A Genetic Algorithm for the Single Machine Total Weighted Tardiness Problem

N. Liu, Mohamed A. Abdelrahman, and
Department of Electrical and Computer Eng.,
Tennessee Technological University,
Cookeville, TN 38505, USA
nl8239@tntech.edu, mabdelrahman@tntech.edu

Srini Ramaswamy
Department of Computer Science,
Tennessee Technological University,
Cookeville, TN 38505, USA
srini@acm.org

Abstract—Scheduling problems are important NP-hard problems. Genetic algorithms can provide good solutions for such optimization problems. In this paper, we present a genetic algorithm to solve the single machine total weighted tardiness scheduling problem, which is a strong NP-hard problem. The algorithm uses the natural permutation representation of a chromosome, heuristic dispatching rules combined with random method to create the initial population, position-based crossover and order-based mutation operators, and the best members of the population during generations. The computational results of problem examples with 10 and 25 jobs and general problems with 50, 100, 200 and 500 jobs show the good performance and the efficiency of the developed algorithm.

I. INTRODUCTION

The single machine total weighted tardiness problem is to schedule n jobs on a single machine to minimize the sum of the weighted tardiness of all the jobs. Consider n jobs to be processed without interruption on a single machine that can handle only one job at a time. Each job j, available for processing at time zero, has a positive processing time p_j, a positive weight w_j, and a positive due date d_j. For a given sequence of jobs, the tardiness of job j is defined as $T_j = \max \{0, C_j - d_j\}$, where C_j is the completion time of job j. The objective of the total weighted tardiness problem is to find a processing order of all the jobs, i.e., a schedule that minimizes the sum of the weighted tardiness $\sum_{i=1}^{n} w_i T_i$ of all jobs.

For arbitrary positive weights, the single machine total weighted tardiness problem is strongly NP-hard [1]. Many scheduling problems, which do not have efficient, so-called polynomial time, optimal algorithms, are the so-called NP-hard problems, i.e., efficient optimal algorithms are unlikely to exist for these problems. An algorithm is referred to as a polynomial time algorithm when the number of iterations in the algorithm is polynomial in the size (n, the number of jobs) of the problem. Consider the various permutations of the n jobs of the total weighted tardiness problem to find the optimal schedule. Even for a modest-sized problem, complete enumeration is not computationally feasible since it requires the evaluation of n! sequences (e.g., a 20-job problem requires the evaluation of more than $2.4 * 10^{18}$ sequences).

As single machine scheduling problems can provide help and insight into resolving, understanding, managing, and modeling more complex multi-machine scheduling problems, the single machine total weighted tardiness problem has received much attention in literature. The problem has been tackled by enumerative algorithms: branch and bound algorithms [2, 3, 4, 5, 6] and dynamic programming algorithms [7] to generate exact solutions, i.e., solutions that are guaranteed to be optimal. But the branch and bound algorithms are limited by computational times and the dynamic programming algorithms are limited by computer storage requirements, especially when the number of jobs is more than about 50. Thereafter, the problem has been extensively studied by heuristics, i.e., solution procedures that generate good or even optimal solutions, but do not guarantee optimality.

These heuristics include heuristic dispatching rules and local search heuristics. As there is no one best dispatch rule for all problem environments, in other words, dispatching rules do not consistently provide good quality solutions [9], in recent years, much attention has been devoted to local search heuristics [8, 9, 10, 11, 12, 13, 14]. These local search heuristics mainly include neighborhood search methods (e.g., decent methods, simulated annealing, threshold accepting, and tabu search) [8, 9, 12, 13] and genetic algorithms (GA) [12, 14].

Paper [12] compares the performance of a number of local search heuristics that have the binary representation, namely, decent methods, simulated annealing, threshold accepting, tabu search, and GA, for total weighted tardiness problems with the number of jobs n = 40, n = 50 and n = 100. It indicates that its binary encoded GA performs very well and requires comparatively little computation time; it is a viable alternative to other heuristic methods, especially in view of its small maximum relative deviations and modest computation time; its multistart version (i.e., running independently the GA several times with different initial populations) improves upon the single-start version and has the added advantage of reliability due to its non-sensitivity to the random number seed. Good solutions are also obtained in a short time by a natural permutation encoded GA and its multistart version for the problems with 40 and 50 jobs in paper [14]. Multistart version of the GA tends to offer progressive improvements of solutions with moderate additional computation time.

In this paper, we present a new genetic algorithm for solving the single machine total weighted tardiness problem. We represent each chromosome by the natural permutation representation of a solution, i.e., a sequence of jobs defining a scheduling solution, use scheduling heuristic dispatching rules combined with random method to create the initial population, use position-based crossover and order-based mutation operators, and keep the best members of the population during generations. The algorithm has been applied to two 10-job problems, and one 25-job problem to show optimal solutions obtained. It has also been applied to solve the problems with

50, 100, 200, and 500 jobs. Their computational results are compared with those of decent methods (DES and DESO) to show the improvement from the best of either EDD or WSPT.

The paper is organized as follows. Section II describes the genetic algorithm for solving the single machine total weighted tardiness problem. Section III reports the computational results of the algorithm. Section IV summarizes the main conclusions.

II. GENETIC ALGORITHM

Genetic Algorithms (GAs) were originally proposed by John H. Holland [15]. They are search algorithms that explore a solution space and mimic the biological evolution process. There are many GA implementations successfully applied to a great variety of problems [16, 17].

The main components of a genetic algorithm are as follows [18]:

1. A chromosomal representation of solutions (solution encoding).
2. Creation of an initial population of chromosomes (initial population).
3. Measurement of chromosome fitness based on the objective function (fitness).
4. Natural selection of some chromosomes, i.e., parents, in the population for generating new members, i.e., children, in the population (selection).
5. Genetic operators applied to these chromosomes whose role is to create new members, i.e., children, in the population by crossing the genes of two chromosomes (crossover operators) or by modifying the genes of one chromosome (mutation operators) (genetic operators).
6. Natural selection of the members of the population who will survive (replacement).
7. Natural convergence of the whole population that is globally improved at each step of the algorithm (parameter selection).

The performance of a GA depends largely on the design of the above components and the choice of parameters such as population size, probabilities of genetic operators (i.e., crossover rate and mutation rate), and number of generations.

A. Solution Encoding

For the single machine total weighted tardiness problem, 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.

B. Initial Population

In order to approximate an optimal solution as near as possible, the initial population of chromosomes is created by scheduling heuristic dispatching rules, combined with random method. The dispatching rules are as follows:

1. EDD (Earliest Due Date): The next job scheduled is a job that has the earliest due date among the jobs that are not scheduled yet. The resulting sequence is the same as the sequence of jobs arranged in the ascending order of their due dates.
2. WSPT (Weighted Shortest Processing Time): Calculate ratio $S_i = P_i / W_i$. Jobs are scheduled in the ascending order of the ratios.
3. SPT (Shortest Processing Time): Jobs are scheduled in the ascending order of their processing time.
4. BWF (Biggest Weight First): Jobs are scheduled in the descending order of their weights.
5. AU (Apparent Urgency) [19]: Calculate apparent urgency priority AU_j. Jobs are arranged in the descending order of their apparent urgency priorities.

$$AU_j = \left(\frac{w_j}{p_j}\right)\exp\left(\frac{-\max\{0,\ d_j - t - p_j\}}{k \cdot \bar{p}}\right),$$

Here, k is called the look-ahead parameter and is set according to the tightness of the due date; $\bar{p}$ is the average processing time; t is the current time, but for static problems, here t = 0. This heuristic has the same time complexity as EDD and WSPT.

C. Fitness

When a population is generated, each chromosome is evaluated and its fitness is calculated as follows. The total weighted tardiness is computed for each chromosome. Then chromosomes are sorted by the increasing values of their total weighted tardiness. Finally the first chromosome is assigned its fitness with the value of population size minus one. The fitness of a chromosome is assigned the value of the fitness of its forward adjacent chromosome minus one.

D. Selection

By selection methods, chromosomes (parents) are selected from the population for combining to produce new chromosomes (children), i.e., for applying genetic operators. Here, we use random selection method that selects parents randomly.

E. Genetic Operators

1) Crossover: The role of a crossover operator is to combine elements from two parent chromosomes to generate one or more child chromosomes. Here, we use position-based crossover. This randomly chooses 0.5*n (n is the number of jobs) positions, and the characters, i.e., the genes, in these positions, are kept unchanged in the offspring.

2) Mutation: The role of a mutation operator is to provide and maintain diversity in a population so that other operators can continue to work. Here, we use order-based mutation. Two

positions are selected randomly, and two characters, i.e., two genes, in these positions, are interchanged.

We choose position-based crossover and order-based mutation due to their good performance obtained in paper [20].

F. Replacement

This selection is based on elitism. That is to keep the best chromosomes of the current population and their offspring. They will form a new population to survive into the next generation.

G. Parameter Selection

For choosing suitable values of parameters such as population size, crossover rate, and mutation rate, we apply the position-based crossover operator to N/2 pairs of chromosomes selected randomly, where N is the population size. The mutation probability, which determines the mutation rate, is set to $1/\lambda$, where λ is the length of a chromosome. Population size and generation size are dependent on the problem size.

III. COMPUTATIONAL RESULTS

Two 10-job problems, and one 25-job problem are selected as computational examples to show optimal solutions obtained when applying the genetic algorithm. The genetic algorithm is also applied to solve general problems with 50, 100, 200, and 500 jobs to show the performance obtained and the efficiency of the algorithm.

A. Computational Examples

Example 1: Table I presents a 10-job example taken from paper [2] that has an optimal solution 1-2-3-5-4-6-8-9-7-10 with total weighted tardiness 27. Our computational results are the same.

TABLE I
10-JOB PROBLEM OF EXAMPLE 1

I	p_i	d_i	w_i
1	4	3	3
2	1	4	1
3	2	7	4
4	4	8	2
5	1	11	3
6	4	15	5
7	2	16	1
8	2	20	5
9	3	20	3
10	2	25	10

Example 2: Table II presents another 10-job example taken from another reference [21]. The total weighted tardiness of heuristic dispatching rules EDD, WSPT, SPT, BWF, and AU (k = 2) are 496, 383, 535, 355, and 230 respectively. Our computational results are 3-1-8-4-5-9-7-6-10-2 with total weighted tardiness 218.

TABLE II
10-JOB PROBLEM OF EXAMPLE 2

i	p_i	d_i	w_i
1	8	26	4
2	12	28	1
3	6	32	6
4	10	35	5
5	3	38	1
6	11	48	4
7	9	50	5
8	11	51	9
9	13	53	8
10	7	64	1

Example 3: Table III presents one 25-job example taken from paper [11] that has the results 7-17-10-4-6-21-2-12-24-13-1-9-25-3-5-23-8-18-14-15-22-16-19-11-20 with the total weighted tardiness 14,930. Our computation results are 5-9-17-10-4-6-21-2-12-24-13-1-25-7-3-23-8-18-14-15-22-16-19-11-20 with the total weighted tardiness 14,410.

TABLE III
25-JOB PROBLEM OF EXAMPLE 3

Job number	Process times	Due dates	Weights
1	71	595	7
2	8	477	7
3	24	768	7
4	35	259	10
5	25	171	2
6	76	321	7
7	92	157	6
8	63	898	3
9	71	201	6
10	56	218	6
11	69	730	1
12	97	484	9
13	25	567	10
14	85	1055	9
15	93	1058	7
16	78	559	2
17	60	197	7
18	21	965	3
19	84	490	2
20	94	334	1
21	58	352	7
22	92	428	4
23	97	686	6
24	26	493	5
25	56	670	4

B. General Problems

The general problems are generated as follows:

1. Each job j is assigned an integer processing time p_j in the uniform distribution (1, 100), an integer weight w_j in the uniform distribution (1, 10).
2. The difficulty of each problem depends on the range of due dates (RDD) and on the tightness factor of due dates (TF). RDD and TF values are selected from the set {0.2, 0.4, 0.6, 0.8, 1.0} respectively.
3. Each job j is assigned an integer due date d_j in the uniform distribution (P*(1-TF-RDD/2), P*(1-TF+RDD/2)), where $P = \sum_{j=1}^{n} p_j$. When P*(1-TF-

RDD/2) is less than or equal to 0, d_j is in the uniform distribution (1, P*(1-TF+RDD/2)).

4. For each value of n, i.e. 50, 100, 200, and 500, five problems have been generated for each of the 25 pairs of values of RDD and TF. Thus a total of 500 problems have been generated.

As these problems have more than 50 jobs, the optimal solutions are not available. Therefore computational results have been compared with these of decent methods, i.e., DES and DESO, to show the percentage improvement from the best result of either EDD or WSPT.

Decent Methods: These are pairwise interchange methods, including DES and DESO. They begin with the sequence obtained by applying the AU dispatching rule and exchange jobs (u, v) from (1, 2) to (1, 3), (1, 4), …, and (n-1, n), where (u, v) means to exchange the job in the *u*th position with the job in the *v*th position.

DES: This is the strict descent method that only pairwise job exchanges yielding a decrease in objective function, here i.e., the total weighted tardiness, are accepted.

DESO (Descent Method with Zero Interchanges): In this method, pairwise job exchanges yielding no change in objective function are also accepted, in addition to exchanges yielding a decrease in objective function. This is because both jobs may be early and remain so even after the exchange.

Percentage Improvement: The percentage improvement is calculated by the following equation:

$$\%Imp = \frac{\min(T_{EDD}, T_{WSPT}) - T_H}{\min(T_{EDD}, T_{WSPT})} * 100\%$$

where T_{EDD} is the total weighted tardiness obtained after applying EDD rule, T_{WSPT} is the total weighted tardiness obtained after applying WSPT rule, and T_H is the total weighted tardiness obtained after applying a certain heuristic method. Here, the heuristic methods are DES, DESO, and the genetic algorithm.

Tables IV and V present the computational results for the problems with 50, 100, 200, and 500 jobs respectively when applying the GA, which are compared with these of DES and DESO, where "-" means the total weighted tardiness of the problem is zero when applying EDD or WSPT, AU, DES, DESO, or GA respectively. For AU, the lookahead parameter is k = 0.5, 0.9, 2.0, 2.0, and 2.0 for TF = 0.2, 0.4, 0.6, 0.8, and 1.0 respectively. The GA is coded in C language and the computational tests are carried out on a DELL Dimension 8200 PC.

The analysis of the computational results is as follows:

1. Here, for the same problem size, the population and generation sizes of the GA are the same.
2. For the same problem size, different problems have different percentage improvements and computation times. The GA has the best improvements but needs more computation times. Here, for the worst case, the computation times for problems with 50, 100, 200, and 500 jobs are 20 seconds, 45 seconds, 2 minutes 30 seconds, and 12 minutes respectively. The computation times are reasonable price to pay for the improvement in performance.
3. When the problem size increases, the percentage improvement of decent methods declines. However, the percentage improvement of the GA can be stable or better if we enlarge its population size and/or generation size. But there is a trade-off between the improvement in performance and the computation times.

IV. CONCLUSIONS

In this paper we propose a new genetic algorithm for solving the single machine total weighted tardiness scheduling problem. The algorithm is based on the natural permutation representation of a chromosome, the combination of dispatching rules and random method to create the initial population, position-based crossover and order-based mutation, and elitism. Computational results show that the solutions of the GA are better than those of heuristic dispatching rules and decent methods, but it needs more computation time. However, its computation time is reasonable for good solutions. Therefore the GA can obtain good solutions in an efficient way.

V. ACKNOWLEDGEMENT

This study was supported by Manufacturing Research Center of Tennessee Technological University.

VI. REFERENCES

[1] J. K. Lenstra, A. H. G. Kinnooy Kan, and P. Brucker, "Complexity of machine scheduling problems," *Annals of Discrete Mathematics*, vol. 1, pp. 343-362, 1977.

[2] Joel Shwimer, "On the n-job, one-machine, sequence-independent scheduling problem with tardiness penalties: a branch-bound solution," *Management Science*, vol. 18, no. 6, pp. B-301-B-313, February 1972.

[3] H. G. Rinnooy Kan, B. J. Lageweg, J. K. Lenstra, "Minimizing total costs in one-machine scheduling," *Operations Research*, vol. 23, no. 5, pp. 908-927, September-October 1975.

[4] Marshall L. Fisher, "A dual algorithm for the one-machine scheduling problem," *Mathematical Programming*, vol. 11, pp. 229-251, 1976.

[5] Jean-Claude Picard and Maurice Queyranne, "The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling," *Operations Research*, vol. 26, no. 1, pp. 86-110, January-February 1978.

[6] Chris N. Potts and Luk N. Van Wassenhove, "A branch and bound algorithm for the total weighted tardiness problems," *Operations Research*, vol. 33, no. 2, pp. 363-377, March-April 1985.

[7] L. Schrage and K. R. Baker, "Dynamic programming solution of sequencing problem with precedence constraints," *Operations Research*, vol. 26, pp. 444-449, 1978.

[8] Hirofumi Matsuo, Chang Juck Suh, and Robert S. Sullivan, "A controlled search simulated annealing method for the single machine weighted tardiness problem," *Annals of Operations Research*, vol. 21, pp. 85-108, 1989.

[9] C. N. Potts and L. N. Van Wassenhove, "Single machine tardiness sequencing heuristics," *IIE Transactions*, vol. 23, no. 4, pp. 346-354, December 1991.

[10] Dileep R. Sule, "Heuristic method for a single machine scheduling problem," *International Journal of Industrial Engineering*, vol. 1, no. 2, pp. 167-174, June 1994.

[11] Peter A. Huegler and Francis J. Vasko, "A performance comparison of heuristics for the total weighted tardiness problem," *Computers & Industrial Engineering*, vol. 32, no. 4, pp. 753-767, 1997.

[12] H. A. J. Crauwels, C. N. Potts, and L. N. Van Wassenhove, "Local search heuristics for the single machine total weighted tardiness scheduling problem," *INFORMS Journal on Computing*, vol. 10, no. 3, pp. 341-350, Summer 1998.

[13] Ana Martia Madureira, "Meta-heuristics for the single-machine scheduling total weighted tardiness problem," *Proceedings of the 1999 IEEE International Symposium on Assembly and Task Planning*, Porto, Portugal, July 1999, pp. 405-410.

[14] Ana Madureira, Carlos Ramos, and Silvio do Carmo Silva, "A GA based scheduling system for dynamic single machine problem," *Proceedings of the 4th IEEE International Symposium on Assembly and Task Planning*, Soft Research Park, Fukuoka, Japan, May 2001, pp. 262-267.

[15] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, University of Michigan Press, 1975.

[16] Lawrence Davis, eds., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991.

[17] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.

[18] F. Alexandre, C. Cardeira, F. Charpillet, Z. Mammeri, and M. –C. Portmann, "Compu-search methodologies II: scheduling using genetic algorithms and artificial neural networks," in A. Artiba and S. E. Elmaghraby, eds., *The Planning and Scheduling of Production Systems*, Chapman & Hall, 1997, pp. 300-335.

[19] T. E. Morton, R. M. Rachamadugu, and A. Vepsalainen, "Accurate myopic heuristics for tardiness scheduling," *GSIA Working Paper No. 36-83-84*, Carnegie-Mellon University, PA, 1984.

[20] Gilbert Syswerda, "Schedule optimization using genetic algorithms," in Lawrence Davis, eds., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, 1991, pp. 332-349.

[21] Daniel Sipper and Robert L. Bulfin, Jr., *Production: Planning, Control, and Integration*, McGraw-Hill, 1997, p. 399.

TABLE IV
COMPUTATIONAL RESULTS FOR GENERAL PROBLEMS WITH 50 AND 100 JOBS

		50 Jobs			100 Jobs		
RDD	TF	DES %Imp (%)	DESO %Imp (%)	GA %Imp (%)	DES %Imp (%)	DESO %Imp (%)	GA %Imp (%)
0.2	0.2	39.36	39.22	49.10	49.83	46.55	59.73
	0.4	12.62	13.60	27.29	5.54	6.85	26.29
	0.6	3.89	3.81	12.25	0.40	1.05	12.17
	0.8	-1.22	-1.24	6.89	-2.39	-2.38	6.76
	1.0	-1.60	-1.61	1.06	-3.71	-3.70	1.06
0.4	0.2	19.96	22.93	22.46	17.64	17.64	17.64
	0.4	44.30	43.94	60.23	45.79	47.48	60.89
	0.6	19.41	19.80	34.52	13.79	13.07	33.20
	0.8	0.207	0.207	14.10	0.31	0.32	12.93
	1.0	-1.19	-1.19	4.13	-3.97	-4.04	4.21
0.6	0.2	-	-	-	-	-	-
	0.4	52.23	52.67	69.64	49.14	49.70	69.85
	0.6	34.80	34.26	56.18	29.13	30.10	55.39
	0.8	12.58	12.59	28.47	6.12	6.13	21.82
	1.0	0.68	0.64	7.99	-2.91	-2.90	8.30
0.8	0.2	-	-	-	-	-	-
	0.4	46.21	47.77	57.01	57.06	50.03	56.32
	0.6	41.53	41.74	59.09	44.99	45.24	65.30
	0.8	20.64	20.59	42.72	15.29	15.42	35.61
	1.0	3.45	3.47	15.88	1.29	1.28	15.07
1.0	0.2	-	-	-	-	-	-
	0.4	-	-	-	-	-	-
	0.6	52.16	51.54	66.84	56.43	57.22	74.12
	0.8	26.92	27.42	44.78	27.70	27.59	49.10
	1.0	9.41	9.18	26.75	6.28	6.30	22.21
Average		20.78	21.02	33.68	19.70	19.47	33.71

TABLE V
COMPUTATIONAL RESULTS FOR GENERAL PROBLEMS WITH 200 AND 500 JOBS

		200 Jobs			500 Jobs		
RDD	TF	DES %Imp (%)	DESO %Imp (%)	GA %Imp (%)	DES %Imp (%)	DESO %Imp (%)	GA %Imp (%)
0.2	0.2	38.91	42.61	57.17	49.16	49.55	58.26
	0.4	6.38	6.77	27.31	4.66	3.92	22.63
	0.6	-1.07	-1.04	13.51	-1.72	-1.84	11.07
	0.8	-3.65	-3.67	5.79	-2.94	-3.55	5.90
	1.0	-5.55	-5.56	0.98	-5.09	-5.09	0.90
0.4	0.2	27.54	28.24	30.92	21.74	14.86	43.44
	0.4	33.67	34.85	58.04	35.89	35.39	60.25
	0.6	13.11	13.31	33.63	11.12	10.77	31.49
	0.8	0.50	0.49	13.37	0.01	0.01	11.52
	1.0	-4.12	-4.12	3.56	-5.21	-5.21	3.06
0.6	0.2	-	-	-	-	-	-
	0.4	51.68	54.92	76.04	50.10	55.54	76.23
	0.6	24.15	24.09	48.48	19.81	19.73	44.84
	0.8	5.10	5.04	22.58	3.33	3.33	17.77
	1.0	-3.23	-3.23	7.38	-3.71	-3.71	6.49
0.8	0.2	-	-	-	-	-	-
	0.4	55.06	49.87	59.31	39.84	45.60	68.73
	0.6	36.30	36.57	61.42	25.32	25.33	46.50
	0.8	13.63	13.54	33.75	15.18	11.25	29.95
	1.0	-0.43	-0.43	13.88	-0.68	-0.68	11.64
1.0	0.2	-	-	-	-	-	-
	0.4	-	-	-	-	-	-
	0.6	50.90	51.57	70.64	38.52	38.43	61.26
	0.8	24.17	24.34	48.47	15.35	15.32	37.04
	1.0	3.56	3.55	21.79	2.11	2.11	18.03
Average		17.46	17.70	33.72	14.89	14.81	31.76