Acta Polytechnica **61**(2):307–312, 2021
https://doi.org/10.14311/AP.2021.61.0307

PARALLEL MACHINE SCHEDULING WITH MONTE CARLO TREE SEARCH

Anita Agárdi*, Károly Nehéz

University of Miskolc, Faculty of Mechanical Engineering and Informatics, Institute of Information Science, Miskolc-Egyetemváros, 3515, Hungary

* corresponding author: agardianita@iit.uni-miskolc.hu

Abstract. In this article, a specific production scheduling problem (PSP), the Parallel Machine Scheduling Problem (PMSP) with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times is presented. In this article after the introduction and literature review the mathematical model of the Parallel Machines Scheduling Problem with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times is presented. After that the Monte Carlo Tree Search and Simulated Annealing are detailed. Our representation technique and its evaluation are also introduced. After that, the efficiency of the algorithms is tested with benchmark data, which result, that algorithms are suitable for solving production scheduling problems. In this article, after the literature review, a suitable mathematical model is presented. The problem is solved with a specific Monte Carlo Tree Search (MCTS) algorithm, which uses a neighbourhood search method (2-opt). In the article, we present the efficiency of our Iterative Monte Carlo Tree Search (IMCTS) algorithm on randomly generated data sets.

Keywords: Parallel-machine scheduling, Monte Carlo Tree Search.

1. Introduction

A cost-efficient production is one of the main goals of manufacturing companies, because production efficiency means higher profits for the company. A cost-efficient production means that as many jobs as possible are processed on time at the lowest cost. In this article, a specific production scheduling problem, the Parallel Machine Scheduling Problem with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times is presented. In this specific problem, m jobs must be distributed among n machines. Setup time means transition time between jobs. The due date means the day by which the job has to be done. Maintenance time means a time window when the production stops because there is maintenance. In our problem, the objective function is the minimization of the setup times and maximization of the number of created jobs. The Production Scheduling Problems are NP hard, therefore, it is necessary to apply some heuristics to their solution. With heuristics, of course, there is little chance of finding a global optimum, but we can get a good local optimum within a reasonable amount of time. In this article, a specific Monte Carlo Tree Search (MCTS) algorithm is applied to the problem.

The paper is organized as follows: after the literature review, the mathematical model of our specific production scheduling problem is described. In section 4, the Monte Carlo Tree Search is detailed. In section 5, the representation and evaluation of Parallel Machine Scheduling Problem with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times are presented. In section 5, our Iterative Monte Carlo Tree Search algorithm is also detailed. After that, the test results are discussed. In section 7, the conclusion and remarks are presented.

2. Literature review

Production Scheduling Problem (PSP) [1, 2] is a common problem in manufacturing systems. The problem lies in scheduling n jobs between m machines. Over time, many types of PSP have evolved. In the following paragraph, some types of production scheduling problems are presented based on the literature.

In the case of Parallel Machine Scheduling Problem (PMSP) [3], m jobs must be distributed among n machines. Fuzzy Parallel Machine Problem [4] has some fuzzy parameter (setup time, capacity constraint, time windows etc.). Machine eligibility restriction [5] states if a machine is capable of processing a job (operation). In the case of production scheduling, the machines can be identical [6] or uniform [7]. The job can have due date [8], release time [9] and time window [10]. The due date means the day by which the job has to be done. Release date means that the job needs to be started at least in this predetermined time. The time window is the combination of the release time and due date. Jobs with priority level [11] should be created as soon as possible. The job processing time can be not only fixed but also variable [12]. The PMSP [9] can have several objective functions, for example, the minimization of the setup times [13], the minimization of the total earliness and tardiness time [14] or objective function, which takes into account environmental impacts (for example green scheduling problem) [7]. The constraints and components can

Machine	Maintenance time
Machine 1	$[1,4],[7,8]$
Machine 2	$[4,5]$
Machine 3	$[1,2]$
Machine 4	$[2,4]$

Table 1. The components of the machines

Job	Duration	Due date
Job 1	1	1
Job 2	1	9
Job 3	4	6
Job 4	1	2
Job 5	3	9
Job 6	1	4
Job 7	4	10

Table 2. The components of the jobs

also be used together, for example, in article [15] the authors also use release and due dates, family set-ups, etc.

3. Parallel Machine Scheduling Problem with Job and Machine Sequence Dependent Setup Times, Due Dates and Maintenance Times

In this specific problem, m jobs must be distributed among n machines. Setup time means the transition time between jobs. The due date means the day by which the job has to be done. Maintenance time means a time window when the production stops because there is maintenance. In our problem, the objective function is the minimization of the setup times, and maximization of the number of created jobs. In the following, a simple example and its solution are presented, after that, the mathematical model of our problem is detailed.

Table 1 illustrates the components of the machines, Table 2 presents the components of the jobs, while in Table 3, the setup times of Machine 1 can be seen.

Figure 1 presents the results of the Parallel Machine Scheduling Problem with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times.

Figure 1 presents the solution of the problem. As shown on the Gantt chart, 4 machines create 7 jobs. Job 1 and Job 2 belongs to Machine 1. Machine 2 has Job 3. Machine 3 has two Jobs: Job 4 and Job 5. Machine 4 also has two Jobs: Job 6 and Job 7.

3.1. The mathematical model

Assume that our problem contains n jobs and m machines. Each job has $m*(n+1)$ setup times because the setup time is job- and machine-dependent. Each

	R0	J1	J2	J3	J4	J5	J6	J7
R0	0	0	1	2	2	3	1	1
J1	1	0	1	2	2	3	1	1
J2	2	3	0	4	1	2	1	1
J3	1	2	1	0	1	2	1	2
J4	1	2	1	2	0	1	2	1
J5	1	2	3	1	2	0	1	1
J6	0	1	2	1	3	1	0	4
J7	1	0	1	2	1	2	1	0

Table 3. The setup times for Machine 1

job has a processing time and a due date. Each machine has (any) maintenance times. Each maintenance time means a time window, i.e.:$[e,l]$. One machine can only process one job at a time. The notations and their definitions are illustrated in Table 4.

The objective function is the minimization of the setup times:

$$Z = min(Z_1 + Z_2) \tag{1}$$

where Z_1 means the job-job setup time:

$$Z_1 = \sum_{i=1}^{m}\sum_{j_0=1}^{n}\sum_{j_1=1}^{n} ST_{i,j_0,j_1} * x_{i,j_0,j_1} \tag{2}$$

and Z_2 means the machine-job setup time:

$$Z_2 = \sum_{i=1}^{m}\sum_{j_1}^{n} ST_{i,i,j_1} * x_{i,i,j_1} \tag{3}$$

Constraint 1: Each job can be assigned to one machine:

$$\sum_{i=1}^{m}\sum_{j_0=1}^{n} x_{i,j_0,j_1} + \sum_{i=1}^{m} x_{i,i,j_1} = 1 \forall j_1 = 0...n \tag{4}$$

Constraint 2: Production continuity:

$$\sum_{i=1}^{m}\sum_{j_0=1}^{n} x_{i,j_0,j_1} = \sum_{i=1}^{m}\sum_{j_0=1}^{n} x_{i,j_1,j_0} \forall j_1 = 0...n \tag{5}$$

Constraint 3: Due date constraint:

$$y_j \le DD_j \forall j = 1...n \tag{6}$$

Constraint 4: Maintenance times:

$$y_{j_0} - PT_{j_0} \le e, y_{j_0} \le le \in MT_{i,k} \tag{7}$$

$$x_{i,j_1,j_0} = 1 j_1 = 1...n \forall j_0 = 1...n \tag{8}$$

occurs in at least one case

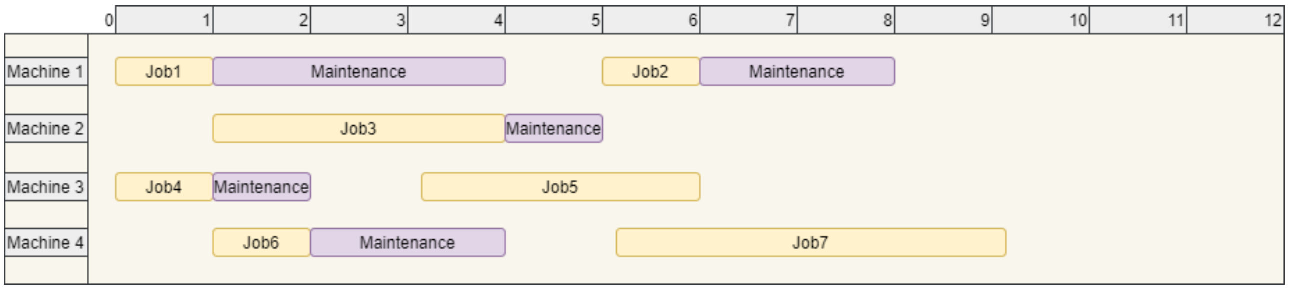


Figure 1. The Gantt chart of the production scheduling

Notation	Definition
m	number of machines
n	number of jobs
PT_j	processing time of job j
ST_{i,j_0,j_1}	setup time of job j_1 for machine i after job j_0
ST_{i,i,j_1}	setup time of job j_1 for machine i
MT_i	maintenance times of machine i, where $MT_{i,k} = [e, l]$.
DD_j	due date of job j
y_j	ending time of job j
$x_{i,j_0,j_i} \in 0, 1$	decision variable

Table 4. The notations and their definitions

4. Monte Carlo Tree Search

The Monte Carlo Tree Search (MCTS) algorithm creates a tree structure to solve a specific problem. During the algorithm, the search tree grows gradually and asymmetrically. The algorithm starts at the root node and repeatedly picks a random child as a successor until it reaches a leaf node. The MCTS guides the search towards better candidate successors. It uses a balance between exploration and exploitation. The exploration means that the algorithm needs randomness to explore the search space. The exploitation means that the MCTS takes an advantage of the best option we know. The algorithm repeats the following four phases [16]:

(1.) Selection: The algorithm starts from the root node and recursively chooses the best child based on the UCB formula until a leaf node is reached. This phase can be seen in Figure 2.

(2.) Expansion: When a leaf is reached, the algorithm further expands, unless there is no possible leaf. This phase can be seen in Figure 3.

(3.) Simulation: Run playout from the leaf. This phase can be seen in Figure 4.

(4.) Backpropagation: Update the evaluation (fitness) values from the result. This phase can be seen in Figure 5.

The iteration of these four steps is called simulation. The UCB formula provides a balance between exploration and exploitation:

$$UCB_j = X_j + C * \sqrt{\frac{log_{10}}{N_j}} \tag{9}$$

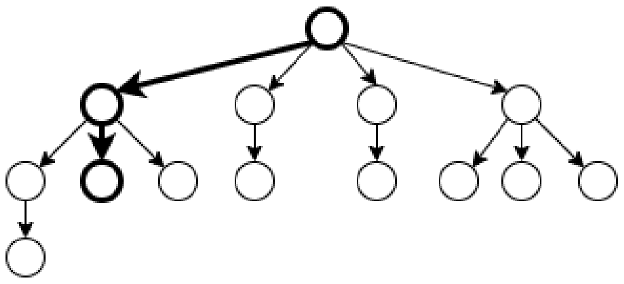


Figure 2. Monte Carlo Tree Search: Selection

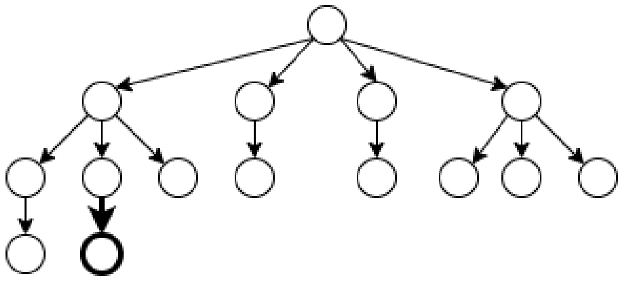


Figure 3. Monte Carlo Tree Search: Expansion

where X_j is the average win-rate (or fitness) of the j-th child, C is the UCB constant, most of the time $C = 2$. N_j is the visit count of a child j, and N is the visit count of the parent.

Algorithm 1 presents the base Monte Carlo Tree Search technique. The Tree Policy means selecting or creating a leaf node from the nodes already contained within the search tree (i.e., the selection and expansion process). The Default Policy means a simulation (i.e., playing out the domain from a given non-terminal state to the estimating value). v_0 indicates the root

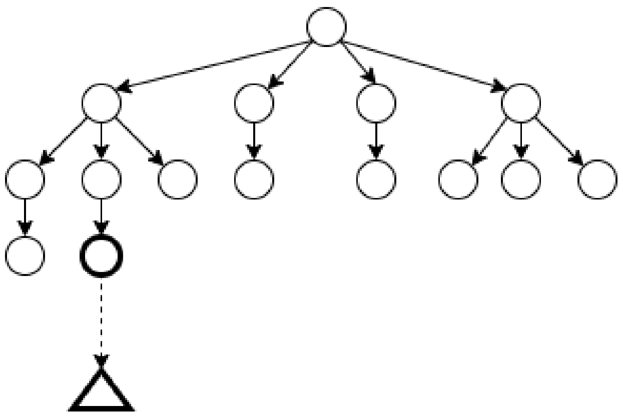


Figure 4. Monte Carlo Tree Search: Simulation

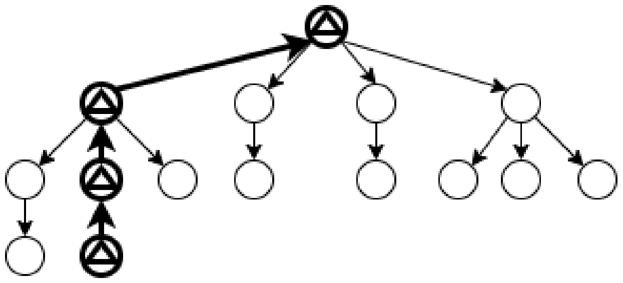
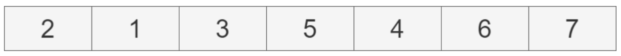


Figure 5. Monte Carlo Tree Search: Backpropagation

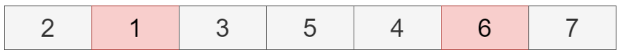
Algorithm 1 Monte Carlo Tree Search[17]

```
1: procedure Monte Carlo Tree Search
2:     Create root node $v_0$ with state $s_0$
3:     while Termination criterion is not met do
4:         $v_l \leftarrow TREEPOLICY(v_0)$
5:         $\delta \leftarrow DEFAULTPOLICY(s(v_l))$
6:         $BACKUP(v, \delta)$
       return $BESTCHILD(v_0)$
```



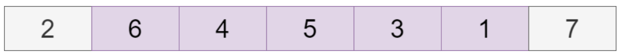
2	1	3	5	4	6	7

Figure 6. Representation of our problem



2	1	3	5	4	6	7

Figure 7. Before 2-opt



2	6	4	5	3	1	7

Figure 8. After 2-opt

Algorithm 2 Iterative Monte Carlo Tree Search

```
1: procedure Monte Carlo Tree Search
2:     while Term. criterion is not met do
3:         Create root node $v_0$ with state $s_0$
4:         while Term. criterion is not met do
5:             $v_l \leftarrow TREEPOLICY(v_0)$
6:             $v \leftarrow DEFAULTPOLICY(s(v_l))$
7:             $BACKUP(v, \delta)$
8:             while Term. criterion is not met do
9:                 Application of 2-opt on $v$
10:                $BACKUP(v, \delta)$
11:        $V \leftarrow BESTCHILD(v_0)$
12:        while Term. criterion is not met do
13:            Application of 2-opt on $V$.
       return $V$
```

node, and its state is s_0. v_l is the last node reached during the tree policy. Its corresponding state is indicated with s_l. δ means the gain of the terminal state reached with a random simulation from state s_l.

5. Algorithm for Parallel Machine Scheduling Problem with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times

Representation of our problem Our problem is represented with a permutation (Figure 6). In our approach, the permutation means a given sequence of job indices. Each element of the permutation is assigned to the first machine. If the constraints of any element (any job) of the permutation are not satisfied, then the other items (jobs) will be scheduled to the next machine.

Edge swapping (2-opt) Edge swapping is the swapping of elements between two randomly selected permutation items.

Iterative Monte Carlo Tree Search (IMCTS) Our algorithm (Algorithm 2) is the modification of the Monte Carlo Tree Search.

We start from a random state (initial permutation). Switching from one state to another means an exchange (2-opt) operation. So by state, we mean a particular permutation (solution).

Our MCTS is an iterative algorithm, which means, that the tree is re-built iteratively. After a tree is built, in the result solution, the 2-opt operation is also performed.

6. Test results

The test results are described in this section. Our data set was artificially generated. A total of 7 different data sets were used. The first five data sets have 17 jobs and 5 machines, and the last two have 25 jobs and 10 machines. The best, worst, and the average results of the runs are illustrated in the tables, where P. means the problem, F. means the fitness value, T. means

	Best res.		Average res.		Worst res.	
P.	**F.**	**T. (s)**	**F.**	**T. (s)**	**F.**	**T. (s)**
1	124	16	135	17.6	142	19
2	100	20	112.4	21.6	118	23
3	127	18	134	19	147	20
4	114	20	122.6	21	137	23
5	96	20	109.8	20.4	118	22

Table 5. The test result for IMCTS in the case of 17 jobs and 7 machines

	Best res.		Average res.		Worst res.	
P.	**F.**	**T. (s)**	**F.**	**T. (s)**	**F.**	**T. (s)**
1	247	90	288.6	93.4	312	99
2	242	88	262.8	92.2	314	96

Table 6. The test result for IMCTS in the case of 25 jobs and 10 machines

the time in sec and res. means the results. Based on the Table 5, for smaller problems, our IMCTS algorithm proved to be efficient. The situation is similar for a larger data set, this is illustrated in Table 6. In summary, based on the test results, the IMCTS algorithm proved to be efficient based on our entire data set.

7. Conclusion

In this article, a specific production scheduling task, the Parallel Machine Scheduling Problem with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times was introduced. After the introduction, the literature on production scheduling was presented. In section 3, the mathematical model of the Parallel Machine Scheduling Problem with Job and Machine Sequence Setup Times, Due Dates and Maintenance Times was detailed. After that, the Monte Carlo Tree Search algorithm is detailed. In section 6, our algorithms, their representation and evaluation were presented. In section 7, the efficiency of the Iterative Monte Carlo Tree Search was tested with artificially generated datasets. Based on the test results, the efficiency of our representation and evaluation is demonstrated. In this article, we presented an example of using the MCTS for scheduling. The authors believe that the IMCTS method they developed can be effective for additional scheduling tasks. Our further research direction is the development and testing of other hybrid algorithms in the field of optimization.

References

[1] J. Y. Leung. *Handbook of scheduling: algorithms, models, and performance analysis.* CRC press, 2004.

[2] M. Pinedo. Scheduling: theory, algorithms, and systems. 5-th ed. cham, 2016.

[3] E. Vallada, R. Ruiz. A genetic algorithm for the unrelated parallel machine scheduling problem with sequence dependent setup times. *European Journal of Operational Research* **211**(3):612–622, 2011. https://doi.org/10.1016/j.ejor.2011.01.011.

[4] O. A. Arık, M. D. Toksarı. Multi-objective fuzzy parallel machine scheduling problems under fuzzy job deterioration and learning effects. *International Journal of Production Research* **56**(7):2488–2505, 2018. https://doi.org/10.1080/00207543.2017.1388932.

[5] G. Bektur, T. Saraç. A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server. *Computers & Operations Research* **103**:46–63, 2019.

[6] D. Biskup, J. Herrmann, J. N. Gupta. Scheduling identical parallel machines to minimize total tardiness. *International Journal of Production Economics* **115**(1):134–142, 2008. https://doi.org/10.1016/j.ijpe.2008.04.011.

[7] H. Safarzadeh, S. T. A. Niaki. Bi-objective green scheduling in uniform parallel machine environments. *Journal of cleaner production* **217**:559–572, 2019. https://doi.org/10.1016/j.jclepro.2019.01.166.

[8] M. Ji, W. Zhang, L. Liao, et al. Multitasking parallel-machine scheduling with machine-dependent slack due-window assignment. *International Journal of Production Research* **57**(6):1667–1684, 2019.

[9] G. Centeno, R. L. Armacost. Minimizing makespan on parallel machines with release time and machine eligibility restrictions. *International Journal of Production Research* **42**(6):1243–1256, 2004. https://doi.org/10.1080/00207540310001631584.

[10] L. Nie. Parallel machine scheduling problem of flexible maintenance activities with fuzzy random time windows. In *Proceedings of the tenth international conference on management science and engineering management*, pp. 1027–1040. Springer, 2017.

[11] S. Ghanbari, M. Othman. A priority based job scheduling algorithm in cloud computing. *Procedia Engineering* **50**(0):778–785, 2012.

[12] A. Agnetis, J.-C. Billaut, S. Gawiejnowicz, et al. Scheduling problems with variable job processing times. In *Multiagent Scheduling*, pp. 217–260. Springer, 2014.

[13] K.-C. Ying, Z.-J. Lee, S.-W. Lin. Makespan minimization for scheduling unrelated parallel machines with setup times. *Journal of Intelligent Manufacturing* **23**(5):1795–1803, 2012. https://doi.org/10.1007/s10845-010-0483-3.

[14] Z. Wu, M. X. Weng. Multiagent scheduling method with earliness and tardiness objectives in flexible job shops. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **35**(2):293–301, 2005. https://doi.org/10.1109/tsmcb.2004.842412.

[15] M. Ciavotta, C. Meloni, M. Pranzo. Speeding up a rollout algorithm for complex parallel machine scheduling. *International Journal of Production Research* **54**(16):4993–5009, 2016.

[16] T.-Y. Wu, I.-C. Wu, C.-C. Liang. Multi-objective flexible job shop scheduling problem based on monte-carlo tree search. In *2013 Conference on Technologies and Applications of Artificial Intelligence*, pp. 73–78. IEEE, 2013. https://doi.org/10.1109/taai.2013.27.

[17] C. B. Browne, E. Powley, D. Whitehouse, et al. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* **4**(1):1–43, 2012. https://doi.org/10.1109/tciaig.2012.2186810.