PARALLEL MACHINE EARLINESS-TARDINESS SCHEDULING
Comparison of Two Metaheuristic Approaches

Marcin Bazyluk, Leszek Koszalka
Chair of Computer Systems and Networks, Wroclaw University of Technology, Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland

Keith J. Burnham
Control Theory and Applications Centre, Coventry University, Coventry, U.K.

Keywords: Task scheduling, parallel machines, heuristics, genetic algorithms, tabu search.

Abstract: This paper considers the problem of parallel machine scheduling with the earliness and tardiness penalties (PMSP_E/T) in which a set of sequence-independent jobs is to be scheduled on a set of given machines to minimize a sum of the weighted earliness and tardiness values. The weights and due dates of the jobs are distinct positive numbers. The machines are diverse - each has a different execution speed of the respective jobs, thus the problem becomes more complex. To handle this, it two heuristics are employed, namely: the genetic algorithm with the MCUOX crossover operator and the tabu search. The performances of the both approaches are evaluated and their dependency on the shape of the investigated instances examined. The results indicate the significant predominance of the genetic approach for the larger-sized instances.

1 INTRODUCTION

The problem of job scheduling on parallel machines (PMJS) has been considered recently in many scientific and industrial communities. In the classical example of this type of problem the collection of independent jobs is investigated. Each of the tasks is to be assigned to the single set of the available machines which may be identical or different. The problem considered here extends the PMJS with the weighted earliness and tardiness penalties calculated for the execution of a job before or after its due date. A single job consists of the following parameters, namely its weight, also defined as the importance, and its due date. Baker and Scudder (Baker and Scudder, 1990) provide a survey on the job scheduling problem involving a due date determination. In this paper, the due date for the each job is considered as a specific moment in the time. If a given job is accomplished at its due date, the requirements are fulfilled and the penalty is not subjected. The specification of a single machine is implemented as a vector describing its processing speed of the individual jobs. It is to be noted, that the identical machines are characterized by the identical vectors.

In the case, when the due dates of all of the jobs are set to 0 (thus all the jobs require the fastest possible execution with no earliness penalty) and their weights are equal, the problem considered is NP-hard (Du and Leong, 1990). It is to be remarked, that when the weights are arbitrary positive numbers, the task becomes NP-hard in a strong sense. On the other hand, the problem considered here may be thought of as even more complex. Kanet examined the problem of minimizing the total weighted earliness with the tardiness on a single machine and common due dates in (Kanet, 1981). He proposed the algorithm characterized by a polynomial complexity. His work was extended by Hall and Posner to the case of several identical machines in (Hall and Posner, 1991). This paper forms a further extension with the due dates allowed to be different.

The example solution to the problem considered, is a schedule of all jobs that are assigned to the separate machines in a set of queues - one for each machine. The premise is to minimize the sum of the penalties for all the jobs. The penalty for a single job is calculated according to the Equation (1) where p_i and w_i are the penalty and the weight of the job i, respectively. $e_i = max\{0, d_i - c_i\}$ denoted the earliness of the job i and $t_i = max\{0, c_i - d_i\}$ is the tardiness of the job i with d_i being its corresponding due date

Bazyluk M., Koszalka L. and J. Burnham K. (2008).
PARALLEL MACHINE EARLINESS-TARDINESS SCHEDULING - Comparison of Two Metaheuristic Approaches.
In *Proceedings of the Fifth International Conference on Informatics in Control, Automation and Robotics - ICSO*, pages 80-85
DOI: 10.5220/0001476500800085

and c_i its completion time (the time instant when the processing of a job has been finished). The following formula is introduced, see (2), accounting for the total earliness and tardiness penalty (p), where I denotes the set of jobs.

$$p_i = w_i e_i + w_i t_i \tag{1}$$

$$p = \sum_{i \in I} w_i e_i + \sum_{i \in I} w_i t_i \tag{2}$$

2 PROBLEM FORMULATION

Let the problem of the scheduling of the N independent lobs on a set of M available parallel machines be considered. Each job may be completed by any of the available machines, however the time required differs, in general, depending on the machine chosen. If the job has been accomplished too early or too late, the penalty is calculated for the earliness or tardiness, respectively, according to the difference between the due date of the job and the real execution time. It is noted, that the penalty is also proportional to the weight of the particular job. By considering the possible decisions during the job scheduling procedure, two optimization techniques have been developed to minimize the total penalty for all the jobs. The mathematical model presented here is a modification of that proposed by Cao, Chen and Wan in (Cao et al., 2005). The next two subsections introduce the nomenclature and the definitions required, to formulate the problem.

2.1 Known Parameters

i, j $= 1,2,\ldots,N$, job indeces
m $= 1,2,\ldots,M$, machine index
z_{im} processing time of the i-th job on the m-th machine, $z_{im} = \{1,2,\ldots,10\}$
w_i weight of the i-th job , $w_i = \{1,2,\ldots,10\}$
d_i due date of the i-th job , $d_i = \{1,2\ldots,5\}$
c_i execution time of the i-th job
e_i earliness of jo i, $e_i = max\{0, d_i - c_i\}$
t_i tardiness of jo i, $t_i = max\{0, c_i - d_i\}$

2.2 Decision Variables

$$x_{ijm} = \begin{cases} 1 & \text{if job } j \text{ follows job } i \text{ on machine } m, \\ 0 & \text{otherwise,} \end{cases}$$

$$y_{im} = \begin{cases} 1 & \text{if job } i \text{ is executed by machine } m, \\ 0 & \text{otherwise,} \end{cases}$$

$i = 0,1,\ldots,N,\ j = 1,\ldots,N,$
$j \neq i, m = 1,\ldots,M$

With the decision variables defined, the following recursive formula (Equation (3)) is proposed for the calculation of the moment of the job completion. The job completion is defined as the sum of its processing time together with the moment when the job that is executed directly before the one being considered here, is completed.

$$c_j = \sum_{i=0}^{N} \sum_{m=0}^{M} x_{ijm} c_i + z_{jm} \tag{3}$$

2.3 Assumptions

$$\text{Min } T_c = \sum_{i=1}^{N} w_i (e_i + t_i) \tag{4}$$

$$\sum_{i=1, i \neq j}^{N} \sum_{m=1}^{M} x_{ijm} = 1 \tag{5}$$

$$\sum_{i=1, i \neq j}^{N} x_{ijm} = y_{jm}, \tag{6}$$

$$\sum_{j=1, j \neq i}^{N} x_{ijm} \leq y_{im} \tag{7}$$

$$c_j + A(1 - x_{ijm}) \geq c_i + z_{jm} \tag{8}$$

In all the assumptions introduced, the parameters are defined as follows:

$$i = 1,2,\ldots,N,\ j = 1,2,\ldots,N,\ i \neq j,$$

$$m = 1,2,\ldots,M,\ c_i > 0,\ c_j > 0$$

In the model presented, Equation (4) describes the objective function as the sum of the penalties for all the jobs which are to be minimized. The following equations defines the restrictions to which the decision variables and the parameters are subjected. Equation (5) ensures that, first, every job is processed on a single machine and, second, that its execution is not divided into separate parts. Equation (6) imposes a restriction that the j-th job must immediately follow any other job on the m-th machine (or be left in the first position in the case when i = 0). Thereby, the intervals in the machine usage are not allowed. Equation (7) states that, if the i-th job is processed on the m-th machine, it will be immediately followed by at most one other job on this machine. Therefore, the jobs can be executed one at the each time instant. Equation (8) expresses the finite completion time of each job, where the scalar A is a large positive number.

Due to the evident complexity of the problem considered, the investigations are restricted to the class of heuristic algorithms. In Sections 3–4 two heuristic based approaches are presented.

START

- Find the first solution.

LOOP

- Find the best solution which is not tabu.
- Update the global best solution.
- Add a rule to the tabu list.
- If the fixed number of the iterations are completed, STOP;
  else if the fixed number of the iterations without the improvement regarding the the best global solution is completed, STOP;
  else repeat LOOP.

Figure 1: Tabu search algorithm.

3 TABU SEARCH ALGORITHM APPROACH

The tabu search algorithm (TSA) was developed mainly due to the work of the Polish mathematicians, namely E. Nowicki and C. Smutnicki (Nowicki and Smutnicki, 2005). It belongs to the class of local search algorithms, where a concept of the local neighbourhood of a given solution is considered. The local neighbourhood is defined as a set of solutions that are different from the basic solution with respect to a single attribute. Many different types of neighbourhood can be found in the literature, however the swap and insert types, are considered in this paper.

TSA consists of the subsequent steps to determine the best representative of the neighbour solutions set, where a single step is conducted in each iteration. It may occur that one of the neighbour solutions is superior in respect to the basic solution (i.e. the solution which the neighbourhood is considered at the given iteration). In this case, the step is made towards the direction of the newly found neighbourhood solution. This, therefore, allows the algorithm to alleviate the possibility to be trapped in local extremes of the objective function considered. To avoid repetitive steps between the same solutions in the adjacent extremes, the tabu list consisting of the assumed *a priori* number of the rules regarding the quality of the solutions visited recently, is proposed. Figure 1 provides a more formal description of the proposed algorithm.

There exists, however, an exception when the solution found is superior to the best one to date. In this case the algorithm steps towards its direction unconditionally and the tabu list is erased. It is remarked, that a considerable amount of time is required to estimate even a single solution in each iteration of the algorithm, for the instances relatively large. This leads

START

- Generate the first population.

LOOP

- Find the best chromosome in the current population.
- Update the global best chromosome.
- Choose the representing chromosomes for the crossover.
- Create a new population.
- Execute a swap mutation with the fixed probability.
- Execute a bit mutation with the fixed probability.
- If the fixed number of the iterations are completed, STOP;
  else if the fixed number of the iterations without the improvement regarding the the best global solution is completed, STOP;
  else repeat LOOP.

Figure 2: Genetic algorithm.

to the conclusion that the quality of the first generated solution from which the algorithm starts is essential. In this paper, a modified algorithm is suggested, to handle this task.

The swap move is carried out via the exchange of the positions between two randomly chosen jobs taken from the list assigned to a single machine. Insert one removes a random job from a one machine and allocates it on the second machine, directly before randomly chosen job from its list.

4 GENETIC ALGORITHM APPROACH

For more details on the subject of the genetic algorithms (GAs), see (Schmitt, 2000) and (Davis, 1991). Due to the fact that many different approaches have been described in the literature for solving the problem considered, this section presents the exact parameters of the algorithm implemented.

The more formal definition of the algorithm proposed is presented in figure 2. The crossover operator is the basis of the genetic algorithms, hence, it is essential to focus on its proper implementation. The issue that was shown experimentally in (Bazyluk et al., 2006) for a simpler type of job scheduling problem is that the utilization of a popular PMX leads to an impossibility to obtain satisfying results. Therefore, the multi-component uniform order-based (MCUOX) crossover operator proposed by Sivrikaya and Ulusoy in (Sivrikaya-Serifoglu and Ulusoy, 1999) was chosen for the purpose of this paper.

It is noted, that in the scheme chosen a single gene

accommodates both the object and the associated selection associated with this object. In the case of the problem considered, this refers to the situation where each gene corresponds to the job-machine pair. The construction of a descendant from two parent chromosomes is presented in Figure 3.

START

- With the first position on the parents and the descendant.

LOOP

- Choose one parent randomly.
- Find the first job corresponding to the chosen parent which has not been assigned to the descendant.
- If the machine assigned to the job is the same for both parents, make the same selection for the job;
  else choose one of the machines from the parents randomly and assign the job to it.
- Assign the job-machine pair to the first empty position of the descendant.
- If all genes of the descendant chromosomes are set, STOP;
  else proceed to the next gene on the descendant and repeat LOOP.

Figure 3: MCUOX crossover algorithm.

The mutation implemented comprises two mechanisms. The first, randomly selects two positions on a chromosome and exchanges their contents. The second, incorporates the reassignment of the machine to a randomly chosen job in a chromosome. It is to be noted, that this operation may lead to the same machine selection as before.

The selection of the chromosomes for the crossover from a population builds on a rule that the probability of choosing a chromosome to be a parent in the next generation is proportional to its fitness. The value of the fitness function is for a given chromosome is defined by Equation (9) where K is the size of the population and $F_k = CT_c(k)^{-1}$ the fitness function of the chromosome k that is inversely proportional to the objective function with C being a fixed constant.

Table 1: Experiment parameters.

tabu list size (TSA)	10
population size (GA)	25
swap mutation probability (GA)	0,1
bit mutation probability (GA)	0,05
crossover probability (GA)	0,9

$$P_k = \frac{F_k}{\sum_{k=1}^{K} F_k} \tag{9}$$

The appropriate tuning of the parameters of the GA is one of the crucial issues that influences significantly its efficiency and effectiveness, see (Grefenstette, 1986) for more detailed discussion. For this purpose, in the problem considered, a meta-genetic algorithm with a regular PMX operator was additionally implemented.

5 NUMERICAL ANALYSIS

5.1 Generation of Test Instances

Both implemented algorithms were validated for the same instances of the various sizes considered. Due to the fact that no benchmark problems were found in the literature, such as in particular the presented shape, the test benchmarks were generated randomly. After the tuning of the parameters the following values, presented in Table 1, were set.

5.2 Research Results

The implemented algorithms were evaluated on a IBM-compatible machine, equipped with the Intel Pentium M 740 1,73GHz processor and 512 Mbytes of RAM. The first experiment was conducted for a set of 100 small instances comprising of 10 jobs and 3 machines. The evolution of the average objective function, denoted (T_c), for the both algorithms is depicted in Figure 4. It is observed, that TSA begins with the solution that is nearly twice as large as T_c, however it relatively quickly overtakes GA (first intersection of the graphs in Figure 4). This property can be noted for all the instances consisting of the job sizes 100 or less, in general. In the case of TSA, as opposed to GA, it seems less difficult to determine the optimal solution, but, on the other hand, significantly more computation time is required. This is due to the property of the slow movement across the arguments area of the TSA based algorithm. It is to be remarked, that the aforemention drawback of TSA is not significant for the relatively small instances. It can be expected that the GA would obtain the solution of a similar quality as the TSA after a certain number of iterations, which can be observed in Figure 5 (the second intersection).

In the following investigation the impact is placed to examine the two aforementioned intersection

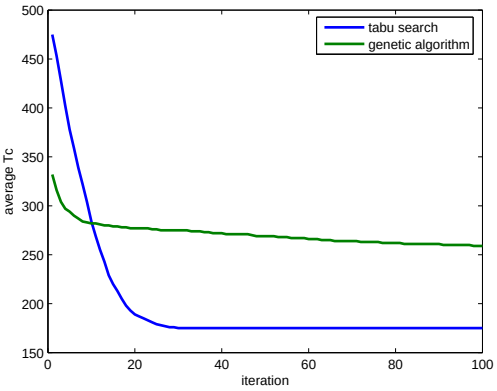

Figure 4: Objective function evolution.

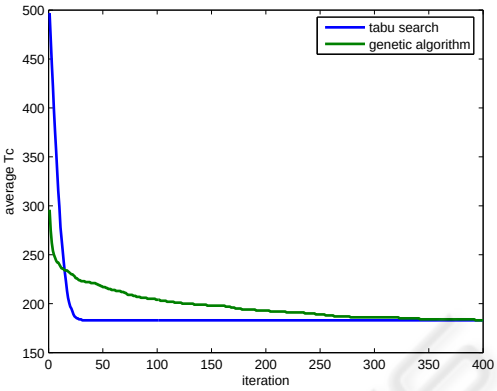

Figure 5: Objective function evolution.

points. In all the experiments conducted the three areas of the objective function evolution were observed:

1. predominance of GA in the first iterations (up to the 1st intersection)
2. predominance of TSA for a given set of iterations (up to the 2nd intersection)
3. predominance of GA again (after the 2nd intersection)

The purpose of the following investigation is to analyze the location of the both intersection points with the accordance to the size of the particular instance.

Analogously to the results presented previously, see Figures 4-5, the first intersection is observed relatively quickly (it is a one of the first iterations) when the instances smaller than 100 jobs and 10 machines are considered. Therefore, the usage of the TSA has proven to be a considerably better choice, when the relatively short calculation time is to be expected. On the other hand, in the cases when the location of the optimum solution is of the prime interest, the GA algorithm is the more appropriate choice. The radical changes are observed for the relatively large instances of 100 jobs or more. In these cases, the TSA is not able to move sufficiently fast across the arguments area and, thus, the first intersection point, after a larger number of iterations, is reached. The example average objective function evolution for 100 instances of 200 jobs and 10 machines is presented in Table 2. It is noted, that with the growth of the instance size, the corresponding time required for a completion of a single iteration increases, accordingly. After 10 iterations it may be readily observed that the TSA evolved from 59509 to 59137 with the decrease of $0,63\%$. On the other hand, the utilization of the GA leads to the decrease from 55828 to 54381 at the ratio of $2,59\%$. The time required for carrying out the experiment in the same configuration setup, for the time

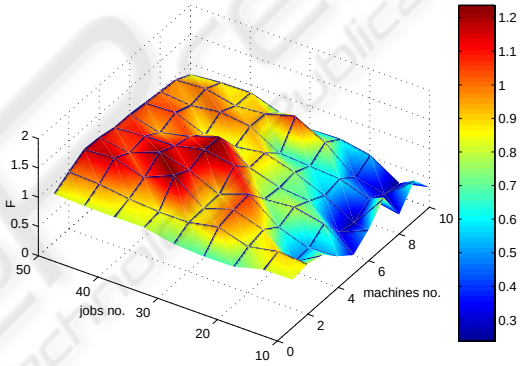

Figure 6: Effectiveness proportion of GA to TSA for different instance sizes.

long enough to reach the first intersection point, was approximately calculated to be few hours.

Table 2: Objective function evolution.

Iteration	T_c (TSA)	T_c (GA)
1	59509	55828
2	59470	55016
3	59396	54859
4	59346	54758
5	59315	54670
6	59271	54539
7	59243	54529
8	59210	54455
9	59174	54410
10	59137	54381

The further work will comprise the investigation of the three-dimensional graphs to illustrate the comparison results as a function of the both attributes of the

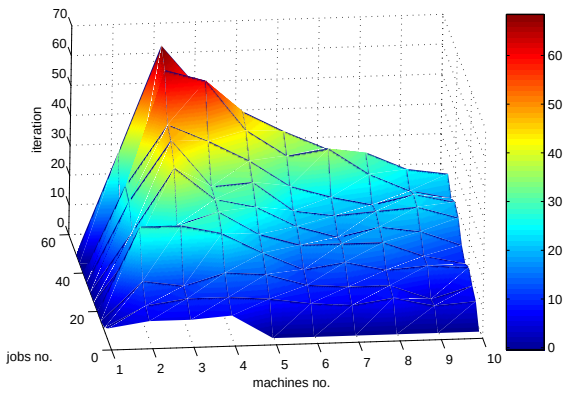

Figure 7: Location of first intersection point for different instance sizes.

instance size i.e. the number of the jobs and the number of the machines. Figure 6 illustrates the proportion, denoted (F), of the objective function value obtained via the GA and the TSA. The experiment was carried out for 100 instances of the each size taken from the set of 1 to 10 machines and 10 to 50 jobs. Both algorithms were investigated for 25 iterations. The value of $F > 1$ on the graph corresponds to the situations where the efficiency of GA was superior to TSA, $F < 1$ otherwise.

Figure 7 presents the average number of iterations, when the first intersection was observed. The first intersection point can be interpreted as a point starting from which the GA is superior over the TSA.

6 CONCLUSIONS

In the paper the problem of the parallel machine job scheduling with the weighted earliness and tardiness has been addressed. Two heuristic algorithms, that proven to be efficient, have been proposed and numerically validated. The investigation of the sensitivity of both approaches as a function of the size of the problem instance, has been carried out. The results obtained suggest that the TSA is appropriate to handle the relatively small and medium instances. On the other hand, utilization of the GA coupled with MCUOX crossover operator, becomes more beneficial with the increase of the problem instances. An important property was observed, namely a significant deterioration of the efficiency of the TSA for instances containing $2-5$ machines in comparison with other values was noted, see Figures 6-7. Considering these figures from the instance of (1 machine, 10 jobs) to (10 machines, 50 jobs) a constant improvement of the GA in comparison with the TSA up to the point of its predominance, can be seen. It has been shown experimentally that the point is located in the vicinity of the instance size of 100 and 200 jobs, see (Figure 5). Furthermore, the execution time of GA with MCUOX increases with a decreasing rate as a function of the increasing problem size whilst the increasing rate is observed for the TSA.

The problem extension could consist of the sequential dependency of jobs and the possibility of introducing the idle time intervals between subsequent execution of jobs. Further improvement of the proposed heuristics can be achieved by considering a hybrid algorithm that inherits the advantages of both approaches i.e. the GA and the TSA.

REFERENCES

Baker, K. and Scudder, G. (1990). Sequencing with earliness and tardiness penalties: a review. *Operations Research*, 38:22–36.

Bazyluk, M., Koszalka, L., and Burnham, K. (2006). Using heuristic algorithms for parallel machines job scheduling problem. *Computer Systems Engineering*.

Cao, D., Chen, M., and Wan, G. (2005). Parallel machine selection and job scheduling to minimize machine cost and job tardiness. *Computers and Operations Research*, 32:1995–2012.

Davis, L. (1991). *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York.

Du, J. and Leong, J. (1990). Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research*, 483–495:1990.

Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems Man and Cybernetics*, SMC-16:122–128.

Hall, N. and Posner, M. (1991). Earliness-tardiness scheduling problems, i: weighted deviation of completion times about a common due date. *Operations Research*, 39:836–846.

Kanet, J. (1981). Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics Quarterly*, 28:643–651.

Nowicki, E. and Smutnicki, C. (2005). Metaheuristic optimization via memory and evolution. *Kluwer Academic Publishers*, pages 165–190.

Schmitt, L. (2000). Theory of genetic algorithms. *Theoretical Computer Science*, 259:1–61.

Sivrikaya-Serifoglu, F. and Ulusoy, G. (1999). Parallel machine scheduling with earliness and tardiness penalties. *Computers & Operations Research*, 26:773–787.