Scheduling on Parallel Machines with Mold Constraints

H D Zhao[1, *], J Gao[1] and F Zhu[1]

[1]China State Shipbuilding Corporation System Engineering Research Institute, Beijing, China

*Email: haidan.zhao@outlook.com

Abstract. This paper deals with makespan minimization scheduling problem on two identical parallel machines with mold constraint. The mold constraint in this problem is described as a kind of resource-constrained. In the production of wafer fabrication, two jobs with same mold cannot be processed on two parallel machines at the same time. This problem is quite common in practice, but there are few literatures relevant to this problem. Since the problem is NP-hard, a mathematical model is discussed to describe and solve the problem and a heuristic is proposed. Computational results show that the proposed heuristic can efficiently obtain good solutions for medium and large size problems.

1. Introduction

This problem arises from a real application in the production of wafer fabrication in semiconductor plant. There are silicon wafers starting out blank and pure. In wafer fabrication, the production process consists of five stages: cleaning of the wafer's surface, applying of the photoresist, exposing of the wafer, striping the resist and etching the pattern. The exposing wafer is one of the most important stages because this step is to fulfill customer needs and the final wafer quality is decided in this step. At the stage of exposure, a kind of patterning tool known as mask usually severs as a mold. They are dismountable and contain patterns which can be transferred to an entire wafer in just a single exposure. According to the different customer orders, each wafer has a specific mold. In the wafer exposing process, the machine with molds is very expensive and therefore the number of machines is fewer than the number of molds in the semiconductor plant. Thus, molds are not fixed on the machines and should be changed all the time. To increase the efficiency of this stage, the scheduling problem of molds and jobs should be considered.

In this paper, we consider a scheduling problem of a set $N=\{1,2,...,n\}$ where there are n jobs on two identical parallel machines named M_1 and M_2. There are q mold types in the problem. Each job with a job processing time p_i where $i=1,2,...,n,$ and corresponding mold m_i where $i=1,2,...,n,$ should be processed on one of two identical parallel machines. Let S_k ($k=1,2,...,q$) be the set where jobs belong to the same mold type and V_k be the total processing time of S_k. It is noted that $V_1 \le V_2 \le ... \le V_q$ in this paper. Each machine can install at most two molds and remove them when necessary. Any mold on the machine cannot be replaced when job is processing and two jobs with the same mold cannot be processed at the same time as well. The mold replacement time is not considered in this paper. Each machine can process at most one job at a time. The job preemption, division, or cancellations are not allowed. The objective is to minimize the makespan.

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

Following the three-field notation, the problem can be denoted as $P_2 \mid m_i \mid C_{\max}$, where P_2 designates the two identical parallel machines, m_i represents the mold constraint for job i, and $C_{\max}$ denotes the maximum completion time (or makespan). The $P_2 \mid m_i \mid C_{\max}$ problem is NP-hard since the problem without mold constraint is known to be NP-hard and proved by Sethi[1] and Garey and Johnson[2].

In what follows, we review the literature related to the $P_m \| C_{\max}$ problem. There are several exact algorithms which can obtain the optimal solutions. Dell'Amico and Martello [3] presented a branch and bound algorithm based on sophisticated lower and upper bound computations and the dominance properties. An exact cutting plane algorithm based on the identification of valid inequalities was proposed by Mokotoff[4]. Dell'Amico *et al.*[5] presented an exact algorithm, which is based on a specialized binary search and a branch-and-price scheme. However, the exact algorithm can only be used for the small or medium scale problems. Many heuristics and meta-heuristics have developed for the $P_m \| C_{\max}$ problem. Graham[6] conducted the worst case analysis of the Longest Processing Time first (LPT) rule. In addition, they also provided a worst case analysis of an arbitrary list schedule for $P_m \| C_{\max}$. A more sophisticated heuristic for $P_m \| C_{\max}$, called MULTIFIT, was presented by Coffman et al[7]. The MULTIFIT algorithm based on techniques from bin-packing has a bound of 1.22. Then Friesen[8] improved the bound of this algorithm upon to 1.2. In addition, Lee and Massey[9] analyzed a heuristic combined by the LPT rule and the MULTIFIT algorithms. They used the result of LPT rule as the incumbent and then applied MULTIFIT algorithms with less iteration. Gupta and Ruiz-Torres[10] presented a new heuristic based on bin-packing and list scheduling, called LISTFIT.

Besides, metaheuristics have been applied in the $P_m \| C_{\max}$ problem. Liu and Cheng[11] proposed a kind of genetic algorithm (GA) to minimize makespan for an identical machine scheduling problem. They provided several different scale numerical examples to demonstrate that the genetic algorithm is efficient. Lee *et al.*[12] proposed a Simulated Annealing (SA) approach with hill-climbing moves to tackle the makespan problem on identical parallel machines. Davidović *et al.*[13] studied the static scheduling of independent tasks on homogeneous multiprocessor systems and proposed a Bee Colony Optimization (BCO) algorithm. The BCO algorithm belongs to the class of stochastic swarm optimization methods inspired by the foraging habits of bees in nature. Tian *et al.*[14] presented a Discrete Particle Swarm Optimization (DPSO) algorithm to solve the two stage assembly scheduling problem where the first stage is a workstation of several identical parallel machines and the second stage is a single assembly machine workshop. The results show that DPSO is an effective and efficient for assembly scheduling problem. The DPSO algorithm is also extended to solve our proposed problem due to its good ability to search solution.

We also have reviewed some related references. According to the description of limited resource in the relevant literature, the mold constraints in this problem can be described to a kind of resource constraints. In practice, the resource constraints can be defined as the constraints for jobs handling and processing, such as automated guided vehicles, machine operators, tools, pallets, fixtures, industrial robots and so on. In this paper, the mold used in the wafer process can be as a tool which precludes the job processed with freedom. Hence, we can take this problem as a kind of Resource Constrainted Parallel Machine Scheduling (RCPMS) problem. The RCPMS problem was first studied by Garey and Graham[15] who show that greedy list schedulers can obtain an approximation ratio of $3-m/3$. Garey and Johnson[16] examined the computational complexity of scheduling problems associated with a model. They have proved the RCPMS problem as a NP-hard problem. Blazewicz *et al.*[17] considered the same problem with nonpreemptable jobs and an $O(n^3)$ algorithm was given. Blazewicz *et al.*[18] and Reklaitis[19] gave a comprehensive review of the resource constrainted problem, including applications in the chemical processing industry, semiconductor manufacturing and so on. Daniels *et al.*[20] provided mathematical formulations for static and dynamic versions of the same problem. The characteristic of the problem is the dependence of the processing time on the additional resource allocated. Daniels and Hua[21] extend the formulation of the RCPMS problem for the case where the job assignment to the machines is unspecified. They referred to this problem as the RCPMS problem and stated it as a NP-hard problem. More recently, Emrah *et al.*[22] reviewed the literature related RCPMS problem and presented integer programming models for two problems. Niemeier and Wiese[23] studied the identical parallel machine problem with an orthogonal resource constraint and present an algorithm with an approximation ratio of $2+\in$.

This paper is organized as follows. In next section, a mathematical model and a lower bound are presented for the $P_2\,|\,m_i\,|\,C_{\max}$ problem. In Section 3, a heuristic is developed. The computational results are summarized in Section 4. Finally, we conclude the paper with a summary discussion on the further research directions in Section 5.

2. A mathematical model for $P_2\,|\,m_i\,|\,C_{\max}$

The proposed problem is to schedule jobs with the mold constraints on two identical parallel machines to achieve the aim of minimizing the makespan. We now discuss a mathematical model to describe and solve the $P_2\,|\,m_i\,|\,C_{\max}$ problem. We assume here that the mold types are described as different positive integers a_i, i.e. if the mold types of two jobs are the same, the difference between them is zero. The decision variables are defined as follows:

$$y_h = \begin{cases} 1, & \text{if job } h \text{ is the first job on one of the machines,} \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{ih} = \begin{cases} 1, & \text{if job } i \text{ is scheduled directly before job } h, \\ 0, & \text{otherwise,} \end{cases}$$

$$x_{in+1} = \begin{cases} 1, & \text{if job } i \text{ is the last job on one of the machines,} \\ 0, & \text{otherwise,} \end{cases}$$

Let R is a sufficiently large number and the $P_2\,|\,m_i\,|\,C_{\max}$ problem can be formulated as the follows:

$$\text{Min} \quad C_{\max} \tag{1}$$

s.t.

$$\sum_{h=1}^{n} y_h \le 2 \tag{2}$$

$$y_h + \sum_{i=1, i\ne h}^{n} X_{ih} = 1 \qquad \forall h = 1,\dots,n \tag{3}$$

$$\sum_{i=1, i\ne h}^{n+1} X_{hi} = 1 \qquad \forall h = 1,\dots,n \tag{4}$$

$$C_h \ge p_h y_h \qquad \forall h = 1,\dots,n \tag{5}$$

$$r_{ih} \ge a_i - a_h \qquad \forall i,h = 1,\dots,n \tag{6}$$

$$r_{ih} \ge a_h - a_i \qquad \forall i,h = 1,\dots,n \tag{7}$$

$$C_h \ge C_i + p_h - Rr_{ih} \qquad \forall i,h = 1,\dots,n, i\ne h \tag{8}$$

$$C_h \ge C_i + p_h - R(1-x_{ih}) \qquad \forall i,h = 1,\dots,n, i\ne h \tag{9}$$

$$\sum_{i=1}^{n} X_{in+1} \le 2 \qquad \forall i = 1,\dots,n \tag{10}$$

$$x_{ih} + x_{hi} \le 1 \qquad \forall i, h = 1, \ldots, n, i \ne h \qquad (11)$$

$$y_i + \sum_{h=1, i \ne h}^{n+1} X_{ih} \le 2 \qquad \forall i = 1, \ldots, n \qquad (12)$$

$$C_{\max} \ge C_h \qquad \forall h = 1, \ldots, n \qquad (13)$$

Equation (1) represents the objective function to minimize makespan. Constraint (2) ensures that at most two jobs may be the first job on the two machines. In constraint (3), each job must either start on one of the machines or be preceded by some other jobs. Similarly, constraint (4) presents that each job must either be succeeded by another job or be the last job on one of the machines. Constraint (5) guarantees that the completion time for the first job of each machine must be equal to or greater than its processing time. Constraints (6-8) ensure that two jobs with the same mold type cannot be processed at the same time. In constraint (9), the completion time is defined as the completion time of the predecessor job and its processing time. Constraint (10) states that at most two jobs can be the last jobs on the two machines. Constraint (11) presents that sequence of jobs is fixed by decision variables. Constraint (12) ensures that a job start on one of the machines should also be preceded by some other jobs. Constraint (13) defines $C_{\max}$ as the makespan.

2.1. *Lower bound*

A lower bound is introduced and evaluated with the following heuristics for the $P_2 \mid m_i \mid C_{\max}$ problem. Mokotoff [4] provided a mixed integer programming (MIP) formulation for the $P_m \| C_{\max}$ problem which is slack for $P_2 \mid m_i \mid C_{\max}$ problem. Thus, the optimal solution of the $P_m \| C_{\max}$ problem is used as a lower bound, named LB, for the proposed problem. The MIP formulation for the $P_m \| C_{\max}$ problem is as follows:

Min $C_{\max}$

s.t.

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad 1 \le j \le n \qquad (14)$$

$$C_{\max} - \sum_{j=1}^{n} p_j x_{ij} \ge 0 \qquad 1 \le i \le m \qquad (15)$$

$$x_{ij} \in \{0,1\} \qquad 1 \le i \le m, 1 \le j \le n \qquad (16)$$

where,

$$x = \begin{cases} 1 & \text{if job } j \text{ is assigned to machine } i \\ 0 & \text{otherwise} \end{cases}$$

Example 2. Consider a $P_2 \| C_{\max}$ problem with job processing time given in Table 1. Applying the MIP formulation in a C++ application with Cplex and the optimal solution is 22.

Table 1: The processing time and mold type of Example 1

Job	1	2	3	4	5	6	7	8	9	10
m_i	B	D	A	A	D	B	B	D	C	C
a_i	2	4	1	1	4	2	2	4	3	3
p_i	1	7	2	5	7	1	8	2	2	8

3. The proposed heuristic

In this section, two conditions are provided and one of them is a special case which can be solved easily. For the other conditions, a heuristic based on the Longest Processing Time (LPT) rule are proposed to solve the $P_2 \mid m_i \mid C_{\max}$ problem.

The first condition is a special case presented as follows.

Lemma 1. If $V_1 + V_2 + \cdots + V_{q-1} \le V_q$, there exists an optimal schedule for the $P_2 \mid m_i \mid C_{\max}$.

Proof. As we have emphasized, two jobs with the same mold cannot be processed on the machines at the same time. Thus, V_q is the maximum completion time, shown in Figure 1.

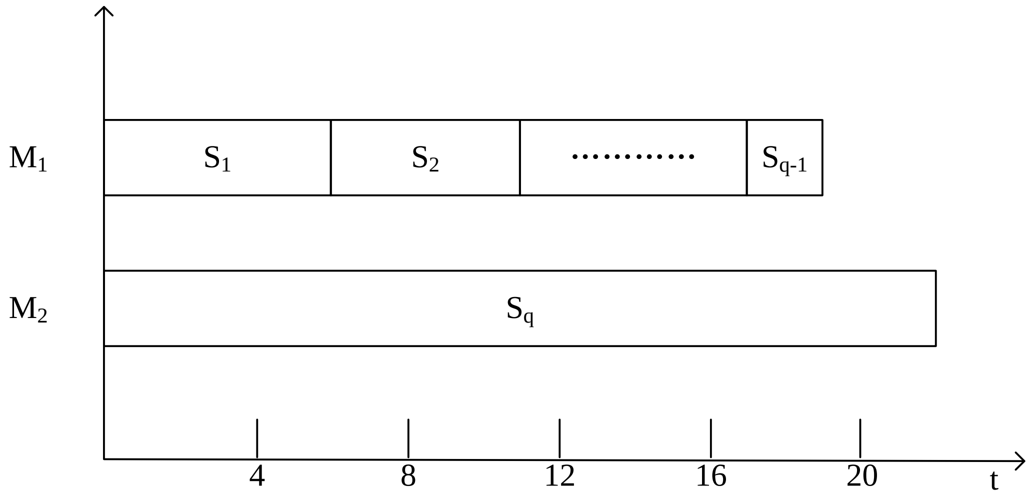

Figure. 1. Gantt chart for Lemma 1.

If $V_1 + V_2 + \cdots + V_{q-1} \ge V_q$, two proposed heuristics are proposed in second condition. The basic idea of two heuristics is extended by LPT rule as a list scheduling method where n jobs are sorted in descending order of processing time. Whenever a machine is available, an unscheduled job with largest processing time will be assigned to this machine. The heuristic is presented as follows.

3.1. Heuristic TLPT

The proposed heuristic can be introduced as a two stage procedure. At the first stage, jobs with the same mold can be grouped as a job set S_k where $k = 1, 2, ..., q$. The job sets, except set S_q with largest total processing time, can be assigned by LPT to the machines. At the second stage, the unscheduled jobs belonging S_q can be arranged by LPT rule to the front of one machine and on the back of the other, alternatively. Let MT_i where $i = 1, 2$, be the completion time of two machines.

The steps of the Two-stage LPT (TLPT) algorithm can be given as follows.

Step 1. Let jobs with the same mold group into a job set S_k. Calculate the total processing time V_k where $k = 1, 2, ..., q$.

Step 2. Select all the job sets except S_q and assign the job sets by LPT rule to the machine with earliest start time. Obtain MT_1 and MT_2.

Step 3. A dummy can be calculated by $|MT_1 - MT_2|$. Arrange remaining jobs by LPT to the machine with shortest machine completion time until the total processing time of those jobs is equal or larger than dummy.

Table 2: Computational results of PD and CPU time

		Percentage Deviation						CPU time	
		(1,10)			(1,100)			(1,10)	(1,100)
q	n	TLPT	VNS	DPSO	TLPT	VNS	DPSO	TLPT	TLPT
4	20	2.24	0.69	0.69	2.03	0.53	0.53	0.07	0.07
	50	0.59	1.25	2.42	0.86	0.66	2.49	0.17	0.16
	100	0.14	4.14	5.26	0.25	4.12	5.88	0.32	0.31
	150	0.19	6.70	9.11	0.11	6.65	8.05	0.51	0.47
	Ave.	0.79	3.19	4.37	0.81	2.99	4.24	0.27	0.25
6	20	1.08	0.00	0.00	1.92	0.04	0.04	0.07	0.08
	50	0.19	0.06	0.18	0.42	0.01	0.35	0.18	0.17
	100	0.22	1.35	2.19	0.04	1.35	2.15	0.34	0.31
	150	0.10	2.68	3.90	0.06	3.09	4.31	0.51	0.42
	Ave.	0.40	1.02	1.57	0.61	1.12	1.71	0.27	0.24
8	20	0.86	0.00	0.00	0.59	0.02	0.02	0.09	0.07
	50	0.15	0.00	0.04	0.07	0.02	0.06	0.18	0.17
	100	0.07	0.52	1.06	0.08	0.41	1.03	0.29	0.29
	150	0.00	1.35	2.08	0.02	1.41	2.11	0.46	0.46
	Ave.	0.27	0.47	0.80	0.19	0.46	0.80	0.25	0.25
10	20	1.20	0.21	0.21	0.53	0.09	0.09	0.08	0.07
	50	0.07	0.00	0.00	0.08	0.02	0.04	0.18	0.16
	100	0.08	0.12	0.59	0.02	0.13	0.40	0.34	0.31
	150	0.00	0.62	1.07	0.03	0.76	1.41	0.52	0.46
	Ave.	0.34	0.24	0.47	0.17	0.25	0.48	0.28	0.25
	Agg.	0.45	1.23	1.80	0.45	1.21	1.81	0.27	0.25

Step 4. Assign the unscheduled job with largest processing time to the first position on the machine with shortest machine completion time.

Step 5. Assign the unscheduled job with largest processing time to the last position on the machine with shortest machine completion time.

Step 6. If $S_q = \varnothing$, obtain a makespan value and then stop; otherwise, repeat Steps 4 and 5.

Example 3. Consider the same problem as Example 1.

Step 1. $S_1 = \{j_3, j_4\}$, $S_2 = \{j_1, j_6, j_7\}$, $S_3 = \{j_9, j_{10}\}$ and $S_4 = \{j_2, j_5, j_8\}$. $V_1 = 7$, $V_2 = 10$, $V_3 = 10$, $V_4 = 16$.

Step 2. S_1, S_2 and S_3 are on M_1 and M_2, respectively.

Step 3. $|MT_1 - MT_2| = 7$. Assign job 5 and job 8 on M_2.

Step 4. Assign job 2 to the first position on M_1 and $S(q) = \varnothing$.

Step 5. $S_q = \varnothing$.

Step 6. $S_q = \varnothing$. The makespan value is 24 and stop

The solution of TLPT is 24 while the optimal solution obtained by MIP is 22. The ratio of ω'/ω is 1.09. The sequences on two identical parallel machines are $\{7,6,1,3,4,2\}$ and $\{5,8,10,9\}$. The Gantt chart is presented as Figure 2.

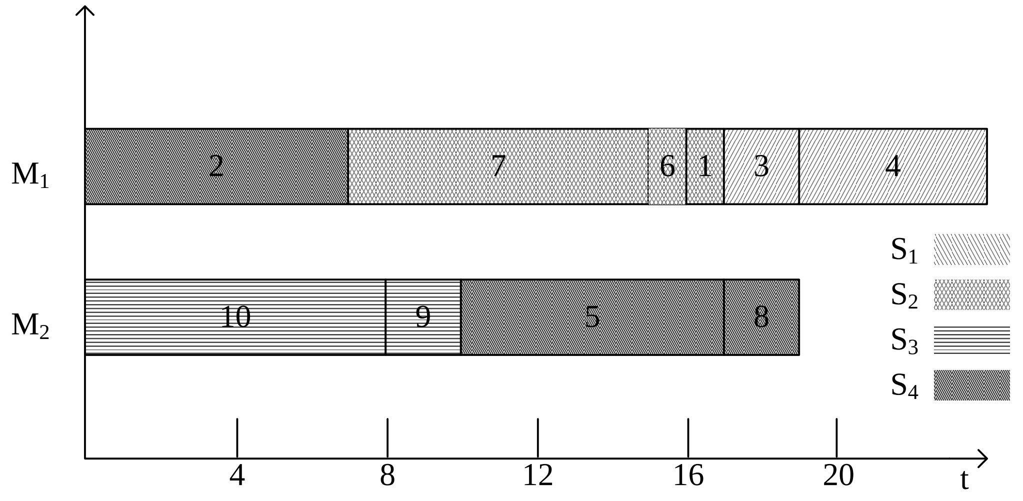

Figure 2. Gantt chart for TLPT.

4. Computational results

To verify the proposed heuristics, extensive computational experiments were conducted. The lower bound was coded in combination of C++ and CPLEX 9.0, and all other algorithms were coded in C++. The algorithms were run on the Intel Core 2 CPU (2.2 GHz) with 2.0 GB RAM.

The test problem was generated with $n = 20, 50, 100, 150$ and $q = 4, 6, 8, 10$. The processing time p_i was given randomly from uniform distributions $U(1,10)$ and $U(1,100)$ respectively. The Discrete Particle Swarm Optimal (DPSO) algorithm by Liao *et al.*[24] and Variable Neighborhood Search (VNS) algorithm by Mladenović and Hansen[25] are used as a comparison. Each of the problems was run ten times. The percentage deviation (PD) is computed as follows:

$$PD = 100 \times \frac{C_{\max}^{H} - LB}{LB} \tag{22}$$

where $C_{\max}^{H}$ is the makespan obtained by the respective heuristics and LB is the lower bound acquired by the MIP formulation. It is noted that the stop criterion of DPSO and VNS is time limited within 10 seconds.

Examining the Table 2, we can observe that TLPT have the performance with $PD \approx 0.45$. Moreover, TLPT has better performance when $q = 4$ and 6. The average PD is getting smaller as n is increasing. Whenever p_i is from uniform distributions $U(1,10)$ or $U(1,100)$, the same situation can be generalized and the heuristic can obtain the solution within 1.

In the Table 2, We can also observe that only when the job number is small, i.e., $n = 20$, VNS and DPSO perform better than the proposed heuristics. It is because the proposed heuristics based on LPT rule perform worse as the job decrease. We also can observe that VNS performs better than DPSO for all problems and the percentage deviation tends to decrease as the mold number and the job number increases. Thus, only VNS is used in the following comparison.

5. Conclusions and future research

In this paper, a two-identical parallel machine scheduling problem with the mold constraint, i.e., $P_2 \mid m_i \mid C_{\max}$, is considered. This problem is a real problem from the production of wafer fabrication. The objective function is to minimize makespan. A mathematical model and a lower bound are proposed to descript and evaluate the problem. A heuristic is provided for finding the solution. In addition, the Discrete Particle Swarm Optimal (DPSO) algorithm and Variable Neighborhood Search (VNS) algorithm are used as a comparison. Extensive experiments with different size problems show that the proposed heuristic can efficiently yield better solutions.

The mold constraint is a common practice in industry, but it is seldom discussed in the scheduling literature. Extension of the results to other machine environments will be worthwhile in the future research.

References

[1] Sethi R 1977 Mathematics of Operations Research **2** 320–30
[2] Garey M R and Johnson D S 1979 Computer and intractability: a guide to the theory of NP-completeness San Francisco W H Freeman
[3] Dell'Amico M and Martello S 1995 Operations Research Society of America Journal on Computing **7** 191–200
[4] Mokotoff E 2004 European Journal of Operational Research **152** 758–769
[5] Dell'Amico M, Iori M, Martello S and Monaci M. 2008 *INFORMS Journal on Computing* **20** 333–344
[6] Graham R L 1966 *Bell System Technical Journal* **45** 1563–1581
[7] Coffman E G, Jr, Garey M R and Johnson D S 1978 Society for Industrial and Applied Mathematics Journal on Computing **7** 1–17
[8] Friesen D K 1984 Society for Industrial and Applied Mathematics Journal on Computing **13** 170–181
[9] Lee C Y and Massey J D 1988 *Discrete Applied Mathematics* **20** 233–242
[10] Gupta J N D and Ruiz-Torres A J 2001 *Production Planning & Control* **12** 28–36
[11] Liu M and Cheng W. 1999 Artificial Intelligence in Engineering **13** 399–403
[12] Lee W C, Wu C C and Chen P 2006 The International Journal of Advanced Manufacturing Technology **31** 328–334
[13] Davidović T, Šelmić M, Teodorović D and Ramljak D 2012 *Journal of Heuristics* **18** 549–569.
[14] Tian Y, Liu D, Yuan D and Wang K 2012 The International Journal of Advanced Manufacturing Technology **66** 481–499
[15] Garey M R and Grahams R L 1975 Society for Industrial and Applied Mathematics Journal on Computing **4** 187–200
[16] Garey M R and Johnson D S 1975 Society for Industrial and Applied Mathematics Journal on Computing **4** 397–411
[17] Blazewicz J, Kubiak W, Rock H and Szwarcfiter J 1987 *Acta Informatica* **24** 513–524
[18] Blazewicz J, Dror M and Weglarz J 1991 *European Journal of Operational Research* **51** 283–300
[19] Reklaitis G V 1996 Batch Processing Systems Engineering **143** 660–705
[20] Daniels R L, Hoopes B J and Mazzola J B 1996 *Management Science* **42** 1260–1276
[21] Daniels R L and Hua S Y 1999 *Computers & Operations Research* **26** 143–155
[22] Emrah B E, Ceyda O and Irem O 2013 *European Journal of Operational Research* **230** 449–463
[23] Niemeier M and Wiese A 2013 *Approximation and Online Algorithms* **7846** 242–256
[24] Liao C J, Tjandradjaja E and Chung T P 2012 *Applied Soft Computing* **12** 1755–1764
[25] Mladenović N and Hansen P 1997 *Computers & Operations Research* **24** 1097–1100