IJRSET Volume 2, Issue 5 Pages: 12-18

ISSN 2394-739X

International Journal for Research in Science Engineering and Technology

Minimizing Total Weighted Tardiness on Single Machine

[1] **Yasothei Suppiah,**
[1] Faculty of Engineering and Technology,
[1] Multimedia University,
[1] Melaka, Malaysia.

[2] **Kho Phin Shen,**
[2] Faculty of Engineering and Technology,
[2] Multimedia University,
[2] Melaka, Malaysia.

Abstract: -

This paper considers a single machine total weighted tardiness scheduling problem. Total weighted tardiness scheduling problem is proven to be NP-hard and cannot be solved in a reasonable time for large size problem. A metaheuristic which is genetic algorithm is developed to solve such scheduling problem in this paper. Besides that, dispatching heuristics are also developed which serves as an initial solution to genetic algorithm. The developed genetic algorithm has the capability to provide good results and good improvement compared to dispatching heuristics.

Keywords: - single machine, scheduling, tardiness, genetic algorithm, dispatching heuristic

1. INTRODUCTION

Single machine is defined as a machine that can only handle one job at a time without any interruption [1] and [2]. Single machine can be used to solve bottleneck problem in production lines such as automobile assembly line, packing machine in finish line etc [3]. By solving the bottleneck problem, the scheduling problem for the entire production line problems is solved. In real life practice, complex production systems are solved by decomposing the system into a series of single machine problem. This is why single machine problem are important and it have attracted a lot of researcher's interest and attention [4].

Scheduling plays an important role in most of the manufacturing industries. It is a decision making process that allocates limited resources to the processing tasks to meet certain requirement [5], [6] and [7]. A proper allocation of resource enables the company to optimize its objectives and achieve its goals. Since different industries have different objectives and there are a variety of algorithms, it is important to have an efficient scheduling algorithm to meet the objectives of every industry.

Tardiness is one of the commonly used performance criteria or objective in solving scheduling problem. It depicts the lateness of a job if it is completed after its due date and is zero when the job meets its due date or completed before its due date. As tardiness relates to operation cost [4], minimizing the tardiness becomes a strong motivation for the industry to minimize their cost. In this paper, a weight is added to each job to increase the complexity of tardiness problem. The weight is known as the priority level of job. Every job has different priority levels, thus it is a difficult task in deciding which job need to be schedule first as some jobs are more important than the other. Lawler [8] and Lenstra et al. [9] has been the pioneer in the literature to prove that the single machine problems with the total weighted tardiness criterion is NP-hard.

Exact methods such as branch and bound and dynamic programming are able to provide optimal solutions but at the expense of exponential growth of computational time.

Such algorithms have been used to tackle the single machine total weightd tardiness problem [10], [11] ,[12] and [13]. Therefore, many researchers develop heuristic algorithms instead which caters to large size scheduling problems. Dispatching heuristics have been a popular practice to many real-life industrial applications such as wafer fabrication plants, automatic guided vehicle systems, etc. It is one of the most common approaches in the industry as they can be easily understood and implemented besides providing reasonable solutions in a rather short time. Besides the dispatching heuristic, metaheuristics are gaining popularity in OR literature such as simulated annealing, genetic algorithm, ant colony optimization and tabu search. Genetic algorithm was developed by John Henry Holland [14] and [15]. The basic idea of genetic algorithms was inspired by the Darwin's theory of evolution, where the strong chromosomes tend to adapt and survive while the weak die. Many researchers have chosen genetic algorithm to tackle scheduling problem due to its flexibility [16]. Antonio Ferrolho and Manuel Crisostomo [17], Gursel A. Suer et. al [3] and Liu et. al [4] are some of the researchers who have applied genetic algorithm to solve scheduling of single machine with total weighted tardiness problem. In this paper, a genetic algorithm and three dispatching heuristics are developed. The developed dispatching heuristics are:

a) EDD (Earliest Due Date)
b) SPT (Shortest Processing Time)
c) LPT (Longest Processing Time)

This paper is organized as follows: The problem statement is provided in the next section, and then followed by the description of the heuristics' algorithms and procedure for testing and validation. Finally, results and discussions and conclusions are presented.

2. PROBLEM STATEMENT

This paper deals with single machine scheduling problem that minimizes the total weighted tardiness of jobs. The scheduling problem can be defined as follows:

There are N jobs waiting to be processed on a machine. The machine can handle only a single job at a time where no interruption is allowed during the process. Each job i has its own processing time (p_i), due date (d_i) and weight (w_i). The objective of this paper is to find a good sequence of jobs that minimizes the total weighted tardiness. The tardiness is defined as

$$T_i = \max(C_i - d_i, 0)$$

Where C_i is the completion time of job i.

These are the assumptions of single machine model for this paper:

- Machine can only process one job at a time.
- No setup time or settings are needed.
- All jobs are available for processing at time zero.
- Processing times, due dates and weights of jobs are known at the beginning of schedule (deterministic problem).
- Preemptions are not allowed, once processing of a job is started, it cannot be interrupted.
- The next scheduled job start immediately after the current job is completed.

3. DESCRIPTION OF HEURISTIC ALGORITHM

A Dispatching Heuristics

- EDD: Jobs are scheduled in sequence of increasing manner of due date
- SPT: Jobs are scheduled in sequence of increasing manner of processing time
- LPT: Jobs are scheduled in sequence of decreasing manner of processing time

B Genetic Algorithm

The genetic algorithm developed in this paper consists of these properties:

- Initial solution
- Selection
- Genetic operator
- Replacement
- Stopping criteria

Initial Solution

The dispatching heuristics which have been developed will be used as initial solution for the genetic algorithm.

Selection

In selection, a pair of chromosomes (parents) are select from the population to perform crossover and mutation to generate new chromosomes (children). In this research, a pair of parents is selected randomly in population to generate new chromosomes.

Genetic Operator

Using different types of operator will result in different final schedule. A good choice of genetic operator is important as it will affect the performance of genetic algorithm. Crossover and mutation operators are performed at every iteration. A position based crossover and order based mutation are used in this project due to its good performance and simplicity [18].

For the position based crossover, 0.5*N(N=number of job) number of job positions of a parent are randomly selected and the positions are fixed to perform crossover. The jobs in the fixed position of a parent will be kept unchanged in the offspring and the unallocated jobs in another parent will fill in the unfixed job positions in the child.

An example is provided to illustrate the crossover operators. Fig. 1 shows how the crossover operator is performed for a problem of 10 jobs. Parent 1 and parent 2 provide two different sequences of jobs. In the first diagram, a new chromosome or a child is generated by first duplicating the genes (jobs) in positions 3, 5, 6, 8 and 10 of parent 1. Then the remaining position in the child will be filled up by taking the gene from parent 2. In the second diagram, a child is generated by duplicating the genes (jobs) in positions 3, 5, 6, 8 and 10 of parent 2. Then the remaining position in the child will be filled up by taking the gene from parent 1. In this research, the jobs in the fixed position of parent 1 will be kept unchanged in the offspring and the unfixed job positions in the offspring will be taken from parent 2.

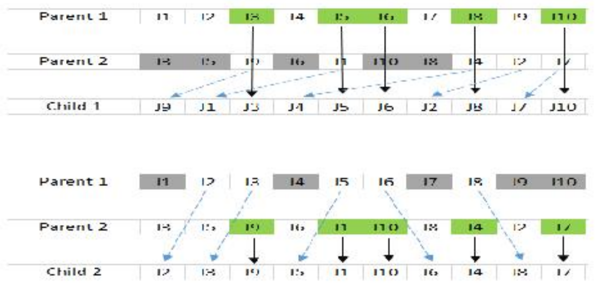

Figure 1: Illustration of position based crossover

On the other hand, for the order based mutation, two job positions are randomly selected from a parent. The jobs in the selected positions are swapped to form a new child. Fig. 2 provides an example of how the mutation works. A child is reproduced by exchanging the genes (jobs) on the selected positions. In this case, position 3 and 6 are selected and the gene are exchanged to create a new chromosome or child. The job positions are randomly selected at every iteration. For example in Fig. 2, job positions may be randomly selected at position 3 and position 6 at this iteration but the positions will be randomly selected again at next iteration.

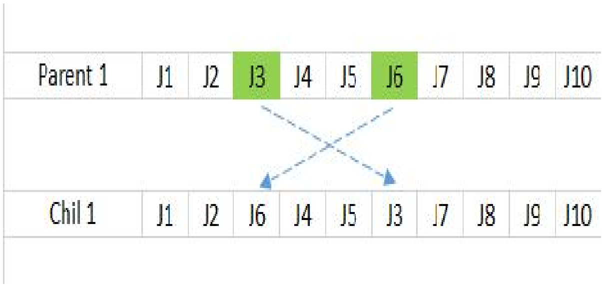

Figure 2: Illustration of order based mutation

Replacement

The two new born children created by crossover and mutation will be updated into the population pool. The fitness of these chromosomes will be compared to those chromosomes in the population. Chromosomes with better fitness will replace the less fit chromosomes. Chromosomes with better fitness will survive and reproduce and the less fit chromosomes will die. Fig. 3 illustrates how the replacement process is done.

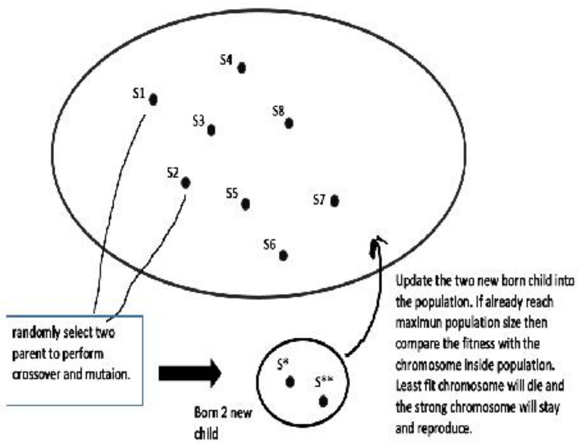

Figure 3: Replacement process

Stopping Criteria

The genetic algorithm in this research applies a fixed value of number of iterations as stopping criteria. The genetic algorithm stops after the maximum number of iteration is reached.

The flow chart of the developed genetic algorithm is shown in Fig. 4.

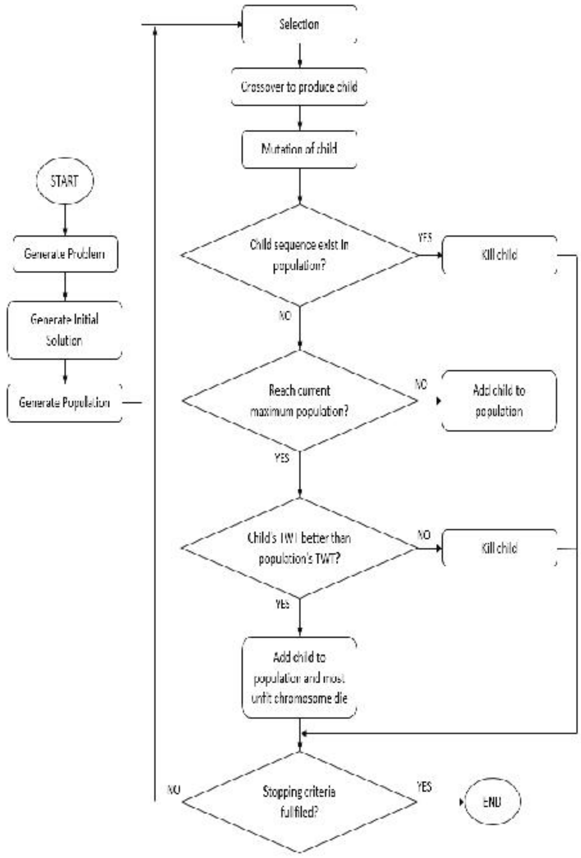

Figure 4: Flow chart of developed genetic algorithm

4. TESTING AND VALIDATION

Table I gives the details on how the simulation of data is performed to test the performance of the developed heuristics. Jobs are randomly generated by a set of job parameters which is extracted from the literature [2], [19] and [20].

Job Parameters	Value
Processing Time	Uniform between [1,100]
Weight	Uniform between [1,10]
Due Date	Uniform between [P*(1-TF-RDD/2), P*(1-TF+RDD/2)] If P*(1-TF-RDD/2)<=0 then [1, P*(1-TF+RDD/2)]

Table 1: Experiment Job Parameters

P is the total processing time of all jobs whereas TF and RDD is the tightness factor and the range of due date respectively. A high value of RDD shows that the range of due dates is very wide. A low value of RDD shows that the range of due dates is very narrow. On the other hand, a high value of TF shows that the due dates are tight and a low value of TF shows that the due dates are loose. In this paper, 25, 50 and 100 job size problems are conducted. Each job size has 25 combinations of RDD and TF values. 5 instances were generated for each combination of RDD and TF. Thus, there are total of 375 problem instances need to be generated and solved. Some preliminary experiments were conducted to ensure a suitable value for the genetic algorithms parameters such as the population size and iterations. Table II provides the genetic algorithm parameters used for different job sizes.

	Setting		
	RDD and TF value	Population size	Iterations
25 jobs	{0.2 , 0.4, 0.6, 0.8, 1.0}	10	2000
50 jobs	{0.2 , 0.4, 0.6, 0.8, 1.0}	10	10000
100 jobs	{0.2 , 0.4, 0.6, 0.8, 1.0}	10	10000

Table 2: Experiment settings

In this paper, Microsoft Excel spreadsheets VBA is used as a platform to develop both genetic algorithm and dispatching heuristic.The genetic algorithm was tested and validated by comparing its solution quality with respect to the dispatching heuristics.

5. RESULTS AND DISCUSSIONS

Experiments are carried out according to the data generation and parameters setting as discussed earlier. For every combination of RDD and TF, 5 problem instances were generated. All the dispatching heuristics and the genetic algorithm were tested on each of the 5 problem instances. An average value of the total weighted tardiness were calculated and recorded in Tables III, IV and V for all the developed methods. Columns 1 and 2 provide the combination of RDD and TF values. Columns 3-6 provide the average total weighted tardiness for EDD, SPT, LPT and genetic algorithm (GA) for 5 problem instances. The best value among the dispatching heuristics will be chosen as the initial solution for the genetic algorithm for every case. The last column provides the average time take by genetic algorithm to solve each of the 5 problem instances. There is no time reported in these tables for the dispatching heuristics since they solve all the problems in less than 2 seconds.

RDD	TF	Dispatching Rules			Genetic Algorithm	Time (s)
		EDD	SPT	LPT		
0.2	0.2	2347	2426	11141	485	23
	0.4	9792	8273	29544	3292	23
	0.6	25837	20361	46885	10454	22
	0.8	56451	36826	86870	25167	22
	1.0	59926	45841	103798	37451	22
0.4	0.2	282	3982	13296	213	23
	0.4	5167	12352	29656	2339	23
	0.6	30272	26841	56927	12312	22
	0.8	48601	34055	86658	23579	23
	1.0	51004	39363	88926	29654	23
0.6	0.2	-	-	-	-	-
	0.4	2648	10039	31852	909	23
	0.6	26054	22742	62593	9801	23
	0.8	36359	32637	78073	15871	23
	1.0	54180	44399	90930	28687	23
0.8	0.2	-	-	-	-	-
	0.4	1756	10302	29580	568	23
	0.6	21705	23973	52775	8251	23
	0.8	33755	29778	67272	13458	23
	1.0	49599	34481	77836	21790	23
1.0	0.2	-	-	-	-	-
	0.4	-	-	-	-	-
	0.6	10626	18440	49673	3283	23
	0.8	31039	28201	73115	13017	23
	1.0	41064	29672	74550	18423	23

Table 3: Average value of total weighted tardiness for 25 jobs

RDD	TF	Dispatching Rules			Genetic Algorithm	Time (s)
		EDD	SPT	LPT		
0.2	0.2	5520	8125	43798	1570	778
	0.4	51498	35709	135075	15018	771
	0.6	107015	72293	220525	35878	753
	0.8	229680	140344	348550	98769	777
	1.0	301790	221887	416620	172663	765
0.4	0.2	170	11643	51005	106	768
	0.4	25376	35819	121741	7620	765
	0.6	86884	75312	216338	29149	757
	0.8	237143	141721	352812	86420	767
	1.0	262151	170534	414069	130146	766
0.6	0.2	-	-	-	-	-
	0.4	12262	40575	130670	3467	756
	0.6	70016	81004	176006	19137	767
	0.8	132779	98183	290971	54251	768
	1.0	232320	170956	361416	121932	767
0.8	0.2	-	-	-	-	-
	0.4	7419	50897	112071	2426	776
	0.6	74113	88602	228807	23354	770
	0.8	113872	98523	252444	45430	769
	1.0	197444	140471	324008	89779	771
1.0	0.2	-	-	-	-	-
	0.4	-	-	-	-	-
	0.6	51655	93463	216597	15272	768
	0.8	104340	99925	239429	34195	761
	1.0	169922	123903	293587	66424	764

Table 4: Average value of total weighted tardiness for 50 jobs

RDD	TF	Dispatching Rules			Genetic Algorithm	Time (s)
		EDD	SPT	LPT		
0.2	0.2	18794	28109	178239	4881	6144
	0.4	162768	156363	467890	54041	6171
	0.6	443682	314751	868318	141276	6040
	0.8	882297	537135	1360216	374016	6065
	1.0	1237412	859252	1764080	690059	6125
0.4	0.2	261	44136	199530	89	6136
	0.4	104705	131286	468666	25125	6066
	0.6	390871	322381	853279	130173	6116
	0.8	756848	516165	1288516	315322	6027
	1.0	1098777	706538	1583378	525424	6045
0.6	0.2	-	-	-	-	-
	0.4	45834	166559	495079	11343	6086
	0.6	348195	367942	870655	113384	6100
	0.8	699992	511310	1218327	270245	6131
	1.0	943504	579414	1376541	407011	6115
0.8	0.2	-	-	-	-	-
	0.4	10954	199940	547947	4339	6145
	0.6	264808	378719	914872	78918	6018
	0.8	611113	477830	1153198	228601	6082
	1.0	827167	571106	1292603	347094	6123
1.0	0.2	-	-	-	-	-
	0.4	-	-	-	-	-
	0.6	83796	277270	843185	25381	6039
	0.8	365951	317697	930453	100770	5997
	1.0	718108	507739	1237291	266852	6084

Table 5: Average value of total weighted tardiness for 100 jobs

The boxes filled with '-'means that when EDD is applied to the problem, the total weighted tardiness of these problems are already zero. There will be no improvement after genetic algorithm is applied, as the value of the total weighted tardiness will still be zero. The reason is some combinations of RDD and TF produce easy problems with "loose" due dates. As there is no optimal solution available for these problems, the quality of the final solutions of the GA is then compared to other developed dispatching heuristics solutions. This is done by calculating the percentage relative improvement, PRI (%) where the formula is extracted from [21]. The performance of developed genetic algorithm can be evaluated by using the formula:

$$PRI(\%) = \frac{TWT_{DR} - TWT_{GA}}{TWT_{DR}} * 100$$

TWT_{DR} and TWT_{GA} are the average value of the total weighted tardiness of the dispatching heuristics and genetic algorithm respectively. Table VI provides all the values of the PRI (%) where the final solution of the genetic algorithm is compared to EDD and SPT rules. From tables III-V, it was shown that the best dispatching rules are always EDD and SPT. Total weighted tardiness of LPT are too big compared to the EDD and SPT. So the PRI

(%) calculated on Table VI only focuses on EDD and SPT since the LPT is not worthy to be compared to genetic algorithm solution quality. The first two columns in Table VI provide the combination of RDD and TF. The third and fourth columns provide the PRI(%) of genetic algorithm compared to EDD and SPT respectively for the case of 25 jobs. The fifth and sixth columns provide the PRI(%) of genetic algorithm compared to EDD and SPT respectively for the case of 50 jobs. The last two columns provide the PRI(%) of genetic algorithm compared to EDD and SPT respectively for the case of 100 jobs.

For example, the value of EDD-GA for the combination of RDD=0.2 and TF=0.2 which can be seen in column 3 is calculated as:

$$PRI(\%) = \frac{2347 - 485}{2347} * 100 \approx 79.33\%$$

		Percentage Relative Improvement(%)					
		25 jobs		50 jobs		100 jobs	
RDD	TF	EDD-GA	SPT-GA	EDD-GA	SPT-GA	EDD-GA	SPT-GA
0.2	0.2	79.33	80.00	71.55	80.67	74.02	82.63
	0.4	66.38	60.20	70.83	57.94	66.79	65.43
	0.6	59.53	48.65	66.47	50.37	62.25	46.79
	0.8	53.64	28.94	56.99	29.62	57.60	30.36
	1.0	37.50	18.30	42.78	22.18	44.23	19.69
0.4	0.2	24.46	94.65	37.64	99.08	65.90	99.79
	0.4	62.07	81.06	69.97	78.70	76.00	80.86
	0.6	59.32	54.12	64.45	61.29	66.69	56.62
	0.8	51.48	30.76	63.55	39.02	58.33	38.91
	1.0	41.85	24.66	50.35	23.68	52.18	25.63
0.6	0.2	-	-	-	-	-	-
	0.4	65.67	90.94	71.72	91.45	75.25	93.18
	0.6	62.38	56.90	72.66	76.37	67.43	69.18
	0.8	53.59	48.30	59.14	44.74	61.39	47.14
	1.0	47.05	35.38	47.51	28.67	56.86	29.75
0.8	0.2	-	-	-	-	-	-
	0.4	67.65	94.48	67.30	95.23	60.38	97.82
	0.6	61.98	65.58	68.48	73.64	70.19	79.16
	0.8	60.13	54.80	60.10	53.88	62.59	52.15
	1.0	56.06	36.80	54.52	36.08	58.03	39.22
1.0	0.2	-	-	-	-	-	-
	0.4	-	-	-	-	-	-
	0.6	69.10	82.19	70.43	83.65	69.71	90.84
	0.8	58.06	53.84	67.22	65.77	72.46	68.28
	1.0	55.13	37.91	60.90	46.39	62.83	47.44

Table 6: PRI(%) of average total weighted tardiness of GA

As can be observed in Table VI, for the cases of 25 jobs, the range of improvement of genetic algorithm compared to EDD and SPT falls in between 24.46%-79.33% and 18.30%-94.65% respectively. As for the cases of 50 jobs, the range of improvement of the genetic algorithm compared to EDD and SPT falls in between 37.64%-72.66% and 22.18%-99.08% respectively. For the cases of 100 jobs, the range of improvement of genetic algorithm compared to EDD and SPT falls in between 44.23%-76.00% and 19.69%-99.79% respectively. The genetic algorithm shows a wider range of improvement to SPT solution when the TF values are small. This also shows that the EDD provides better solution when the TF values are small for a fixed value of RDD. On the other hand, when TF value increases for a fixed value of RDD, the SPT tends to provide a better solution. Based on the overall results, the developed genetic algorithm provides good solutions to this scheduling problems. Although the EDD and SPT provide quick solutions to the scheduling problem, the genetic algorithm provides huge improvement to total weighted tardiness value which shows good solution quality. Having said that, the EDD and SPT are also good dispatching heuristics where reasonable solutions are provided in different cases or complexities. EDD always give better results of total weighted tardiness when the TF values are low, while SPT always give better results of total weighted tardiness when TF values are high. Despite the fact that the developed genetic algorithm gives good results of total weighted tardiness but it takes a longer computational time which is as a trade-off to its solution quality. All experiments are run on a PC Xeon E3 CPU with 3.4GHz and 8GB RAM in window 7 operating system.

CONCLUSION

In this paper, a single machine scheduling problem has been addressed. Single machine problems have been widely studied in operation research field due to its practical application of solving bottleneck problems in a more complex setting of machines in the industry. By developing genetic algorithm for this problem, the authors aim to find a good schedule that minimizes the total weighted tardiness of jobs for a single machine problem. From the extensive computational experiments, it was shown that the genetic algorithm has outperformed the performance of EDD, SPT and the LPT in a very reasonable time. This shows that the genetic algorithm is a good metaheuristic to solve scheduling problem for the single machine total weighted tardiness problem.

REFERENCES

[1] T.S. Abdul-Razaq, C.N. Potts and L.N.V. Wassenhove, A survey of algorithms for the single machine total weighted tardiness scheduling problem, Discrete Applied Mathematics, 26, pp 235-253, 1990.

[2] H.A. J. Crauwels, C.N. Potts, and L.N.V. Wassenhove, Local search heuristics for the single machine total weighted tardiness scheduling problem. Informs Journal on Computing, 10(3), 341-350, 1998.

[3] G. A. Suer, X. Yang, O.I. Alhawari, J. Santos and R. Vazquez, “A genetic algorithm approach for minimizing total tardiness in single machine scheduling,” International Journal of Industrial Engineering and Management, 3, pp. 163-171, 2012.

[4] N.Liu, M. Abdurrahman and S.Ramaswamy, A genetic algorithm for single machine total weighted tardiness scheduling problem, International Journal of Intelligent Control and Systems, 10, pp. 218-225, 2005.

[5] M. Pinedo and C. Xiuli, Operation Scheduling with Applications in Manufacturing and Services, McGraw-Hill Companies, 1999.

[6] M. L. Pinedo, Planning and Scheduling in Manufacturing and Services, New York: Springer, 2005.

[7] D. R. Sule, Industrial Scheduling, Boston: PWS Publishing Company, 1997.

[8] E. L. Lawler, A pseudo polynomial algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics, 1, pp. 331-342, 1977.

[9] J. K. Lenstra , A. H. G. Rinnooy Kan and P. Brucker , Complexity of Machine Scheduling Problems, Annals of Discrete Mathematics, 1, pp. 342-362, 1977.

[10] A. Jouglet, P. Baptiste and J. Carlier, Exact Procedure for Single Machine Total Cost Scheduling, IEEE International Conference on Systems, Man and Cybernetics, 6-9 October 2002

[11] J. C. Picard and M. Queyranne, The Time-Dependent Traveling Salesman Problem and its Application to the Tardiness Problem in One-Machine Scheduling, Operation Research, 26(1), pp. 86-110, 1978..

[12] C. N. Potts and L. N. V. Wassenhove, A Branch and Bound Algorithm for the Total Weighted Tardiness Problems, Operation Research, 33(2), pp. 363-377, 1985.

[13] J. Shwimer, On the n-job, One-machine, Sequence-independent Scheduling Problem with Tardiness Penalties: A Branch-Bound Solution, Management Science, 18(6), pp. B-301-B-313, 1972.

[14] J. H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, 1975.

[15] J. H. Holland, Adaption in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence, The MIT Press Cambridge, 1992.

[16] I.M. Alharkan, Algorithms for Sequencing and Scheduling,1st. ed., Riyadih: King Saud University, p.276, 2005.

[17] A. Ferrolho and M. Crisostomo, Single Machine Total Weighted Tardiness Problem with Genetic Algorithms, Computer Systems and Application, pp. 1-8, 2007.

[18] G. Syswerda, "Schedule Optimization Using Genetic Algorithms, Handbook of Genetic Algorithm, Van Nostrand Reinhold, New York,chapter 21,pp. 332-349, 1991.

[19] M. J. Geiger, On heuristic search for the single machien total weighted tardiness problem - some theoretical insights and their empirical verification, Europen Journal of Operation Research 207, pp. 1235-1243, 2010.

[20] P. A. Huegler and F. J. Vasko, A Performance Comparison of Heuristics for the Total Weighted Tardiness Problem, Computers and Industrial Engineering, vol. 4, no. 32, pp. 753-767, 1997.

[21] F. Jin, S. Song and C. Wu, A simulated annealing algorithm for single machine scheduling problems with family setups, Computers and Operations Research, vol. 36, no. 7, pp. 2133-2138, 2009.