.00	0.000
150	0.6	21.00	12.78	0.20	0.111	0.203	10.27	9.42	4.72	0.220	0.189	0.297	0.351	66.80	36.76	5.60	0.000	7.88	0.020
	0.8	39.91	14.52	0.49	0.313	0.448	9.15	8.67	7.23	0.316	0.067	0.205	0.316	77.33	62.27	0.28	0.250	9.38	0.282
average		**21.19**	**11.39**	**2.13**	**0.351**	**0.276**	**11.96**	**6.77**	**4.88**	**0.331**	**0.231**	**0.327**	**0.456**	**52.56**	**36.04**	**2.10**	**0.244**	**9.77**	**0.130**

**This table confirms the final results achieved from table 1

ISBN 978-3-901608-32-2

in which the jobs are interchanged two by two up to the end of the schedule, for each fitness function values is calculated and the minimum is selected.

***pair wise interchange** is a method that a near optimal schedule is obtained through suitable pair wise interchange between tasks [3].

We compare the performances of the above mentioned GA1, GA2, GA3, GA4, Hmst and Hmoore using the following deviation measure:

$$\% \text{ deviation} = 100 \times \frac{F - best}{best}$$

Where F is the fitness function value (presented in section 3.2) of GA1, GA2, GA3, GA4, Hmst and Hmoore for each replication.

Best is the minimum value of F among each set of instances (replication).

The average percent of deviation is calculated for each class of problems as shown in table 1.

In table 2 the maximum of the above mentioned deviations are reported.

With respect to Table 1, for w=0.1 , w=0.5 Hmoore and GA4 outperform than other solutions as they obtained the minimum average value of deviations both in high processing time and low processing time, respectively.

In w=0.9 GA4 is preferred as the best solution in high processing time, but for low processing time Hmst is recognized as the best one.

The quality of the results in table 1 are obviously confirmed through table 2 by comparing the average of maximum deviations for the preferred solutions.

5 Conclusion

In this study we first addressed the single machine scheduling problem. As evaluation measures we considered maximum earliness and number of tardy jobs. Then we proposed a GA for solving such problem in which a heuristic algorithm is applied on the randomly generated initial population. The computational results for solving different problems show the performance of the proposed algorithm. In order to show the efficiency well, we utilized pair wise interchange, On MST and Moore sequences and then compared with the results achieved by different GAs and for simulating the system based on the real industrial systems we utilize the uniform distribution for processing time of jobs. Consequently the results show the best solution for different weights of objective function.

6 References

[1] M. Azizoglu, S. kondakci, M. Koksalan, Single machine scheduling with maximum earliness and number tardy, Computers and Industrial Engineering 45 (2003) 257–268.

[2] Kenneth R. Baker, Introduction to sequencing and scheduling, Wiley, New York, 1974.

[3] S.M.T. Fatemi Ghomi, F. Jolai, A pair wise interchange algorithm for parallel machine scheduling, Journal of Production Planning and Control, 9: 685–689, 1998.

[4] David.E. Goldberg, Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, 1989.

[5] R.M. Hallah, Minimizing total earliness and tardiness on a single machine using hybrid heuristic, Computers and Operations research, 34: 3126–3142, 2007.

[6] J.A. Hoogeveen, Single machine bi-criteria scheduling, unpublished PhD thesis, CWI, Amsterdam, 1992.

[7] M.K. Omar, C. Siew, Minimizing the sum of earliness/tardiness in identical parallel machines schedule with incompatible job families: An improved MIP approach, Applied Mathematics and Computation, 181: 1008-1017, 2006.

[8] M. Koksalan, A. Burak Keha, Using genetic algorithms for single machine bi-criteria scheduling problems, European Journal of Operations Research, 145: 543–556, 2003.

[9] C.Y. Lee, G.L. Vairaktarakis, Complexity of single machine hierarchical scheduling: a survey, Complexity in Numerical Optimization, 19: 269–298, 1993.

[10] J.M. Moore, A n job one machine sequencing algorithm for minimizing the number of late jobs, Management Science, 15: 102–109, 1968.

[11] N. Safaei, S.J. Sadjadi, M. Babakhani, An efficient genetic algorithm for determining the optimal price discrimination, Applied Mathematics and Computation, 181: 1693–1702, 2006.

