Near-Optimal Solutions for Mold Constraints on Two Parallel Machines

Abir BEN HMIDA[1,4]*, Mahdi JEMMALI[2,3,4]

[1] Department of Information Systems, Faculty of Computer and Information Technologies at Khulais, Jeddah University, Saudi Arabia

[2] Department of Computer Science and Information, College of Science at Zulfi, Majmaah University, Majmaah, 11952, Saudi Arabia

[3] Mars Laboratory, University of Sousse, Sousse, Tunisia

[4] Department of Computer Science, Higher Institute of Computer Science and Mathematics of Monastir, Monastir University, Monastir, 5000, Tunisia
a.hhmida@uj.edu.sa (**Corresponding author*), m.jemmali@mu.edu.sa, mah_jem_2004@yahoo.fr

Abstract: This paper investigates the mold constraints for the two parallel machines problem. The goal is to minimize the makespan. The resource constraint can be defined as a mold. This constraint is also defined as jobs with conflict. Two jobs can't be executed by the two machines simultaneously. The problem is proved to be a non-deterministic polynomial NP-hard one. Different heuristics have been proposed in this paper to solve the studied problem. A novel meta-heuristics algorithm is developed to enhance the proposed heuristics. The computational results and discussions show the performance of the proposed solutions. A different class of instances is implemented to test the proposed heuristics. The results show that the developed heuristics reaches the optimal solution in 96.4% of instances. Therefore, the obtained results represent near-optimal solutions for the studied problem.

Keywords: Mold constraint, Heuristics, Makespan, Scheduling, Local-search.

1. Introduction

The mold constraints are used in several domains such as the wafer fabrication in a semiconductor plant.

In (Hong et al., 2009), the authors developed a mathematical model to describe the problem of two identical parallel machines in the production of wafer fabrication. The mold constraint is also taken into consideration. Three heuristics have been proposed in the latter work and compared with two existing metaheuristics.

Authors Hong et al. (2008) treated the problem of resource-constrained parallel machine scheduling with setup time in the practical context of microelectronic components manufacturing. The mold constraint is also taken into consideration. The authors proposed a random-key genetic algorithm to solve the problem.

In (Zhao et al., 2018), the authors investigated the two parallel machines with mold constraints. The authors proposed a mathematical system and a heuristic to solve the studied problem. In (Zhang et al., 2018), the authors proposed a mixed differential evolution genetic algorithm (DEGA) to solve the problem with mold constraint. The objective was the minimization of the makespan.

Many studies were developed to solve parallel machine scheduling problems. The problem is NP-hard in the strong sense. Several studies in the specialized literature proposed meta-heuristic algorithms. The mixed-integer programming model can be used to solve the problem of the parallel machine. Li et al. (2011) developed a multi-objective mathematical model for the problem of scheduling plastic injection machines under mold constraints. Keskinturk et al. (2012) developed a bee algorithm for mold project scheduling and the proposed method proved its efficiency.

Chung et al. (2019) treated the problem of two identical parallel machines with mold constraints. Three heuristics are proposed to solve the problem. A worst case of $\frac{3}{2}$ is proven in the latter paper.

Kowalczyk and Leus (2017) studied the problem of conflicting jobs. A graph of jobs is given in order to define the conflicting jobs. More precisely, conflicting jobs mean that these jobs can't be processed on the same machine. An exact solution is developed based on the branch and price.

Hà et al. (2019) treated the parallel machines problem with conflict graph. The main objective of this work is to maximize the total weight of tasks which completion times do not exceed the due date.

Zinder et al. (2021) solved the problem of maximization of the total weight of on-time jobs in parallel machines with a conflict graph.

https://doi.org/10.24846/v31i1y202207

In (Haouari & Jemmali, 2008), the authors proposed several heuristics to solve the problem of parallel machines. In addition, the authors proposed two branch and bound methods to solve the problem optimally. The results showed the performance of these branch and bound algorithms.

Davidović et al. (2012) proposed a mathematical model to find the optimal quantity of molds for the rubber gloves production planning problem. Tian et al. (2013) presented a heuristic algorithm based on mathematical programming for a lot-sizing and scheduling problem in mold-injection production. Arnaout (2021) used the Worm optimization algorithm to solve the two parallel machine problems under a single server constraint and proved its efficiency through extensive computational results.

Recently, several algorithms were developed to solve different problems in scheduling (Jemmali, 2021a; Jemmali, 2021b). These algorithms can be utilized and applied to the studied problem. In addition, algorithms developed in (Jemmali, 2021c) and in (Jemmali & Alourani, 2021) can be adopted.

In addition, in a recent work (Hà et al., 2021), three objective functions in parallel machines problem with conflict jobs were discussed. These functions are minimizing the makespan, minimizing the weighted summation of the jobs' completion time, and maximizing the total weights of completed jobs. Several mixed integer linear programming models are proposed to solve the problems. As described above, the parallel machine problem is NP-hard. No exact method reaches the optimal solution in polynomial time. This paper proposes a discrepancy search-based approach to obtain a near-optimal solution to the scheduling problem under mold constraints on parallel machines.

The rest of the paper is organized as follows. Section 2 is reserved for the problem description. Section 3 presents the proposition of the new lower bound of the studied problem. The proposed algorithms that were developed to solve the presented problem are detailed in section 4. The development of the novel meta-heuristic CDDS (Climbing Depth-bounded Discrepancy Search) is presented in section 5. The experimental results are detailed in section 6. The conclusion is given in section 7.

2. Problem Description

The problem definition and some of its relevant proprieties are detailed in this section. In addition, the parallel machines scheduling problem with as well as some of its characteristics are recalled. This is because of the key role of the latter problem in providing algorithms for the current studied problem.

The two parallel machine scheduling problem with mold constraints is stated as follows. A set of two machines M_1 and M_2, identical parallel machines, is given. A set $J = \{1, 2, ..., n\}$ of n jobs has to be processed on the machines respecting the following constraint. Two jobs j_1 and j_2 in the same mold can't be processed on the two machines simultaneously. These jobs j_1 and $j_2$are conflicts. The index of mold is denoted by m_1 with $1 \le i \le h$, where h is the number of molds. The processing time of each job is denoted by p_j. Each job can be processed only on one machine. The preemption of the execution of each job is not allowed. The completion time on each machine is denoted by C_1 and C_2 on machines M_1 and M_2, respectively. The completion time of a job j is denoted by t_j. The time of the mold replacement is equal to zero in this study. The maximum completion time is calculated as follows:

$$C_{\max} = \max_{1 \le j \le n} t_j = \max(C_1, C_2) \quad (1)$$

The objective is to minimize the $C_{\max}$. The problem is denoted as $P2 \mid m_i \mid C_{\max}$.

3. Lower Bound

A new lower bound for the studied problem is proposed. This lower bound is based on the idea of the relaxation problem. The following proposition gives the lower bound.

Proposition 1

The lower bound $\tilde{L}_{TV}$ detailed in (Haouari et al., 2006) is a lower bound for the studied problem.

Proof

Since the two parallel machines problem $P2 \mid\mid C_{\max}$ is a relaxation problem $P2 \mid m_i \mid C_{\max}$, the lower bound $\tilde{L}_{TV}$ detailed in (Haouari et al., 2006) is a lower bound for the studied problem. In this paper the developed lower bound $\tilde{L}_{TV}$ is denoted by *LB*.

4. Proposed Algorithms

Several algorithms will be presented in this section. These algorithms are based on three approaches. The first approach is the application of the dispatching rules LPT (Longest Processing Time) and SPT (Smallest Processing Time) with some modifications adapted to the studied problem. The second approach is based on the critical mold. The third approach is based on the classification into groups of the jobs belonging to different molds.

4.1 Sorting with Modifications Algorithm (*SM*)

This algorithm is based on four functions. The first function is the longest processing time (LPT). The second one is the smallest processing time (SPT). The third function is based on the idea of reducing the time-out applying LPT. The time-out is a time slot in a machine during which any job should be processed. Indeed, when a job *j* is scheduled and if the completion time of this job on M_1 is the same as on M_2 then, the machine that reduces the time-out is chosen. The fourth function is based on the idea of reducing the time-out applying SPT. Now, after applying these four functions, the best solution will be picked. This is conducted to the *SM* value.

4.2 Longest-Mold Variants Algorithm (*LM*)

The summation of all the processing times of jobs that belong to a mold m_i is denoted by Cm_i. Each mold will be represented as a fictive job Fm_i with processing time Cm_i. Indeed, the number of fictive jobs is *h*. One denotes by $Fm = \{Fm_1, ..., Fm_h\}$.

LT(*Ls*) denotes the procedure that applies the LPT heuristic on a list of jobs *Ls*. The first part of the algorithm *LM* is to call LT(*Fm*). The obtained makespan is denoted by LM_1. The second part of the algorithm *LM* is based on calculating the second value LM_2 as follows.

Firstly, the longest mold which is the mold that has the maximum Cm_i has to be determined. All the jobs in a mold m_i will be grouped in a set Sm_1. An intuitive value is calculated as $V = \frac{\sum_{j=1}^{n} p_j}{2}$. The jobs in Sm_i are sorted according to the non-increasing order of their processing time. A sequence S_q is created by the arrangement of all sorted jobs in Sm_i $\forall i \in \{1, ..., h\}$. Now, the jobs belonging to *Sq* on M_1 are started to be scheduled until reaching *V*. The remaining jobs will be scheduled on M_2. The obtained makespan will be denoted by LM_2. The value returned by the algorithm *LM* will be $\min(LM_1, LM_2)$.

The instructions given by *LM* heuristic are illustrated in Algorithm 1. Hereafter, the function DFJ() calculates the load of each mold and determines the fictive jobs Sm_2. NINCRS(*Ls*) is the function that sorts a list of jobs according to the non-increasing order of their processing time. SJ(*Ls*, *X*) denotes the function that schedules a list of jobs on the first machine without exceeding *X*. The remaining jobs will be scheduled on the second machine. The makespan will be returned by SJ().

Algorithm 1. *LM* Algorithm

```
Call DFJ(J, h)
LT(Fm)
Calculate V For (i = 1 to h)
    Call NINCRS(Sm_i)
EndFor
Determine Sq
LM_2 =SJ(Sq, V)
LM = min(LM_1, LM_2)
```

4.3 Grouped-Mold Algorithm (*GM*)

n_i denotes the number of jobs in Sm_i and $T = \max_{1 \le i \le h} n_i$. J_i^k denotes the job in the mold *i* and in the k^{th} position in the set Sm_i $\forall i \in \{1, ..., h\}$ and $\forall k \in \{1, ..., T\}$. Now, the groups Gr^k $\forall k \in \{1, ..., T\}$ are constructed as follows. $Gr^1 = \{J_1^1, J_2^1, ..., J_h^1\}$, $Gr^2 = \{J_1^2, J_2^2, ..., J_h^2\}$,..., $Gr^T = \{J_1^T, J_2^T, ..., J_h^T\}$. It is worthy to note that $\forall i \in \{1, ..., h\}$, if $k > n_i$ which implies the non-existence of J_i^k.

The first part of the algorithm *GM* is to calculate the GM_1 value as follows. The jobs in Gr^1 are scheduled according to LPT. After that, the jobs in Gr^2 are scheduled according to LPT, and so on until scheduling the jobs in Gr^T. The obtained makespan will be denoted by GM_1.

The second part of the algorithm *GM* is to calculate the GM_2 value as follows. The procedure utilized to calculate GM_1 is applied on all the sets Sm_i excluding those containing only one job. The jobs in the excluded sets will be considered as a new group and will be scheduled one by one separately on the most available machine. The

obtained makespan will be denoted by GM_2. The value returned by the algorithm *GM* will be $\min(GM_1, GM_2)$.

The instructions given by *GM* heuristic are illustrated in Algorithm 2. Hereafter the function DG() determines groups $Gr^l \ \forall k \in \{1,...,T\}$.

Algorithm 2. *GM* Algorithm

```
Call DFJ(J, h)
Call NINCRS(Fm)
Call DG ()
GM_1 = LT(Gr)
For (i = 1 to T) do
    If len(Gr^i = 1) then
        Grp pushback(Gr, Gr^i)
    EndIf
EndFor
GM_2 = LT (Grp)
GM = min(GM_1, GM_2)
```

Example 1

Taking into consideration the example presented in Table 1, *GM* algorithm works as follows. Firstly, $Gr^1 = \{3,5,4,6\}$, $Gr^2 = \{7,8\}$, $Gr^3 = \{1,2\}$, $Gr^4 = \{10\}$, and $Gr^5 = \{9\}$ are obtained. The first considered sequence is $Gr = \{\{3,5,4,6\},\{7,8\},\{1,2\},\{10\},\{9\}\}$. The sequences on the two identical parallel machines are $\{3,5,4,6\}$ and $\{7,8,1,2,10,9\}$. The makespan $GM_1 = 12$.

Table 1. p_j and *h* values for ten jobs for Example 1

j	1	2	3	4	5	6	7	8	9	10
p_j	2	1	3	5	3	1	3	2	1	2
h	3	4	3	1	4	2	3	4	3	3

The second part of algorithm works as follows. $Gr^1 = \{3,5\}$, $Gr^2 = \{7,8\}$, $Gr^3 = \{1,2\}$, $Gr^4 = \{10\}$, $Gr^5 = \{9\}$, $Gr^6 = \{4\}$, and $Gr^7 = \{6\}$ are obtained. The considered sequence is $\{\{3,5\},\{7,8\},\{1,2\},\{10\},\{9\},\{4\},\{6\}\}$. The sequences on the two parallel machines are $\{3,5,10,9,6\}$ and $\{7,8,1,2,4\}$. The makespan $GM_2 = 13$. Therefore $GM = \min(GM_1, GM_2)$.

4.4 Grouped-Variant Mold Algorithm (*GV*)

Firstly, the groups that have only one job are conserved. These jobs will constitute a set $J1$. The jobs in the remaining groups will be scheduled according to LPT. Now, the jobs in $J1$ will be scheduled one by one. The obtained makespan will be denoted by GV_1. The second step is to apply the above procedure but instead of applying LPT in the remaining groups SPT is applied. The obtained makespan will be denoted by GV_2. The value returned by the algorithm *GV* will be $\min(GV_1, GV_2)$.

4.5 Improved Longest Mold Algorithm (*IL*)

This algorithm applies the same procedures detailed in *LM* algorithm. The difference is the value of *V*. Indeed, in this algorithm $V = LB$. There is no dominance between *LM* and *IL*.

4.6 Improved Algorithms

All the algorithms presented above will be improved using an enhancing procedure denoted by CDDS which will be detailed in the next $\tilde{SM}$ section. Applying the CDDS on SM, LM, GM, GV, and IL the new algorithms will be denoted by $\widetilde{SM}$, $\widetilde{LM}$, $\widetilde{GM}$, $\widetilde{GV}$, and $\widetilde{IL}$, respectively.

4.7 Best Algorithm

This algorithm is obtained after calling all the algorithms described in subsection 4.6, and the best value will be returned. This algorithm is denoted by BA. Thus, $BA = \min(\widetilde{SM}, \widetilde{LM}, \widetilde{GM}, \widetilde{GV}, \widetilde{IL})$,

5. Enhanced Procedure CDDS

In this section, an existing local search method based on tree-search, called Climbing Depth-bounded Discrepancy Search (CDDS) is used (Hmida et al., 2011). This method is a hybrid one based on two discrepancy-based methods: DDS (Hmida et al., 2010) and Climbing Discrepancy Search (CDS) (Hmida et al., 2007). CDDS introduced an intensification process around promising solutions. This discrepancy search-based starts with an initial solution which will be explained below. The exploration of neighborhoods through several iterations makes the new values to be more and more far from the solution fixed as the referenced one. After each iteration, the best solution will be picked. If the picked solution is better than the referenced solution, then the latter one will be updated, and the instruction is restarted. The advantage of CDDS is that the neighborhoods are defined and structured by the discrepancy principle. Thus,

variable-size neighborhoods are built using a gradual increase of the allowed discrepancies. The utilization of a discrepancy-based procedure conducts to the organization of the local search approach. The discrepancy principle is reinforced by using a depth limit to restrict the exploration in less-promised areas.

Thus, applying a discrepancy consists in scheduling another job than that given by the heuristic. For example, when the depth is equal to 4 and the sequence returned by a heuristic is {6,8,4,3,2,1,5,10,9,7} for $n=10$. The generated neighborhood is based on the permutation of the 4 jobs in the given sequence. For this example, the generated neighborhood is {**3**,6,8,4,2,1,5,10,9,7}, {6,**3**,8,4,2,1,5,10,9,7}, {6,8,**3**,4,2,1,5,10,9,7}, {**4**,6,8,3,2,1,5,10,9,7}, {6,**4**,8,3,2,1,5,10,9,7} and {**8**,6,4,3,2,1,5,10,9,7}.

The discrepancy number is calculated as follows. The first value obtained by the heuristic represents zero discrepancies. All the other values represent 1-discrepancy.

The instructions given by CDDS algorithm are illustrated in Algorithm 3. Hereafter, $ISP()$ is the procedure which applies one heuristic among SM, LM, GM, GV, and IL returns the first reference solution denoted by S_0. The $CLP()$ is the procedure which applies discrepancy, computes new leaves in the tree denoted by , and returns the $C_{\max}$. d denotes the number of discrepancies $d=\{1,...,D_{\max}\}$ and $D_{\max}=\{1,...,n\}$ is the maximum allowed discrepancy which is a parameter of the algorithm.

Algorithm 3. The CDDS algorithm

$d=1$
$S_0=ISP(J)$
While ($d<D_{\max}$) do
  $S'=CLP(S_0,d)$
  If ($C_{\max}(S')<C_{\max}(S_0)$)
    $S_0=S'$
    $d=0$
  EndIf
  $d=d+1$
EndWhile

The proposed approach is based on discrepancy search algorithms combined with several algorithms implemented to find appropriate solutions. It is important to start with a possible solution and then update it during the evolution process. The encoding scheme adopted in this paper is based on the five heuristics presented above.

Example 2

Taking into consideration the example presented in Table 2, SM algorithm works as follows. The sequence {7,10,2,5,4,3,8,9,1,6} is obtained. The sequences on the two identical parallel machines are {7,2,4,3,9} on M_1 and {10,5,8,1,6} on M_2. The makespan obtained by the SM algorithm is 24.

Table 2. p_j and h values for ten jobs for Example 2

j	j	1	2	3	4	5	6	7	8	9	10
p_j	p_j	1	7	2	5	7	1	8	2	2	8
h	h	2	4	1	1	4	2	2	4	3	3

In general, the mold constraint implies that if a job j is processed by a machine, in the period of the processing time of j, no other job that requires the same mold can be processed on the other machine. This time slot can invoke a time-out.

Figure 1 represents the schedule of jobs described in Table 2. This figure shows that job 2 on machine M_1 and job 5 on machine M_2 require the same mold, then job 5 will be postponed until job 2 is finished and job 8 will also postponed until job 5 is finished. The results (S_0) are also shown in Figure 1. The initial $C_{\max}=24$.

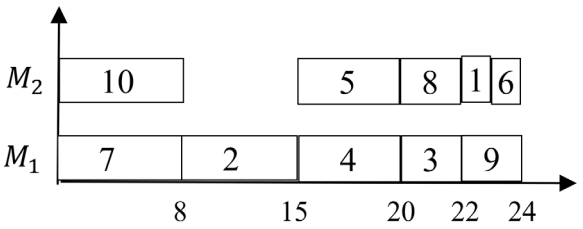

Figure 1. The initial solution

By applying CDDS algorithm, and choosing $D_{\max}=3$, the neighborhood set of the first reference solution at $d=1$, is:

{{2,7,10,5,4,3,8,9,1,6};7,2,10,5,4,3,8,9,1,6}; {7,10,2,5,4,3,8,9,1,6}; {10,7,2,5,4,3,8,9,1,6}}. Based on the first neighbor, the obtained sequences on the two identical parallel machines are {2,10,4,1,6} and {7,5,3,8,9}.

The makespan obtained by the SM algorithm is 22. The solution is presented in Figure 2. The latter solution will be considered as the new reference solution and the process will be repeated to improve the final completion time.

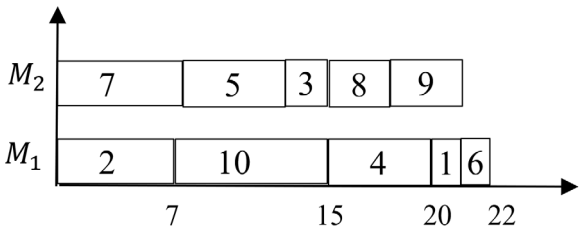

Figure 2. The second reference solution

6. Experiments and Discussion

This section reports about the experiments conducted to demonstrate the performance of the proposed algorithms for scheduling on parallel machines with the mold constraints. The experiments were performed by Visual Studio C++ on an Intel® Core™i5-1035G1 CPU@1.00GHz 1.19 GHz / 8.00 GB.

Three classes of instances were generated to assess the proposed algorithms. The variables (n,h) were fixed as follows. $h=\{3,4,5,6,7\}$ and $n=\{10,20,30,50,100,150,200\}$. The processing time p_j was generated randomly from the uniform distributions as follows: Class 1 is $U(5,20)$, Class 2 is $U(10,50)$ and Class 3 is $U(10,100)$. For fixed values of n, h, and class, 10 instances were generated. Indeed, the total instances are $5\times7\times3\times10=1050$.

Several indicators are calculated to measure the performance of the proposed algorithms:

- A: Value returned by the presented algorithm;
- $Gp=\frac{A-LB}{LB}$: Gap between the presented algorithm and the proposed lower bound;
- G_a: Average of Gp values;
- Pr: Percentage of the instances that $A = LB$;
- $Time$: Average running time in seconds.

Table 3 presents the performance of all the proposed algorithms according to Pr, $Time$, and G_a. The best algorithm is BA with a percentage of 96.4%, zero average gap, and an average running time of 146.316 s. The percentage of 96.4% proves that the proposed algorithms reach the optimal solution in 96.4% of instances.

The second best algorithm is $\widetilde{GM}$ with a percentage of 90.2%, an average gap of 0.001, and an average running time of 29.501 s. The optimal solution is reached in 90.2% of instances. The results demonstrate that the proposed algorithm is effective and efficient in solving parallel machines with mold constraints.

Table 4 presents the performance of all the proposed algorithms according to G_a when n is varying. This table shows that the average gap has a higher value when $n=10$ for all algorithms. For $n\geq100$ the average gap is equal to zero for all the algorithms. For both algorithms $\widetilde{GM}$ and BA the average gap is equal to zero excepting when $n=10$.

Table 4. Performance of all the proposed algorithms according to G_a when n is varying

n	$\widetilde{SM}$	$\widetilde{LM}$	$\widetilde{GM}$	$\widetilde{GV}$	$\widetilde{IL}$	BA
10	0.018	0.034	0.007	0.030	0.035	0.002
20	0.005	0.010	0.000	0.006	0.009	0.000
30	0.001	0.004	0.000	0.002	0.003	0.000
50	0.002	0.002	0.000	0.001	0.002	0.000
100	0.000	0.000	0.000	0.000	0.000	0.000
150	0.000	0.000	0.000	0.000	0.000	0.000
200	0.000	0.000	0.000	0.000	0.000	0.000

Table 3. Performance of all the proposed algorithms according to Pr, $Time$ and G_a

	$\widetilde{SM}$	$\widetilde{LM}$	$\widetilde{GM}$	$\widetilde{GV}$	$\widetilde{IL}$	BA
P_r	87.2%	65.1%	90.2%	73.0%	68.5%	96.4%
$Time$	6.719	66.831	29.501	33.525	9.739	146.316
G_a	0.004	0.007	0.001	0.006	0.007	0.000

Table 5 presents the performance of all the proposed algorithms according to G_a when h is varying. This table shows that the average gap decreases when h increases for all the algorithms expecting $\widetilde{GM}$. The maximum average gap of 0.016 is obtained by $\widetilde{LM}$ when $h=3$. The zero average gap is reached for $\widetilde{GM}$ when $h=3$ and for BA for all h values.

Table 5. Performance of all the proposed algorithms according to G_a when h is varying

h	$\widetilde{SM}$	$\widetilde{LM}$	$\widetilde{GM}$	$\widetilde{GV}$	$\widetilde{IL}$	BA
3	0.012	0.016	0.000	0.011	0.013	0.000
4	0.003	0.005	0.001	0.005	0.006	0.000
5	0.001	0.005	0.001	0.004	0.005	0.000
6	0.001	0.004	0.001	0.004	0.006	0.000
7	0.001	0.004	0.002	0.002	0.004	0.000

This table shows that the maximum average gap of 0.011 is obtained when $Id=5$ (the pair of (n,h) is (10,7)).

Table 6. Example of *Id* for n={10,20}

Id	n	h	$\widetilde{GM}$
1		3	0.003
2		4	0.006
3	10	5	0.007
4		6	0.008
5		7	0.011
6		3	0.000
7		4	0.000
8	20	5	0.000
9		6	0.001
10		7	0.000

Figure 3 illustrates the variation of the average gap according to Id for the $\widetilde{GM}$ algorithm The Id value is the number of the pair (n,h).

Table 6 shows the determination of the Id value for $n=\{10,20\}$.

7. Conclusion

This paper treats the problem of two machines under mold constraints. The problem is proved to be a NP-hard one. A new lower bound of the studied problem was developed. Several heuristics were proposed. A novel meta-heuristic is proposed to enhance the proposed heuristics. The experimental results showed the importance of the enhancement of the meta-heuristics. Compared with the values obtained by the lower bound of the studied problem, the best-proposed heuristic reaches the optimal solution in 96.4% of instances. It can be easily seen that the proposed algorithm obtained a near-optimal solution with a 96.4% success rate and, therefore, proved to be very effective. A comparison between different heuristics is studied and discussed in the experimental results. The proposed heuristics can be enhanced to solve a large scale of instances. In addition, the proposed heuristics can be used in a branch and bound method for the elaboration of the exact solution of the problem.

Acknowledgments

This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. (UJ-21-DR-154). The authors, therefore, acknowledge with thanks the technical and financial support from the University of Jeddah.

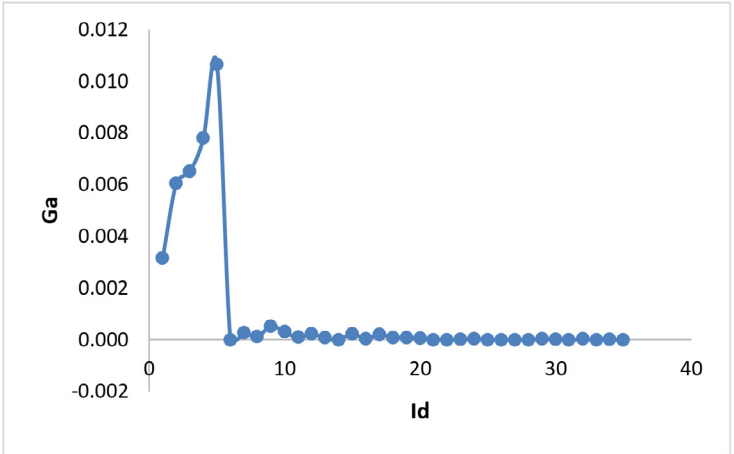

Figure 3. The variation of the average gap according to *Id* for the $\widetilde{GM}$ algorithm

REFERENCES

Arnaout, J. P. (2021). Worm optimisation algorithm to minimise the makespan for the two-machine scheduling problem with a single server, *International Journal of Operational Research, 41*(2), 270-281.

Chung, T., Gupta, J. N. D., Zhao, H. & Werner, F. (2019). Minimizing the makespan on two identical parallel machines with mold constraints, *Computers & Operations Research*, *105*, 141-155.

Davidović, T., Šelmić, M., Teodorović, D. & Ramljak, D. (2012). Bee colony optimization for scheduling independent tasks to identical processors, *Journal of Heuristics*, *18*(4), 549-569.

Hà, M. H., Ta, D. Q. & Nguyen, T. T. (2021). Exact Algorithms for Scheduling Problems on Parallel Identical Machines with Conflict Jobs, *arXiv preprint arXiv:2102.06043*.

Hà, M. H., Vu, D. M., Zinder, Y. & Thanh, T. (2019). On the capacitated scheduling problem with conflict jobs. In *2019 11th International Conference on Knowledge and Systems Engineering (KSE),* (pp. 1-5). DOI: 10.1109/KSE.2019.8919323

Haouari, M., Gharbi, A. & Jemmali, M. (2006). Tight bounds for the identical parallel machine scheduling problem, *International Transactions in Operational Research*, *13*(6), 529-548.

Haouari, M. & Jemmali, M. (2008). Tight bounds for the identical parallel machine-scheduling problem: Part II, *International Transactions in Operational Research*, *15*(1), 19-34.

Hmida, A. B., Haouari, M., Huguet, M. J. & Lopez, P. (2010). Discrepancy search for the flexible job shop scheduling problem, *Computers & Operations Research*, *37*(12), 2192-2201.

Hmida, A. B., Haouari, M., Huguet, M. J. & Lopez, P. (2011). Solving two-stage hybrid flow shop using climbing depth-bounded discrepancy search, *Computers & Industrial Engineering*, *60*(2), 320-327.

Hmida, A. B., Huguet, M. J., Lopez, P. & Haouari, M. (2007). Climbing depth-bounded discrepancy search for solving flexible job shop scheduling problems. In *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2007),* (pp. 217-224).

Hong, T.-P., Sun, P.-C. & Jou, S.-S. (2009). Evolutionary computation for minimizing makespan on identical machines with mold constraints, *WSEAS Transactions on Systems and Control*, *7*(4), 339-348.

Hong, T.-P., Sun, P.-C. & Li, S.-D. (2008). A heuristic algorithm for the scheduling problem of parallel machines with mold constraints. In the *7th WSEAS International Conference on Applied Computer & Applied Computational Science, 7*(6), (pp. 642–651).

Jemmali, M. (2021a). Intelligent algorithms and complex system for a smart parking for vaccine delivery center of COVID-19, *Complex & Intelligent Systems*, 1-13.

Jemmali, M. (2021b). An optimal solution for the budgets assignment problem, *RAIRO: Recherche Opérationnelle*, *55*, 873.

Jemmali, M. (2021c). Projects Distribution Algorithms for Regional Development, *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, *10*(3), 293-305.

Jemmali, M. & Alourani, A. (2021). Mathematical model bounds for maximizing the minimum completion time problem, *Journal of Applied Mathematics and Computational Mechanics, 20*(4), 43-50.

Keskinturk, T., Yildirim, M. B. & Mehmet Barut, M. (2012). An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times, *Computers & Operations Research*, *39*(6), 1225-1235.

Kowalczyk, D. & Roel Leus, R. (2017). An exact algorithm for parallel machine scheduling with conflicts, *Journal of Scheduling*, *20*(4), 355-372.

Li, K., Shi, Y., Yang, S. & Cheng, B. (2011). Parallel machine scheduling problem to minimize the makespan with resource dependent processing times, *Applied Soft Computing*, *11*(8), 5551-5557.

Ríos-Solís, Y. Á., Ibarra-Rojas, O. J., Cabo, M. & Possani, E. (2020). A heuristic based on mathematical programming for a lot-sizing and scheduling problem in mold-injection production, *European Journal of Operational Research*, *284*(3), 861-873.

Tian, Y., Liu, D., Yuan, D. & Wang, K. (2013). A discrete PSO for two-stage assembly scheduling problem, *The International Journal of Advanced Manufacturing Technology*, *66*(1-4), 481-499.

Zhang, Z., You, F. & Zhao, X. (2018). A Research on Lot Streaming in Parallel Machines Scheduling with Mold Constraint, *Industrial Engineering Journal*, *21*(3): 59.

Zhao, H. D, Gao, J. & Zhu, F. (2018). Scheduling on parallel machines with mold constraints. In *2nd International Conference on Advanced Technologies in Manufacturing and Materials Engineering (ATMME 2018), 389* (pp. 30-37).

Zinder, Y., Berlińska, J. & Peter, C. (2021). Maximising the total weight of on-time jobs on parallel machines subject to a conflict graph, *Mathematical Optimization Theory and Operations Research*, 280-295.