International Journal of Industrial Engineering and Management (IJIEM), Vol. 3 No 3, 2012, pp. 163-171
Available online at www.iim.ftn.uns.ac.rs/ijiem_journal.php
ISSN 2217-2661

UDK 519.67

A Genetic Algorithm Approach for Minimizing Total Tardiness in Single Machine Scheduling

Gürsel A. Süer
Industrial and Systems Engineering, Ohio University, Athens, OH 45701, USA
Xiaozhe Yang
Industrial and Systems Engineering, Ohio University, Athens, OH 45701, USA
Omar I. Alhawari
Industrial and Systems Engineering, Ohio University, Athens, OH 45701, USA
Joel Santos
Electrical and Computer Engineering, University of Puerto Rico – Mayagüez, Mayagüez, Puerto Rico 00681, USA
Ramon Vazquez
Electrical and Computer Engineering, University of Puerto Rico – Mayagüez, Mayagüez, Puerto Rico 00681, USA

Received (9 Jul 2012); Revised (22 September 2012); Accepted (30 September 2012)

Abstract
Minimizing total tardiness in single machine scheduling is known as NP-hard. In this paper, the problem is extended to include non-zero ready times and the preemption of jobs is not allowed. First, a mathematical model is developed. Due to computational complexities with the mathematical model, a Genetic Algorithm approach is also proposed and later its performance is compared with optimal solutions. The results show that GA can find optimal solution for small problems and near optimal solutions for large problems. The results also show that among Delay-only, Non-delay-only, and Random strategies, Non-delay strategy produced more robust solutions whereas random strategy found the optimal solution in smaller problem categories.

Key words: *Single Machine Scheduling, Tardiness, Genetic Algorithms, Mathematical Modeling.*

1. INTRODUCTION

Scheduling is one of the most critical functions in any manufacturing organization due to limited resources, increased customer expectation and fierce competition both domestically and internationally. Meanwhile, cost reduction and profit maximization continue to be strong motivations for all manufacturing companies in the environment of globalization and internationalization. As tardiness relates to operational costs, manufacturing scheduling can affect the performance of a company and hence chance of its survivability.

This paper focuses on single machine scheduling problem with nonzero ready times, which can be characterized as $1|r_j|\Sigma T_i$ using the Graham and Lawler classification [1]. Furthermore, it is assumed that all jobs arrive at different times with their arrival times known in advance. Even though the single machine manufacturing systems are rare in practice, the results can be used for bottleneck machines in production lines as well as manufacturing cells such as rotary injection molding machine in a shoe manufacturing cell, casting machine in a jewelry manufacturing cell, packing machine in a finishing line, robot in a highly flexible manufacturing cell, and so on. The objective is to decide the job sequence in order to minimize the total tardiness (TT), which measures the summation of tardiness of all the jobs, i.e. $TT = \Sigma T_i$ where $T_i = max\{0, C_i - D_i\}$, C_i is the completion time, D_i is the due date, and T_i is the tardiness of job *i*. A summary of solution techniques is given in Figure 1 for single machine scheduling problem with various constraints. The mathematical model can be used to reach the optimal solution in the case of zero ready times. However, when it comes to nonzero ready times, two possibilities can be considered; (1) preemption allowed and (2) preemption not allowed.

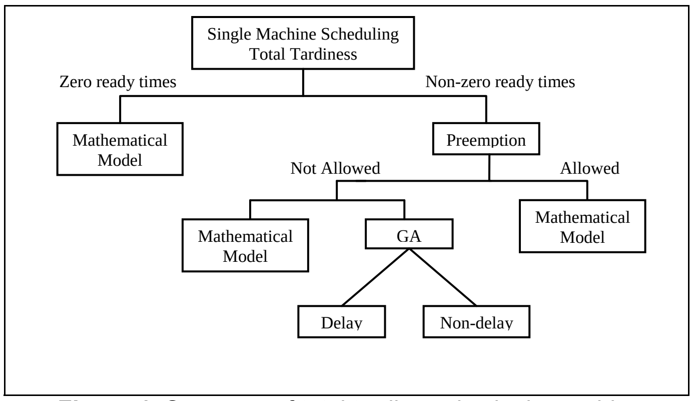

Figure 1. Summary of total tardiness in single machine scheduling

In the former case, a job's processing could be interrupted and another job can be assigned. In the latter case, once a job is assigned, it has to be processed to completion without any interruption. Here, two strategies have to be considered; 2a) assign one of the available jobs when the machine becomes available (Non-delay) or 2b) keep the machine idle until a particular job arrives (Delay).

This problem is known as NP-hard and some branch-and-bound procedures have been suggested for similar problems. This paper focuses on "preemption-not-allowed" case with the objective of minimizing the total tardiness by using genetic algorithm (GA) and mathematical model, while defining delay and non-delay strategies for each job independently makes this problem even more complicated.

Table 1. Dataset for example problem 1

Job	1	2	3	4	5
Processing time	2	2	9	4	6
Ready time	10	11	26	12	18
Due date	12	13	35	24	24
Strategy	Delay	Non-delay	Delay	Delay	Non-delay

A job sequence of 1-2-5-4-3 along with delay and non-delay assignment strategies for each job independently is shown in Figure 2 with a simple example problem given in Table 1.

Job	Idle Time (IT)	1	2	4	5	IT	3
C_i	10	12	14	18	24	26	35
T_i			1				

Figure 2. Gantt chart for example problem 1

Steps for generating the sequence are listed in Table 2, which gives the total tardiness to be 1 for this particular example.

Table 2. Steps for generating sequence for example problem 1

Step	Next job	Strategy	Job Available?	Decision	Start Time/ Completion Time
1	1	Delay	no, r_1=10	wait until r_1, then assign	12/10
2	2	Non-delay	yes, r_2=11	assign	14/12
3	5	Non-delay	no, r_5=18	consider next job	-/-
4	4	Non-delay	yes, r_4=12	assign	18/14
5	5	Non-delay	Yes, r_5=18	assign	24/18
6	3	Delay	yes, r_3=26	Wait until r_3, then assign	35/26

The remainder of this paper is organized as follows: section 2 presents the literature review of various solution techniques in single machine scheduling. In section 3, different genetic algorithm strategies are proposed. Furthermore, a MIP mathematical model, which provides optimal solutions, is described in details in section 4. In section 5, data analysis as well as computational results is presented. Finally, a brief summary is concluded in section 6.

2. LITERATURE REVIEW

Scheduling a set of jobs which are to be processed on a single machine to minimize total tardiness is known as the single machine total tardiness problem (SMTTP). Due to the complexity of SMTTP, minimizing total tardiness has been proved to be NP-hard by Du and Leung [2] with a given set of independent jobs on one single machine, meaning that it is impossible to find an optimal solution without using the enumerative algorithm. Moreover, computational time increases exponentially as the problem size grows. There has been various research works on single machine scheduling problem. Algorithms, including branch-and-bound approach, dynamic programming algorithm, and heuristic approaches have been applied to SMTTP.

Optimization approaches, such as branch-and-bound approach, dynamic programming algorithm, and mathematical model can guarantee the optimality. Schrage and Baker [3] developed a dynamic programming approach to the SMTTP. Hirakawa [4] proposed a quick branch-and-bound based optimal algorithm for SMTTP to minimize total tardiness, based on a branch-and-bound algorithm. Kondakci et al. [5] presented a new branch-and-bound algorithm for SMTTP. Furthermore, a mathematical modeling approach was developed by Panneerselvam [6] to minimize total tardiness and weighted total tardiness as well. Although mathematical model can provide optimal solutions, the number of constraints and variables become very large as the problem size grows. Therefore, Panneerselvam [7] argued that the mathematical model is limited to solve only small size problems due to the limitation of any operations research software.

In this case, heuristic approaches become a good option since they are designed with least numbers of steps to find near optimal solutions in reasonable time periods. Potts and Van Wassenhove [8] developed an algorithm, which decomposed the problem into subproblems so that they were sufficiently small in order to be solved by dynamic programming. Problems with up to 100 jobs were tested in this case. A net benefit of relocation heuristic (NBR) was developed by Holsenback and Russell [9], which avoided the enumeration of all possible sequences and could determine which job should come last to reduce the tardiness. Panwalker et al. [10] presented a P-S-K heuristic which was substantially better than others in respect of computational time. Alidee and Rosa [11] studied the group scheduling problem for minimizing the Total Tardiness on a single machine with a large number of jobs and machines and suggested various heuristic algorithms. Meanwhile, both weighted and unweighted tardiness were considered in their work by utilizing the Modified Due Date (MDD) algorithm proposed by [12]. For the same problem in a small scale, Gupta and Chantaravaraparan [13] proposed a mixed integer linear programming model and developed heuristics which are modifications of the NBR Heuristic

(1992) and the PSK Heuristic (1993). More recently, Baptiste [14] presented an algorithm for SMTTP with release dates and preemption jobs. Koulamas and Kyparisis [15] developed a polynomial time algorithm for SMTTP in the presence of deadlines. However, it was found that the algorithm could be extended only when deadlines and due dates were compatible and all job release times were equal. Later, the same algorithm was applied to single machine scheduling problems with setup times which were proportionate to the length of the already scheduled jobs, defined as past-sequence-dependent setup times (2008). Kanet and Li [16] generated a rule for Weighted Modified Due Date (WMDD) and compared it against other rules for weighted tardiness. Furthermore, Tian et al. [17] studied the single machine scheduling problem to minimize the total tardiness. Meanwhile, some optimality properties have been identified based on a polynomially solvable special case. As an extended research, an $O(n^2)$ time algorithm was proposed by Tian et al. [18] to minimize total tardiness of n equal-length preemptive jobs on a single machine.

To further improve the solution quality, meta-heuristics is introduced as an alternative approach compared to heuristics. Franca et al. [19] proposed a genetic algorithm (GA) as well as new meta-heuristic evolutionary algorithm, named memetic algorithm, for single machine scheduling problems with due dates and sequence dependent setup time. Later, memetic algorithm was further applied by Maheswaran et al. [20] to solve the single machine total weighted tardiness problems. Meta-heuristics developed by Feldmann and Biskup [21], including evolutionary strategies, simulated annealing, and threshold accepting, are efficient in obtaining near-optimal solutions by solving 140 benchmark problems with up to 1000 jobs. Moreover, an Ant Colony Optimization (ACO) was proposed by Cheng et al [22] for SMTTP. Vallada et al. [23] presented an evaluation of heuristics and meta-heuristics for the m-machine flowshop scheduling problem with the objective of minimizing total tardiness.

Even though single machine scheduling problem has received considerable amount of attention in the past, ready times are assumed to be zero in most of the problem definitions. In this paper, non-zero ready times is introduced to SMTTP. Furthermore, for studies that address total tardiness in the literature, they allow preemption or make very specific assumptions such as deadlines and due dates are compatible and all job release times are equal, etc. However, in this paper, all jobs Delay, all jobs Non-delay and Mixed strategies (some jobs delay and others non-delay) are considered in minimizing total tardiness and preemption is not allowed. This is believed to be a significant contribution to the literature, since there is no other literature addressing exactly the same problem to the best knowledge of authors. Dessouky and Deogun [24] presented a branch-and-bound procedure to minimize the average flow time when there are jobs with non-zero ready times. Later, Deogun [25] improved the previous approach by dividing the problem into subproblems and then applied branch-and-bound algorithm to each subproblem. Recently, Süer et al. [26] proposed an evolutionary programming (EP) to minimize the average flow time of a single machine scheduling problem in the presence of non-zero times and when preemption is not allowed. There are also some other researchers focusing on weighted completion time problem with nonzero ready times. In this paper, nonzero ready time as well as preemption not allowed situation are introduced, making the scheduling task more complicated.

Genetic algorithm (GA), first proposed by John Holland in the 1960s [27] and further developed by Goldberg [28], is a heuristic search algorithm that simulates the process of natural selection and evolution. In building genetic algorithm, five fundamental issues that affect the performance of GA must be addressed: chromosome representation, initialization of the population, selection strategy, genetic operators, and termination criterion. In the following subsections, those issues are introduced and described specifically for the proposed genetic algorithm.

3. GENETIC ALGORITHM SCHEME

3.1 Chromosome Representation

For any GA, the chromosome representation determines how the problem is structured in GA, as well as the genetic operators that can be used. Determining an appropriate representation of the variables is necessarily the first step in designing the GA. In this paper, a representation is developed in which each gene corresponds to the position of a job in the sequence. Specifically, with *N* genes in each chromosome, position *i* indicates the i^{th} job in the sequence as shown in Figure 3. Since delay and non-delay strategies are assigned to every job independently, each gene is represented by a pair of parameters *(X,Y)*, while *X* denotes the job being assigned and *Y* shows the scheduling strategy adapted (1 for delay, 2 for non-delay).

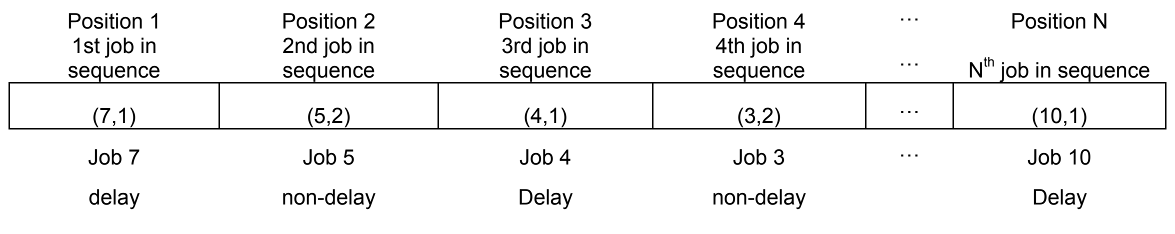

Figure 3. Gene representation for the scheduling problem

3.2 Fitness Function

In this paper, the objective is to minimize the total tardiness in the presence of non-zero ready times, which can be represented as the summation of tardiness values from all jobs as given in equation (1).

$$TT = \sum_{i=1}^{N} T_i \qquad (1)$$

3.3 Initialization of the Population

The initial population consists of *s* chromosomes. Since GA iteratively improves existing solutions, a completely random seeding of the initial population is employed in this paper. Therefore, for each chromosome, the probability that a job can be assigned to the 1st position is the same for all jobs. Once the 1st job in the sequence is determined and assigned, it is removed from the list and the remaining *(N-1)* jobs are to be assigned. The probability that any of the remaining jobs will be assigned to the 2nd position in the sequence is equal for all of the remaining jobs. As soon as the 2nd position is filled, remaining *(N-2)* jobs compete for the 3rd position and the assignment process continues until all jobs are assigned to a position. For scheduling strategy assignment, equal probability has been given to delay and non-delay strategies. In this case, every job will have a probability of 50% to be assigned either delay or non-delay strategy.

3.4 Reproduction

In this paper, a roulette wheel-based reproduction strategy is used. In this case, a fitness function-based reproduction probability is assigned to each chromosome to generate parents for mating. Since reproduction probability should increase as the total tardiness decreases, a simple transformation function is applied to give higher reproduction probability to those chromosomes with lower total tardiness. First, all total tardiness values are added as shown in equation (2). A constant “1” is added to every fitness function to avoid “division by zero”. Then, adjusted fitness values are computed for each chromosome by dividing the total fitness value by the corresponding fitness value (equation 3). Finally, reproduction probability is calculated for each chromosome based on the adjusted fitness value as shown in equation (4). Based on the reproduction probability, random numbers are generated to select mating parents.

$$TFF = \sum_{i=1}^{s} (FF_i + 1) \qquad (2)$$

$$AF_i = (TFF)/(FF_i + 1) \qquad (3)$$

$$p_i = AF_i / (\sum_{i=1}^{s} AF_i) \qquad (4)$$

Where, FF_i is total tardiness for chromosome *i; TFF* is sum of total tardiness values; AF_i is adjusted fitness value for chromosome *i; p_i* is reproduction probability for chromosome *i.*

3.5 Crossover Operator

For each pair, whether or not the crossover operator will be applied is tested first. If a randomly generated number is lower than the crossover probability, then crossover strategy is applied to the pair. The probability of crossover determines the rate at which the crossover is applied. A higher crossover probability can introduce new strings quickly into the population, while a lower one can cause stagnation in neighborhood search.

A single cut point crossover strategy from [29] is applied in the proposed GA. Assume that cut point has been randomly determined after the 3rd gene of the parents as shown in Figure 4. The last three genes are swapped thus generating two new offspring, while the first three genes are kept the same. However, it can be easily observed that both offspring are infeasible since job 6 is missing in the 1st offspring and job 2 is missing in the second one. Therefore, an offspring repair procedure is introduced to both offspring by replacing the repeated genes with the missing ones. Repeated genes are replaced by the missing ones in the order they appear in the other parent.

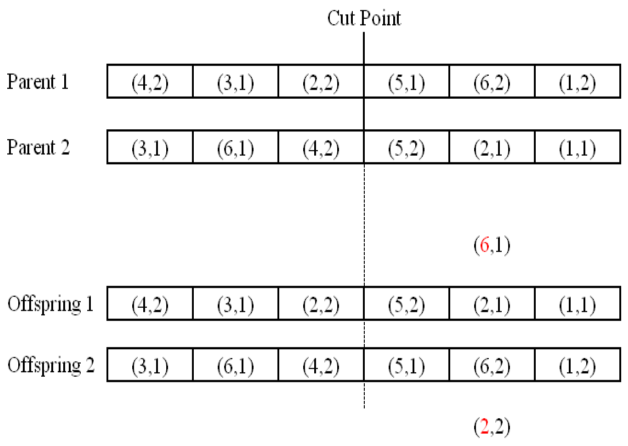

Figure 4. Single cut point crossover strategy

3.6 Mutation Operator

Mutation rate is the probability with which each gene in a new chromosome undergoes a random change after a selection process. A low mutation rate can help to prevent any gene from getting stuck to a single value, while a high mutation rate will result in random search eventually.

In the proposed GA, two kinds of mutation operators are introduced, which are job-based mutation and scheduling strategy-based mutation. Both mutation operators are applied to each gene independently. For job-based mutation, which was proposed by Gen and Chen [29], randomly selected job positions are swapped. As an example, mutation is applied to job 3 by swapping with job 5 randomly determined. For scheduling strategy-based mutation, current strategy of one job can be changed to the other one. Figure 5. shows that non-delay strategy of job 4 is changed to delay after mutation.

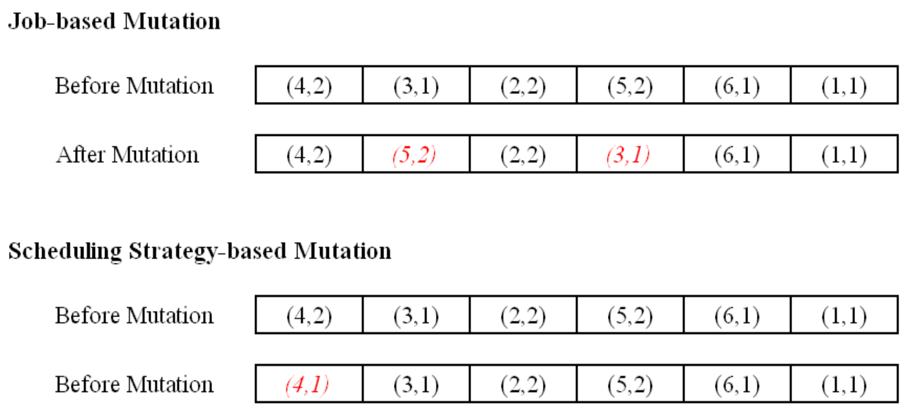

Figure 5. Job-based and scheduling strategy-based mutation

3.7 Selection Strategy

To choose chromosomes from current parents as well as offspring for the next generation, roulette wheel selection scheme from Bäck [30] is applied in this paper where good individuals have more chance to survive in the next generation. Meanwhile, roulette wheel selection can introduce certain randomness to the current population to increase the diversity. In roulette wheel selection, a probability of being selected is assigned to individuals, which is directly proportionate to goodness of their fitness function values. Then, randomly generated numbers determine chromosomes going to the next generation. In this case, chromosome replication in each generation is allowed.

3.8 Termination Criterion

The GA moves from one generation to another by selecting parents and producing offspring until a termination criterion is met. The most frequently used stopping criterion is a specified maximum number of generations. Moreover, since GA will force much of the entire population to converge to a single solution, some acceptable threshold can be established as the termination criterion, such as sum of the deviation among individuals and maximum number of generations without improvement. The experimentations in this paper employ a maximum number of generations as the stopping criterion.

4. MATHEMATICAL MODEL

A mathematical model is developed in this paper to find the sequence of N jobs on a single machine such that the total tardiness is minimized. Previously, [13] developed an integer programming for group scheduling problems to minimize total tardiness on a single machine for zero ready times. Kamat [31] proposed a math model to minimize the total tardiness in a multi-cell environment with zero ready times. However, to the best knowledge of authors, the mathematical model proposed in this paper with non-zero ready times restrictions has not been reported in the literature before. It is assumed that the processing times, ready times and due dates are known for all jobs. ILOG's Optimization Programming Language (OPL), which supports linear and quadratic objectives and constraints, as well as integer and real variables, is used to solve this mathematical model.

<u>*Objective Function:*</u>

Minimize $TT = \sum_{i=1}^{N} T_i$ (5)

<u>*Constraints:*</u>

$C_j = C_{j-1} + \sum_{i=1}^{N} P_i \times X_{ij}$ *j = 1, 2., N;* $C_0 = 0$

(6) $R_j \geq \sum_{i=1}^{N} RT_i \times X_{ij} - C_{j-1} - \sum_{k=1}^{j-1} R_k$ *j = 1, 2., N;* $R_0 = 0$

(7) $CT_j = C_j + \sum_{k=1}^{j} R_k$ *j = 1, 2., N*

(8) $T_j = CT_j - \sum_{i=1}^{N} D_i \times X_{ij}$ *j=1,2., N*

(9) $\sum_{i=1}^{N} X_{ij} = 1$ *j = 1, 2.,N*

(10) $\sum_{j=1}^{N} X_{ij} = 1$ *i=1, 2., N*

(11) $X_{ij} \in (0,1)$ integer; C_j, CT_j, T_j, R_j, $TT \geq 0$

Where,
X_{ij} =1, if job *i* is processed in sequence *j*; 0, otherwise
C_j is initial completion time of job in *jth* sequence (without considering ready times)
CT_j is final completion time of job in *jth* sequence (considering ready times)
T_j is tardiness value of job in *jth* sequence
R_j is waiting time of job in *jth* sequence
TT is total tardiness
D_i *is* due date of job *i*
P_i is processing time of job *i*
RT_i is ready time of job *i*
N is Number of jobs

The objective function given in equation (5) is to minimize the sum of tardiness values of all jobs. Equation (6) computes the initial completion time of job in jth position without considering the ready times, i.e., sums the initial completion time of the previous job and its own processing time. As the key constraint in this model, equation (7) presents the waiting time of each job. On one hand, if the ready time of the job to be processed is smaller than the initial completion time of the previous job plus waiting times of all previous jobs, there will be no extra waiting time for the job to be available. On the other hand, it will require additional waiting time until the job becomes available. Equation (8) gives the actual completion times of each job, while the tardiness of each job is described in equation (9). When the right hand side is positive, tardiness takes a positive value and when it is negative, tardiness becomes zero. Equation (10) guarantees that only one job is assigned to each position. Similarly, equation (11) promises that each job is assigned to only one position.

5. EXPERIMENTATION PERFORMED

In this section, the experimentation performed by using the mathematical model and the proposed genetic algorithm approach is described. All of the experiments were made using Dell PC with dual 2.40GHz CPU and 2.0 GB RAM.

5.1 Generating Datasets

For test purposes, five different problems are generated randomly, including 10-job, 20-job, 30-job, 50-job, and 100-job problem. The processing times, P_i follow uniform distribution *U*(1,10). The due dates, D_i have been generated by using D_i=RT_i +P_i*k where k is uniformly distributed *U*(1,4) and ready times, RT_i follow uniform distribution *U*(0, 40). Moreover, delay and non-delay scheduling strategies are randomly assigned to each job. In GA experimentations, ten runs are made for each case to keep the consistency.

5.2 Genetic Algorithm vs. Optimal Solution

The objective in this section is to test how good GA performs when compared to the optimal solutions. After various parameter combinations have been experimented for each problem independently, the results are shown in Tables 3 and 4.

Table 3. Results of GA and Mathematical Model

Problem	Genetic Algorithm					Math Model	Deviation
	Pop-size	No. Gen	Mut. rate	Cro. rate	Best	Optimal	
10-job	30	1000	0.01	1	22	22	0
20-job	80	5000	0.01	1	240	240	0
30-job	450	5000	0.001	0.8	798	798	0
50-job	400	5000	0.001	0.8	3262	3194	2.12
100-job	1200	10000	0.010	0.07	15493	14572	6.32

Table 4. GA and Math Model CPU Times

Problem	Math Model			Genetic Algorithm
	No. Variables	No. Constraints	CPU time	CPU time
10-job	142	61	2.8 sec.	3 sec.
20-job	482	121	7.1 sec.	63 sec.
30-job	1022	181	41.8 sec.	7 min, 16 sec.
50-job	2702	301	15 min, 25sec.	11 min.
100-job	10402	601	31 min, 46 sec.	22 min, 13 sec.

It can be easily seen that GA can achieve optimal solutions in small problems such as 10-job, 20-job, and 30-job problem. However, GA took longer CPU time to each the optimal solution compared to mathematical model, especially in 20-job and 30-job categories. In 50-job category, GA could not find the optimal solution but reached 2.12% away from it. Furthermore, in 100-job category, although mathematical model was unable to achieve optimality due to memory limitation of the computer, partial solution coming from mathematical model was still better than what GA could obtain. From another perspective, GA was significantly much faster than mathematical model for 50-job and 100-job problems. It is expected that as the problem size goes even higher, the number of variables and constraints of the mathematical model will increase dramatically, which prolongs the computational time. Hence, this limits the use of mathematical model to solve even larger problems.

Table 5. Comparison of different scheduling strategies

Problem	GA best solution (frequency)			Math Model
	Delay	Non-delay	Random	Optimal
10-job	22 (10)	22 (8)	22 (10)	22
20-job	241	241	240 (5)	240
30-job	800	799	798 (2)	798
50-job	3341	3262	3351	3194
100-job	16322	15493	15883	14572

Table 6. Results of Different Assignment Strategies

Run	10-job			20-job			30-job			50-job			100-job		
	Delay	Non-delay	Random	Delay	Non-delay	Random	Delay	Non-delay	Random	Delay	Non-delay	Random	Delay	Non-delay	Rando m
1	22	24	22	241	241	242	800	801	800	3381	3280	3391	15988	15389	15677
2	22	24	22	244	241	243	826	800	799	3394	3267	3401	15813	15444	15534
3	22	22	22	242	243	242	820	802	798	3401	3287	3379	16100	15459	15452
4	22	22	22	241	241	242	844	801	799	3366	3272	3399	16020	15438	15499
5	22	22	22	242	241	240	838	802	801	3341	3262	3402	15892	15334	15601
6	22	22	22	242	242	240	848	800	801	3371	3288	3388	15931	15411	15401
7	22	22	22	244	241	240	805	799	799	3352	3279	3375	16044	15326	15544
8	22	22	22	242	243	240	805	802	804	3345	3293	3369	16101	15206	15534
9	22	22	22	242	242	241	823	800	799	3399	3281	3361	15923	15399	15501
10	22	22	22	242	241	240	811	802	798	3402	3284	3351	15873	15276	15632
Best	22	22	22	241	241	240	800	799	798	3341	3262	3351	15813	15206	15401
Worst	22	24	22	244	243	243	848	802	804	3402	3293	3402	16101	15459	15677
Mean	22	22.4	22	242.2	241.6	241	822	800.9	799.8	3375.2	3279.3	3381.6	15968.5	15368.2	15537.5
STDEV	0	0.84	0	1.03	0.84	1.155	17.06	1.10	1.81	23.71	9.73	17.62	97.87	81.54	82.49
Freq	10	8	10	0	0	5	0	0	2	0	0	0	0	0	0

However, GA can still be used to find acceptable feasible solutions for those problems. Therefore, it can be concluded that GA performed well with respect to both small and large problems and it could find optimal or near optimal solutions in this experimentation.

5.3 Comparison of Different Scheduling Strategies

Since the proposed GA has proven to provide competitive results, the performance of three different assignment strategies, which are random, delay only and non-delay only, are compared in this section by using jobs generated in 6.1. Meanwhile, the assignment strategy of each job was forced to be kept the same during mutation when delay only and non-delay only strategies were applied. Results given in Tables 5 and 6 indicate that strategy selection was not significantly important in 10-job category as all three strategies were able to find the optimal solution.

On the other hand, random strategy was more effective than delay and non-delay in 20-job and 30-job categories based on the best as well as the average solutions. Furthermore, non-delay strategy showed to be more robust in that it could get results with lower standard deviation compared to delay and random strategies. In the mean time, it resulted in most favorable worst values almost in all problem types.

5.4 Parameter Analysis

In this section, various parameters, including population size, number of generations, and etc., are tested. 20-job problem is set as the benchmark problem in this case, and 10 GA runs are performed in each of the experimentation in the followings. Meanwhile, crossover rate is set as 1, and 0.01 is assigned as mutation probability.
With population size fixed at 80 and number of generations increasing from 1000 to 4000, the minimum total tardiness decreases regardless of scheduling strategies as shown in Figure 6. Meanwhile, one observation can be made is that the total tardiness value did not improve even if number of generations were increased to 5000 for all three strategies.

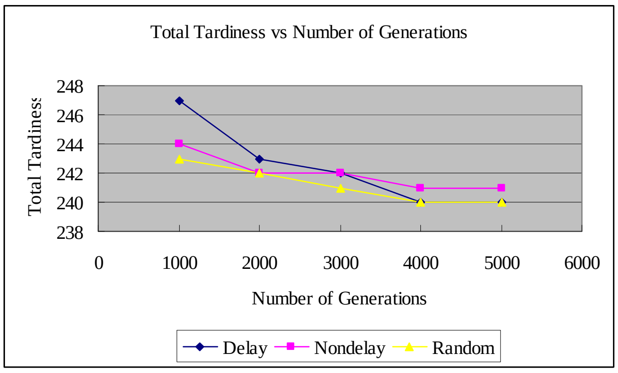

Figure 6. Number of Generations vs. Total Tardiness

From another perspective, while the number of generations is kept to be 5000, the lowest total tardiness values have been obtained when population size is 60 or higher. In this case, non-delay scheduling strategy was not affected by increased population size at all.

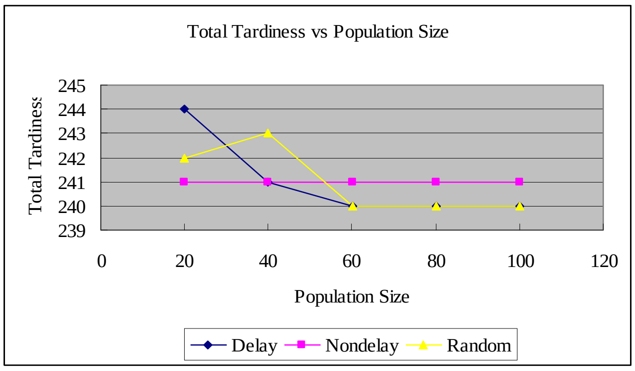

Figure 7. Population Size vs. Total Tardiness

Moreover, random strategy showed mixed results and delay strategy improved as the population size increased from 20-60. However, when population size increased beyond 60 in all three cases, no improvement was observed.

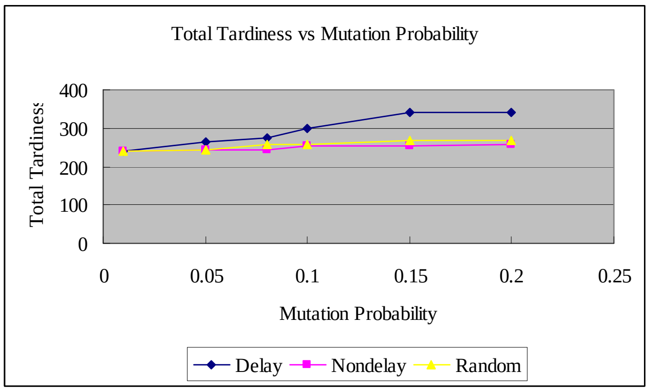

Figure 8. Total Tardiness vs. Mutation Probability

The final test is generated by changing the mutation probability, while number of generations remains to be 5000 as the previous case. Based on the best results from delay, non-delay and random strategies shown in Figure 8, the lowest total tardiness values have been obtained for mutation probability of 0.01 for all strategies.

6. CONCLUSIONS

In this paper, genetic algorithm as well as a mathematical model is proposed for a single machine scheduling problem with nonzero ready times to minimize the total tardiness. Although single machine scheduling problem has been widely studied in the literature, nonzero ready times have been rarely considered. Furthermore, with three different scheduling strategies integrated for each job independently, a new GA chromosome representation is created in this case, which further affects the genetic operators.

The results are encouraging in terms of solution quality as well as speed. GA could find near optimal or optimal solutions for the problems solved during experimentation. It can be concluded that GA can find optimal solutions for small problems. As to big problems, compared with mathematical model, GA may be preferable due to lower computational requirements even though it could not always reach the optimal solutions. On the other hand, the user can try all three scheduling strategies, which are Delay, Non-delay and Random, if there is enough time to solve the problem on hand when using GA. Otherwise, non-delay strategy may be a good option if the user has very limited time and cannot even make replications. However, it would be wise to try random strategy as well, if time permits, since it found the optimal solution more often than others. Finally, increasing number of generations mostly helped to reduce the total tardiness. However, population size beyond 60 did not affect solution quality in this experimentation at all. The computational time requirements grew almost linearly as the number of generations increased and finally lower mutation probability produced better results.

7. REFERENCES

[1] Graham, R.L., Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G. (1979). Optimization and approximating in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 4, 287-326

[2] Du, J., and Leung, J. (1990). Minimizing total tardiness on one machine is NP-hard, Mathematics of Operation Research, Vol 15, no 3.

[3] Schrage, L.E. and Baker, K.R. (1978). Dynamic programming solution of sequencing problems with precedence constraints. Operations Research, 26, 444-449

[4] Hirakawa, Y (1999). A quick optimal algorithm for sequencing on one machine to minimize total tardiness. Int J Prod Econ, 60-61, 20, 549-555

[5] Kondakei, et al. (1994). An efficient algorithm for the single machine tardiness problem. Int J Prod Econ, 36, 2, 213-219

[6] Panneerselvam, R. (1991). Modeling the single machine scheduling problem to minimize total tardiness and weighted total tardiness. Int J Manage Syst 7, 1, 37-48

[7] Panneerselvam, R. (2006). Simple heuristic to minimize total tardiness in a single machine scheduling problem. Int J Adv Manuf Technol, 30, 722-726

[8] Potts, C.N. and Van Wassenhove, L.N. (1982). A decomposition algorithm for the single machine total tardiness problem. Operations Research Letters, 1, 177-182

[9] Holsenback, J.E., and Russell, R.M. (1992). A heuristic algorithm for sequencing on one machine to minimize the total tardiness. Journal of Operations Research Society, vol. 43, 53- 62, 1992.

[10] Panwalkar, S.S., Smith, M.L., and Kolamas, C.P. (1993). A heuristic for the single machine tardiness problem. European Journal of Operations Research, 70, 304- 310.

[11] Alidaee, B., and Rosa, D. (1997). Scheduling parallel machines to minimize total weighted and unweighted tardiness. Computers & Operations Research, Vol 24, No.8, 775 – 788.

[12] Baker, K. R., and Bertrand J. W. (1982). A dynamic priority rule for scheduling against due-dates, Journal of Operational Management, 3, 37-42.

[13] Gupta, N. D., and Chantaravaraparan, S. (2007). Single machine group scheduling with family setups to minimize total tardiness. International Journal of Production Research, 1-16.

[14] Baptiste, P. (2000). Scheduling equal-length jobs on identical parallel machines, Discrete Appl. Math., 103, 1, 21-32

[15] Koulamas, C. and Kyparisis, G.J. (2001). Single machine scheduling with release times, deadlines, and tardiness objectives. European Journal of Operational Research, 447-453, 133.

[16] Kanet, J., and Li, X. (2004). A weighted modified due date rule for sequencing to minimize weighted tardiness. Journal of Scheduling, Vol 7, 261- 276.

[17] Tian, Z.J. et al. (2005). On the single machine total tardiness problem. European Journal of Operations Research, 165, 3, 843-846

[18] Tian. Z.J. et al. (2006). An algorithm for scheduling equal-length preemptive jobs on a single machine to minimize total tardiness. J Sched, 9, 343-346

[19] Franca, P.M., Mendes, A., and Moscato, P. (2001). A memetic algorithm for the total tardiness single machine scheduling problem. European Journal of Operations Research 132, 1, 224–242

[20] Maheswaran, R. et al. (2005). A meta-heuristic approach to single machine scheduling problems. The International Journal of Advanced Manufacturing Technology, 25, 772-776

[21] Feldmann, M. and Biskup, D. (2003). Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approach. Computers and Industrial Engineering archive 44, Issue 2 Special issue: Focused issue on applied meta-heuristics, 307-323

[22] Cheng, T.C.E., Lazarev, A., and Gafarov, E.R. (2009).A hybrid algorithm for the single-machine total tardiness problem. Computers and Operations Research, 36, 2, 308-315

[23] Vallada, E. et al. (2008). Minimizing total tardiness in the m-machine flowshop problem: A review and evaluation of heuristic and metaheuristics. Computers and Operations Research, 35, 4, 1350-1373

[24] Dessouky, M.I. and Deogun, J.S. (1981). Sequencing jobs with unequal ready times to minimize mean flow time. SIAM journal of Computing, 10, 192-202

[25] Deogun, J.S. (1983). On scheduling with ready times to minimize mean flow time. The Computer Journal, 26, 320-328

[26] Süer, G.A. et al. (2003). Evolutionary programming for minimizing the average flow time in the presence of non-zero ready times. Computers and Industrial Engineering, 45, 331-344

[27] Holland, J.H. (1975). Adaptations in natural and artificial systems. Ann Arbor: The University of Michigan Press.

[28] Goldberg, D.E. (1989). Genetic Algorithms in Search Optimization and Machine Learning. Addison Wesley.

[29] Gen, M. and Chen, R. (1997). Genetic Algorithm and Engineering Optimization. John Wiley & Sons, Inc. New York

[30] Bäck, T. (1996). Evolutionary Algorithm in Theory and Practice. Oxford University Press, 120

[31] Kamat, K.U. (2007). Minimizing total tardiness and crew size in labor intensive cells using mathematical models. Master's thesis, Ohio University

Genetski algoritam za minimalizaciju ukupne tromosti u rasporedu jedne mašine

Süer G. A., Yang X., Alhawari O. I., Santos J. and Vazquez R.

Primljen (9. jul 2012.); Recenziran (22. septembar 2012.); Prihvaćen (30. septembar 2012.)

Apstrakt

Minimalizacija ukupne tromosti u rasporedu jedne mašine je poznata kao NP-hard. U ovom radu, problem se proširuje da bi uključio ne-nulta vremena spremnosti, a apropriacija poslova nije dozvoljena. Prvo, razvijen je matematički model. Zbog računarskih kompleksnosti sa matematičkim modelom, takođe je predložen i pristup genetskog algoritma (GA) i njegove performanse su kasnije upoređene s optimalnim rešenjima. Rezultati pokazuju da GA može da pronađe optimalno rešenje za male probleme i približna optimalna rešenja za veće probleme. Rezultati takođe pokazuju da među strategijama Delay-only, Non-delay-only i Random, Non-delay strategija je proizvela mnogo rogobatnija rešenja dok je Random strategija pronašla optimalno rešenje u kategorijama manjih problema.

Ključne reči: *raspored za jednu mašinu, tromost, genetski algoritmi, matematičko modeliranje*

.