International Journal of Engineering & Technology, 7 (2.29) (2018) 91-96

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Solving parallel machine scheduling problem with release dates using genetic algorithm

Yasothei Suppiah[1]*, Ajitha Angusamy[2], Goh Wei Wei[3], Noradzilah bt Ismail[4]

[1,3,4]Faculty of Engineering & Technology, Multimedia University
[2]Faculty of Business, Multimedia University
**Corresponding author E-mail: yasothei.suppiah@mmu.edu.my*

Abstract

This research deals with a scheduling problem for parallel machines environment to minimize total weighted tardiness with the consideration of sequence dependent setup times and release dates. There are two research questions that need to be addressed: 1) How to allocate jobs on machines ? 2) How to sequence jobs on each machine? Therefore, this research aims to find an efficient solution method that answers the research questions with the goal of minimizing the total weighted tardiness with the presence of sequence dependent setup times. Due to the complexity of the problem at hand, the authors have developed genetic algorithm to find a solution to this problem. Furthermore, various dispatching rules were used to enhance the performance of the genetic algorithm in terms of the total weighted tardiness value.

***Keywords*:** *Scheduling; Parallel machine; Genetic algorithm; Dispatching rule*

1 Introduction

In industrial manufacturing environment, a typical set of decision levels consists of facilities design, assignment of products to manufactuing plants where production scheduling and operations scheduling plays an important role.Thus, operations scheduling is at the bottom of the hierarchical planning system. Its purpose is to make the most detailed scheduling decisions, involving the assignment of the operations to specific machines and operators during a given time interval. The parallel machines shop can be regarded as an extension of the single machines, in the sense that the jobs still consist of only one operation (Graves, 1981). There are *m* machines that work in parallel. In general, a given job can be processed on several of the *m*-machines, possibly on any of them; however, different processing times may be required if the machines are not identical. While the single machine is more of theoretical interest, the parallel machines scheduling problem often arises in many real-life situations such as scheduling of a multiple-processors computer system on a set of jobs, and scheduling of jobs in a group technology cell with identical machines. Some other applications can be seen in the hospital scheduling of patients on identical test facilities and in the scheduling of deliveries on trucks. Futhermore, techniques studied in parallel machines problems are used in decomposition procedures in multistage system.

In scheduling, the decision making process of allocating limited resources over time in done with the purpose of optimizing certain objective functions or goals set by the industry (Baker, 1974). It is a regular practice in the shop floor of the industry, as a planning system assigns a completion date to every job and that the scheduler strives very hard to schedule the jobs so that the due dates could be met. Therefore, the complexities of this task have motivated the increase in due date scheduling in which it attracts both the researchers and industrial practitioners. One of the sought-after performance measures in today's world is the total weighted tardiness (TWT). Tardiness is a measure that depicts the lateness of an activity if it is completed after its due dates and takes the value of zero if it meets the due date or completed earlier than its due date. The weight component which is added to the tardiness shows the importance of an activity and the heavy penalty or costs that incurs if the due date is not met. The weight can be viewed as an economic interpretation such as dollar penalties per day of flow for the activity or it can be related to the importance of the customer. The costs or penalty include contract penalties, loss of sale, customer dissatisfaction and potential loss of reputation. Although the due date performance measures such as the TWT poses a theoretical challenge to researches, its significance in the industrial application is indispensable. Zhu and Wilhelm (2006) concluded in their review paper that due date related objectives is a fertile opportunities for future research. Allahverdi et al. (2008), a very prominent reviewer in the area of scheduling, provided an extensive review of the non-batch and batch scheduling problems. From their investigation, they revealed that most of the research papers made an assumption that the setup time is either neglected or considered as a portion of the job processing time. However, they concluded that since scheduling with setup times and cost is gaining popularity among schedulers in the industry; nevertheless this field of research has great potential for future research. In a recent analysis by Conner (2009), half of the 250 industrial projects consist of sequence-dependent setup times. In situations where these setups are applied well, 92% of the customers due dates could be met.

The problem statement of this research can be defined as such: There are a group of machines in parallel and each of the machines can process only one job at a time. We shall denote the number of jobs by n, their processing time by p_j. Each of job i, has a due date d_i, a processing time p_i, a weight w_i and job

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

ready time r_j (or may also be referred to as release date). When a job *k* is processed after job j then a setup time s_{jk} is incurred. The setup time is solely dependent on the job *j* and *k* and is independent of the machine on which the jobs are being processed. The completion time of job *j* is denoted by C_j, and the tardiness of job *j* is defined by $T_{ij} = \max(C_{ij} - d_{ij}, 0).$ By adopting the notations of Lawler et al. (1982), the scheduling problem is being research as $P_m / s_{jk}, r_j / \sum w_j T_j$.

2 Literature Review

The study of parallel machine scheduling problem is essential due to its widespread application in the real world industry (Lin and Lin (2013)). Despite the wide application of the parallel machine scheduling in practice, most of the study of these scheduling problems ignores problems with sequence dependent setup times or release dates (Shih Wei Lin et al., 2011).As the parallel machine problem TWT problem with release dates is an NP-hard scheduling problem (Lenstra et al., 1977, Pfund et al., 2008), the solution methods for industrial size problem focuses on heuristics methods in the literature. The common approach in industry is to use dispatching rules as it is the easiest way to address the parallel machine TWT problem and have been described by Pinedo (2002). With the advancement of computing systems in recent years, dispatching rules continue to be one of the most promising technologies for practical applications (Chen, et al., 2013). The static dispatching rule which requires at most *O*(*n log n*) computational time, for example, the earliest due date (EDD) rule, the shortest processing time (SPT) and the weighted shortest processing time (WSPT) are the simplest and most widely used rule. Very often the comparisons of dispatching rules are between the WSPT and EDD for the TWT problem. These can be seen in the work of Vepsalainen and Morton (1987), Morton and Pentico (1993), Huegler and Vasko (1997) and Volgenant and Teerhuis (1999) and many others. Regardless of producing fast solutions to the scheduling problems, the dispatching rules are known to be myopic and the solution quality is naturally much inferior compared to the optimal solutions (Pfund et al., 2008).

Vepsalainen and Morton (1987) applied the principles of the COVERT and the MOD rule to develop the ATC, a dynamic and consists of composite rule, for the parallel machine TWT problem. Pinedo (1995) stated that the EDD and the ATC rule have been popular while providing reasonably good results for problems related to tardiness. Lee et al. (1997) built upon the ATC approach and developed a 3-phase approach which consists of ATC with setups (ATCS) and followed by simulated annealing for improvement of the solution. Extension of the ATCS method was developed by Pfund et al. (2008) by using a grid approach to determine the scaling parameters for the ATCSR. The grid and regression versions of ATCSR gives better performance than grid and empirically based formula versions of ATCS, BATCS, and X-RM which are prominent algorithms in the literature. The alternative method to enhance the performance of the ATCS is to combine it with other metaheuristic which tends to be a more popular approach in the literature. Eom et al (2002) proposed a three-phase heuristic for the batch sequence dependent setup times in a parallel machine TWT problem. Tabu search was used in the final phase of the algorithm. Chang and Hyun (2003) modified the ATCS rule of Lee et al. (1997) and called is the MATCS rule where they employed a restricted tabu search algorithm with the elimination of non-effective job moves, for finding the best neighbourhood schedule. This method performed better than the basic tabu search and simulated annealing.

Genetic algorithm (GA) is another prominent metaheuristic in the scheduling literature which have been developed by Holland (1975). It is a search process simulating the natural evolutionary process. Starting with a current population of possible solutions to the scheduling problem, the best solutions are allowed to produce new children by the process of mutation and crossover in the aim of providing better generations that meet the goal or the objective of the scheduling. This approach has been found to quickly generate good solutions for a wide variety of scheduling problems (Schaller, 2014). Some successful applications of GA can be found in Malve & Uzsoy (2007), Zhou et al. (2009), Behnamian et al. (2009) ,Demirel (2011), Lin, Pfund and Fowler (2011) and Schaller (2014). In a very recent article, Joo & Kim (2015) developed a hybrid genetic algorithm with the combination of dispatching rule for the unrelated parallel machine and production availability. The objective of this problem is to determine the allocation policy of jobs and the scheduling policy of machines to minimize the total completion time. To solve the problem, a mathematical model for the optimal solution is derived, and hybrid genetic algorithms with three dispatching rules are proposed for large-sized problems.

This research addresses the gap in the literature review by addressing the sequence dependent setup times together with the TWT objective criteria and release dates that is often ignored by researchers in the scheduling environment. Furthermore, the direction of future research moves towards devising intrinsic design of heuristic algorithm that is able to provide efficient solution for the scheduling problem.

3 Methodology

The research methodology is as follows:

1) Allocations of jobs to machines by using dispatching heuristics.
2) Sequence the jobs in machines by dispatch heuristics and genetic algorithm.

3.1 Dispatch Heuristics

Six different dispatching heuristics algorithms are developed to perform the task of allocation of jobs to machines. These heuristics are EDD, WEDD, SPT, ATCSR, BATCS and BATCSmod.

a) EDD (Earliest due date)

Whenever a machine becomes free, unassigned jobs which are sorted in ascending order of the due date are assigned to the machine starting from job with the smallest due date.

b) WEDD (Weighted earliest due date)

Whenever a machine becomes free, unassigned jobs which are sorted in ascending order of d_j / w_j are assigned to the machine starting from job with the smallest value of d_j / w_j.

c) SPT (Shortest processing time)

Whenever a machine becomes free, unassigned jobs which are sorted in ascending order of the process time are assigned to the machine starting from job with the smallest processing time.

d) ATCSR (Apparent tardiness cost with setups and ready times index)

The ATCSR index for job *j* at current time *t* is calculated by:

$$I_{ATCSR}(t,l) = \frac{w_j}{p_j} \exp\left(\frac{-\max(d_j - p_j - \max(r_j t), 0}{k_1 \bar{p}}\right) \exp\left(\frac{-s_{lj}}{k_2 \bar{s}}\right) \exp\left(-\frac{\max(r_j - t, 0)}{k_2 \bar{p}}\right)$$

The urgency of scheduling a job is calculated by using the ATSR index. In this index, k_1, k_2 and k_3 are look-ahead parameters, s_{lj} is the setup time from job l to job j, $\bar{s}$ is the average setup time and $\bar{p}$ is the average processing time of the remaining unscheduled jobs. Whenever a machine becomes free, unassigned jobs which are sorted in descending order of $I_{ATCSR}(t,l)$ are

assigned to the machine starting from job with the largest value of $I_{ATCSR}(t,l)$.

e) BATCS (Batch apparent tardiness cost with setups)
The BATCS index for job *j* at current time *t* is calculated by:

$$I_{BATCS\,\mathrm{mod}}(t,l)=\frac{w_j}{p_j}\exp\left(\frac{-\max(d_j-p_j+\max(r_j-t,0),0}{k_1\bar{p}}\right)\exp\left(\frac{-s_{lj}}{k_2\bar{s}}\right)$$

Here, the index is calculated taking into consideration batch size of 1. In this index, k_1 and k_2 are look-ahead parameters, s_{lj} is the setup time from job *l* to job *j*, $\bar{s}$ is the average setup time and $\bar{p}$ is the average processing time of the remaining unscheduled jobs. Whenever a machine becomes free, unassigned jobs which are sorted in descending order of $I_{BATCS}(t,l)$ are assigned to the machine starting from job with the largest value of $I_{BATCS}(t,l)$.

f) BATCSmod (Batch apparent tardiness cost with setups and modified)
The BATCSmod index for job *j* at current time *t* is calculated by:

$$I_{BATCS\,\mathrm{mod}}(t,l)=\frac{w_j}{p_j}\exp\left(\frac{-\max(d_j-p_j+\max(r_j-t,0),0}{k_1\bar{p}}\right)\exp\left(\frac{-s_{lj}}{k_2\bar{s}}\right)$$

In this index, k_1 and k_2 are look-ahead parameters, s_{lj} is the setup time from job *l* to job *j*, $\bar{s}$ is the average setup time and $\bar{p}$ is the average processing time of the remaining unscheduled jobs. Whenever a machine becomes free, unassigned jobs which are sorted in descending order of $I_{BATCS\,\mathrm{mod}}(t,l)$ are assigned to the machine starting from job with the largest value of $I_{BATCS\,\mathrm{mod}}(t,l)$.

Once the dispatching heuristics assign jobs to each machine, the total weighted tardiness $\sum w_j T_j$ is evaluated by aggregating the total weighted tardiness values of the single machine sequences (obtained on each machine by the dispatching heuristics).

3.2 Genetic Algorithm

Genetic algorithm conducts robust search that was developed based on concepts and techniques from genetic and evolutionary theories. The genetic algorithm contains a current population of possible solutions to the scheduling problem at hand. At each generation, the intrinsic design of the genetic algorithm allows the improvement of the solution pool of the population similar to evolutionary theories whereby genes are refined to be more adaptive and stronger to withstand changes at every generation. In this scheduling problem, the individual fitness is evaluated by its value of total weighted tardiness which is called the fitness function. Therefore, in every generation, the fittest individuals which are the best solution are chosen to produce new solutions (children) by mixing the features of the parent (crossover) or by changing some elements of the selected parent (mutation). The worst child will die off or becomes extinct to pave way for a stronger population to be established. This process of creation of new population which replaces the old ones at every generation is repeated until a specific termination criterion is satisfied.
The following steps describe how the genetic algorithm is applied in this research which consists of initialization, selection, crossover, mutation, evaluation and termination criterion. A chromosome is defined as sequence of jobs in machines which is composed of sub-chromosomes that represents sequence of jobs in each machine. Each job in a chromosome is referred to as a gene.
i)Initialization: Generate initial population by using the dispatch heuristics solutions.
ii)Selection: Two individuals (parents) with the best fitness function are selected. In other words, two chromosomes which provide the lowest total weighted tardiness value are selected from the pool of the population.
iii) Crossover: Once parents are selected, crossover operation is performed with a crossover probability of p_c to generate two new children solutions. A single point crossover method is used where a random number between 2 and *n*-1 is generated (*n* is the length of the chromosome) to determine the crossover point. The genes before the crossover point in the first chromosome represent the first part of the first child chromosome. The second part of the first child chromosome is generated by adding those genes that are not yet in the child chromosome in order of their appearance in the second chromosome. On the other hand, the genes before the crossover point in the second chromosome represent the first part of the second child chromosome. The second part of the second child chromosome is generated by adding those genes that are not yet in the child chromosome in order of their appearance in the first chromosome. Figure 1 below provides an example of how a crossover process is performed. The number of jobs in each machine for the first child is the same as parent 1. Similarly, the number of jobs in each machine for the second child is the same as the second parent.

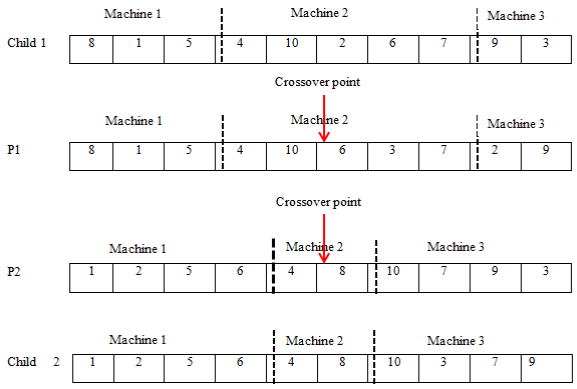


Fig 1:. Crossover operation

iv) Mutation: A mutation operation is performed with a mutation probability of p_m to generate a new child solution. Two genes (jobs) are randomly selected from the parent chromosome and are interchanged.
v) Evaluation: All the child chromosomes from the crossover and mutation processes are evaluated on their fitness function value (total weighted tardiness). These chromosomes will be evaluated together with other chromosomes in the solution pool. If the number of chromosomes does not exceed the size of the population, all the child chromosomes will be included as part of the population pool. Otherwise, the chromosomes that performs the worst will be excluded from being part of the solution pool. This method ensures that the population size is stable while ensuring a good solution pool is created at every generation.
vi) Termination criteria: Process i) - v) is repeated until a termination criterion (maximum number of generations) is met. The chromosome that best contributes to the fitness function value will be selected as the solution to the scheduling problem.
The pseudocode for the Genetic algorithm is presented in Figure 2.

Step 1: Set the input for crossover probability P_c, mutation probability P_m, population size S_{pop} and maximum generation $G_{\max}$.
Step 2: Generate an initial population by using the dispatch heuristics solutions.
Step 3: Perform selection operation by selecting two chromosomes from the population which provide the best and next to best values of fitness function.
Step 4: Perform crossover and mutation operations according to their probabilities.
Step 5: Evaluate the children chromosome on their fitness function.
Step 6: Include the children chromosomes into the population pool until the population size is $\leq S_{pop}$. Otherwise go to step 7.
Step 7: Choose the child chromosome which performs the best in terms of fitness function from Step 5. Compare the fitness function with the chromosomes in the population. The chromosomes that performs the worst will be excluded from the population.
Step 8: Repeat Step 2 – Step 7 until the maximum generation $G_{\max}$ is met.
Step 9: Select the best chromosome in the population pool in terms of its value of fitness function and the sequences of jobs in each machine.

Fig 2: Pseudocode for the genetic algorithm

4. Results and Findings

Once the solution methodology and the heuristics developments are completed there need a systematic testing to investigate the performance of the developed heuristics. In this research, all heuristics model was developed and tested by using Matlab software.

4.1 Data Generation

Random instances are generated to investigate the performance of the developed models. Here, the random instances is established by using similar designs which were considered earlier by Lee and Pinedo (1997) and Pfund et al. (2008).

1. Number of machines is 3, 4 and 5.
2. Number of jobs is 8 and 10.
3. Processing time p_j is uniformly distributed over the interval [50,150].
4. Weights w_j is uniformly distributed over the interval [1, 10].
5. The mean setup time is calculated by $\bar{s} = 2 * \bar{p}$. Setup time is uniformly distributed over the interval $[0, 2\bar{s}]$.
6. Makespan is estimated by $C_{\max} = (0.128003\,\bar{s} + \bar{p}) * 27$.
7. The mean due date is $\bar{d} = 0.1 * C_{\max}$.
8. The due dates are uniformly distributed over the interval $[0.75\bar{d}, \bar{d}]$ with probability 0.9 and uniformly distributed over the interval $[\bar{d}, \bar{d} + (C_{\max} - \bar{d}) * 0.25]$ with probability 0.1.
9. Ready times are assigned the value 0 if random number generated is less than 0.2. Otherwise, ready times are uniformly distributed over the interval $\lfloor d_j - p_j, d_j \rfloor$. If $d_j - p_j < 0$ then the ready times are distributed over the interval $\lfloor 0, d_j \rfloor$.
10. Look ahead parameters are $k_1 = 5, k_2 = 0.5$ and $k_3 = 0.005$.
11. Inputs for genetic algorithm: $P_c = 0.6$, $P_m = 0.2$; $S_{pop} = 50$, $G_{\max} = 300$.

4.2 Results

The solution quality of all the heuristics models are tested with respect to their value of the objective function which is the total weighted tardiness (TWT) value. Comparison among the heuristics will be conducted to investigate which heuristics shows the best performance overall. By taking the heuristics which is the winner of the competition among the developed heuristics, the author adopts the percentage relative improvement (PRI) proposed by Jin et.al. (2009), which is given by the following formula:

$$PRI(\%) = \frac{TWT_{HEU} - TWT_{heubest}}{TWT_{HEU}} * 100$$

where TWT_{HEU} is the TWT for a heuristic for the particular case, and $TWT_{heubest}$ is the TWT for the best heuristic found for the particular case. In this way, the authors investigate how much of improvement of the best heuristic solution from a particular heuristic solution found so far for a particular case. Table 1 shows an output of a problem instance of 8 jobs in 3 machines.

Table 1: Solution for a problem instance of 8 jobs in 3 machines

Heuristic	Mach 1	Mach 2	Mach 3	TWT	PRI%	CPU time (s)
EDD	3,5,2	1,7,4	8,6	3491	48.44	6.597
WEDD	3,7	8,5,4	6,1,2	4331	58.44	6.598
WSPT	1,6	3,7,2	5,8,4	7591	76.29	6.599
ATCSR	3,7	8,1	5,2,6,4	5770	68.8	6.600
BATCS	1,2	3,5,6	8,7,4	3543	49.2	6.601
BATCSmod	1,6	3,5,2	8,7,4	3843	53.16	6.602
GA	3,6,2	1,5,7	8,4	1800	0.00	54.997

Column 1 provides the specific heuristic, columns 2 - 4 provide the sequences of jobs in each machine, column 5 gives the total weighted tardiness value, followed by column 6 which provides the percentage of relative improvement of the best heuristic to the specific heuristic and the last column provides the computational time taken to provide solution for each of the heuristics. From table 1, GA provides the best TWT value, which is 1800 with a relative improvement in the range of 48.44%-76.29%. Among the dispatch heuristics, EDD performs the best whereas WSPT performs the worst. However, the dispatch heuristics consumes less time compared to GA in providing the results. This is because the GA contains a more complex steps and iterations in providing the final output. Table 2 shows an output of a problem instance of 8 jobs in 4 and 5 machines.

In Table 2, column 1 provides the specific heuristic. Columns 2-4 provide results for the case of 8 jobs 4 machines for all the heuristics. Column 2 shows the TWT values, column 3 shows the percentage of relative improvement of the best heuristic to the specific heuristic and column 4 provides the computational time taken to provide solution for each of the heuristics. Columns 5-7 provide results for the case of 8 jobs 5 machines for all the heuristics whereby the explanations are similar to columns 2-4.
In the case of 4 machines problem, both GA and WEDD provide the best answer and shows an improvement in the range of 70.81% - 83.30%. WSPT rule provide the worst results.
On the other hand, in the case of 5 machines problem, GA provided the best results with an improvement of 28%-76.27% compared to other heuristics. In terms of the computational time, the dispatch heuristics works much faster than GA in providing the solution to this scheduling problem.
In general, Tables 1 and 2 indicate that the TWT values decreases in most cases when there is an increase in the number of machines for the same number of jobs. The reason is as more machines are available, the earlier the jobs are processed which enables the tardiness value to decrease. However, in some cases, the combination of job sequences in each machine influences the deterioration of TWT value. Furthermore, in this study, it was found that the computational time to obtain final solution for each heuristic in-

creases when there is an increase in the number of machines for the same number of jobs. Table 3 shows an output of a problem instance of 10 jobs in 3 machines.

Table 2: Solution for a problem instance of 8 jobs in 4 and 5 machines

Heuristic	TWT 4 machines	PRI%	CPU time (s)	TWT 5 machines	PRI%	CPU time (s)
EDD	3320	70.81	14.408	2,515	64.21	21.202
WEDD	969	0.00	14.410	1,563	42.42	21.205
WSPT	5803	83.30	14.412	1,250	28	21.208
ATCSR	3679	73.66	14.414	3,793	76.27	21.211
BATCS	3320	70.81	14.416	1,250	28	21.214
BATCSmod	3320	70.81	14.418	1,250	28	21.217
GA	969	0.00	133.042	900	0.00	200.933

Table 3: Solution for a problem instance of 10 jobs in 3 machines

Heuristic	Mach 1	Mach 2	Mach 3	TWT	PRI%	CPU time (s)
EDD	J3, J8, J2,J4	J9, J10, J6	J1, J5, J7	14, 662	76.78	8.816
WEDD	J3,J6,J1,J2	J9,J7,J4	J10,J8,J5	6,931	50.89	8.817
WSPT	J1,J8,J4	J10,J5,J6	J3,J7,J9,J2	12,022	71.69	8.819
ATCSR	J10,J7,J9	J3,J5,J1,J4	J8,J2,J6	3404	0.00	8.820
BATCS	J10,J8,J6	J1,J7,J4	J3,J5,J9,J2,	5816	41.47	8.821
BATCSmod	J10,J8,J6	J1,J7,J2	J3,J5,J9,J4	5816	41.47	8.823
GA	J10,J7,J9	J3,J5,J1,J4	J8,J2,J6	3404	0.00	77.167

Table 4: Solution for a problem instance of 10 jobs in 4 and 5 machines

Heuristic	TWT 4 machines	PRI%	CPU time (s)	TWT 5 machines	PRI%	CPU time (s)
EDD	6839	72.67	16.663	2065	5.38	23.487
WEDD	6783	72.45	16.665	1954	0.00	23.490
WSPT	7291	74.37	16.667	2681	27.12	23.493
ATCSR	5016	62.74	16,669	4095	52.28	23.497
BATCS	4178	55.27	16.672	3602	45.75	23.500
BATCSmod	4178	55.27	16.675	3602	45.75	23.504
GA	1869	0.00	155.570	1954	0.00	223.763

Table 5: Solution for a problem instance of 50 jobs with machine 3, 4 and 5

Heuristic	TWT 3 machines	PRI%	TWT 4 machines	PRI%	TWT 5 machines	PRI%
EDD	250284.8	53.04	178,391	52.44	149,449	53.83
WEDD	183522	35.96	149,712	43.33	115,056	40.03
WSPT	191367	38.59	150,490	43.62	115,611	40.32
ATCSR	117522.6	0.00	84,842	0.00	68,997	0.00
BATCS	131817.1	10.84	94,769	10.48	75,186	8.23
BATCSmod	132664.5	11.41	98,261	13.66	77,564	11.05

Table 6: Solution for a problem instance of 100 jobs with machine 3, 4 and 5

Heuristic	TWT 3 machines	PRI%	TWT 4 machines	PRI%	TWT 5 machines	PRI%
EDD	1,091,107	55.60	794,457	54.38	645,363	54.69
WEDD	1,311,611	63.07	602,683	39.87	673,497	56.58
WSPT	690,081	29.80	544,522	33.45	458,939	36.29
ATCSR	484,410	0.00	362,405	0.00	292,407	0.00
BATCS	498,346	2.80	372,437	2.69	300,776	2.78
BATCSmod	507,510	4.55	381,168	4.92	302,132	3.22

Table 7: Solution for a problem instanceez of 200 jobs with machine 3, 4 and 5

Heuristic	TWT 3 machines	PRI%	TWT 4 machines	PRI%	TWT 5 machines	PRI%
EDD	4,197,132	56.28	3,065,844	55.04	2,419,332	54.41
WEDD	3,049,731	39.83	2,282,784	39.62	6,294,840	82.48
WSPT	5,885,477	68.82	6,590,456	79.08	1,870,720	41.05
ATCSR	1,835,004	0.00	1,378,420	0.00	1,102,860	0.00
BATCS	1,852,157	0.93	1,396,805	1.32	1,129,059	2.32
BATCSmod	1,862,392	1.47	1,393,506	1.08	1,107,295	0.40

Table 8: Solution for a problem instance of 250 jobs with machine 3, 4 and 5

Heuristic	TWT 3 machines	PRI%	TWT 4 machines	PRI%	TWT 5 machines	PRI%
EDD	6,366,133	57.35	4,754,897	56.83	3,711,955	56.34
WEDD	4,564,284	40.51	3,398,062	39.60	9,238,351	82.46
WSPT	4,141,504	34.44	3,069,509	33.13	9,837,884	83.53
ATCSR	2,715,271	0.00	2,052,485	0.00	1,620,539	0.00
BATCS	2,798,703	2.98	2,093,075	1.94	1,678,853	3.47
BATCSmod	2,781,734	2.39	2,098,458	2.19	1,671,425	3.04

Table 3, Column 1 provides the specific heuristic, columns 2 - 4 provides the sequences of jobs in each machine, column 5 gives the total weighted tardiness value, followed by percentage of relative improvement of the best heuristic to the specific heuristic and the last column provides the computational time taken to provide solution for each of the heuristics. From table 3, both ATCSR and GA provide the best TWT value, which is 3404. EDD performs the worst among all heuristics. Both BATCS and BATCSmod provide the same value of TWT. Similar to the observations in Tables 1 and 2, the dispatch heuristics in this case too consumes less time compared to GA in providing the results. Table 4 shows an output of a problem instance of 10 jobs in 4 and 5 machines.

In table 4, in the case of 4 machines problem, GA provides the best answer and shows an improvement in the range of 55.27% - 74.37%. As both BATCS and BATCSmod provide the best solution among the dispatch heuristics, WSPT rule provide the worst results.

On the other hand, in the case of 5 machines problem, both WEDD and GA provide the best results. These two heuristics provided an improvement of 5.38%-52.28% compared to other heuristics. Furthermore, similar observations was found in terms of improvement of total weighted tardiness value and increment of computational time when there is an increase in the number of machines for the same number of jobs.

The authors performed further testing to analyse the performance of the dispatch heuristics in large size jobs instances as there are inconsistencies in their performance towards the TWT values in Tables 1-4. Table 5 shows an output of a problem instance of 50 jobs with machine 3, 4 and 5

Tables 5-8 clearly indicate that ATCSR outperforms all other dispatch heuristics in terms of the total weighted tardiness value for a large size problem. The design of the algorithm of ATCSR encourages selection of non-ready jobs that contributes to the reduction of the total weighted tardiness value. In other words the algorithm ensures sequences of jobs with reduced setups or giving preference to higher priority jobs arriving in the near future. The experiments on the performances of the genetic algorithm for large job size problems are still in progress as this algorithm consumes extensive computational time and effort to produce solution.

5 Conclusion

This paper addresses the parallel machines scheduling problem with the consideration of release dates and sequence dependent setup times. The aim of this paper is to provide a good sequence of jobs in machines that contribute to the minimization of total weighted tardiness value. Due to the complexity of the problem, the authors have developed six different types of dispatch heuristics and a genetic algorithm. The developed genetic algorithm integrates the solutions of the dispatch heuristics as an initial solution and further thrives to provide a better solution quality. Based on the experiments conducted, the genetic algorithm provides a good solution quality to this scheduling problem. Furthermore, the ATCSR heuristics performs the best among the dispatch heuristics. The authors are further testing the developed algorithm on other combinations of jobs and machines. Future work can be in the direction of reducing the computational time of the genetic algorithm in providing the solution.

Acknowledgement

This research work is supported by the Project Mini Fund (MMUI/160008) supported by Multimedia University.

References

[1]. Allahverdi A., Ng C.T., Cheng T.C.E., Kovalyov Mikhail Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, 985-1032.

[2]. Baker K.R. (1974). *Introduction to sequencing and scheduling* . Wiley New York.

[3]. Behnamian J., Ghomi F., & Zandieh M. (2009). A multi-phase covering Pareto-optimal front method to multi-objective scheduling in a realistic hybrid flowshop using a hybrid metaheuristic. *International Journal of Production Research, 48*, 4949–4976.

[4]. Chang O.K & Hyun J.S. (2003). Scheduling jobs on parallel machines: a restricted tabu search approach. *International Journal of Advanced Manufacturing Technology 22*, 278-287.

[5]. Chen T., Rajendran C. and Wu C.W. (2013). Advanced dispatching rules for large-scale manufacturing. *International Journal of Advanced Manufacturing Technology*, 1-3.

[6]. Conner G. (2009). *10 questions.* Manufacturing Engineering Magazine.

[7]. Demiral T., Özkır V., Demirel N. C. and Taşdelen B. (2011). A Genetic Algorithm Approach for Minimizing Total Tardiness in Parallel Machine Scheduling Problems. *Proceedings of the World Congress on Engineering, Vol II WCE.* London.

[8]. Eom D.H., Shin H.J., Kwun I.H, Shim J.K & Kim S.S. (2002). Scheduling jobs on parallel machines with sequence-dependent family set-up times. *The International Journal of Advanced Manufacturing Technology 19*, 926-932.

[9]. Graves S.C. (1981). A review of Production Scheduling. *Operations Research*, 646-675.

[10]. Holland J.R. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press.

[11]. Huegler P.A. and Vasko F.J. (1997). A performance comparison of heuristics for the total weighted tardiness problem. *Computers and Industrial Engineering, 32(4)*, 753-767.

[12]. Jin F., Song S. and Wu C. (2009). A simulated annealing algorithm for single machine scheduling problems with family setups. *Computers and Operations Research,36* , 2133-2138.

[13]. Joo C.M. & Kim B.S. (2015). Hybrid genetic algorithms with dispatching rules for unrelated parallel machine scheduling with setup time and production availability. *Computers & Industrial Engineering, 85*, 102-109.

[14]. Lawler E.L., Lenstra J.K., Rinnooy Kan A.H.G. (1982). Recent developments in deterministic sequencing and scheduling: A survey. In *Deterministic and Stochastic Scheduling* (pp. 35-74).

[15]. Lee Y.H., Bhaskaran K. and Pinedo M. (1997). A heurisitic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions, 29(1)*, 45-52.

[16]. Lenstra J.K., Rinnooy Kan A.H.G. and Brucker P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 343-362.

[17]. Lin Y.K. and Lin C.W. (2013). Dispatching rules for unrelated parallel machine scheduling with release dates. *International Journal of Advanced Manufacturing Technology, 67*, 269-279.

[18]. Lin Y.K., Pfund M.E. and Fowler J.W. (2011). Heuristics for minimizing regular performance measures in unrelated parallel machine scheduling problem. *Computers & Operations Research, 38 (6)*, 901–916.

[19]. Malve S. & Uzsoy R. (2007). A genetic algorithm for minimizing maximum lateness on parallel identical batch processing machines with dynamic job arrivals and incompatible job families. *Computers & Operations Research, 34*, 3016-3028.

[20]. Morton T.E. and Pentico D.W. (1993). *Heuristic Scheduling Systems: With Applications to Production Systems and Project Management.* John Wiley and Sons.

[21]. Pfund M., Fowler J.W., Gadkari A. and Chen Y. (2008). Scheduling jobs on parallel machines with setup times and ready times. *Computers and Industrial Engineering*, 764-782.

[22]. Pinedo M. (1995). *Scheduling: Theory, Algorithms and Systems.* Prentice Hall.

[23]. Pinedo M. (2002). *Scheduling Theory, Algorithms and Systems, (2nd ed).* Prentice Hall.

[24]. Schaller J.E. (2014). Minimizing total tardiness for scheduling identical parallel machines with family setups. *Computers & Industrial Engineering,72*, 274-281.

[25]. Shih W.L., Zne J.L., Kuo C.Y., Chung C.L. (2011). Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates. *Computers & Operations Research 38*, 809–815.

[26]. Vepsalainen A.P.J and Morton T.E. (1987). Priority rules for job shops with weighted tardiness cost., 33(8), 1035-1047. *Management Science 33(8)*, 1035-1047.

[27]. Volgenant A. and Teerhuis E. (1999). Improved heuristics for the n-job single-machine weighted tardiness problem. *Computers and Operations Research, 26(1)*, 35-44.

[28]. Zhou H., Cheung W. and Leung L.C. (2009). Minimizing weighted tardiness of job-shop scheduling using a hybrid genetic algorithm. *European Journal of Operational Research, 194 (3)* , 637-649.

[29]. Zhu X., Wilhelm W.E. (2006). Scheduling and lot sizing with sequence-dependent setup: a literature review. *IIE Transactions*, 987-1007.