Complexity of preemptive minsum scheduling on unrelated parallel machines

Complexity of Preemptive Minsum Scheduling on Unrelated Parallel Machines

René Sitters[*]

Abstract

We show that the problems of minimizing total completion time and of minimizing the number of late jobs on unrelated parallel machines, when preemption is allowed, are both NP-hard in the strong sense. The former result settles a long-standing open question. A special case of the unrelated machine model is the identical machine model with the restriction that a job can only be processed on a specific subset of the machines. We show that in this model the problem of minimizing total completion time, when preemption is allowed, can be solved in polynomial time by proving that the use of preemption is redundant.

1 Introduction

Suppose that m machines M_i $(i = 1, \dots, m)$ have to process n jobs J_j $(j = 1, \dots, n)$. Each job can be processed on any of the machines but it can only be worked on by one machine at a time. Each machine can process at most one job at a time. The time it takes to process job J_j completely on machine M_i is given by a positive integer p_{ij}. Preemption is allowed, i.e., we may interrupt the processing of a job, and continue it later on the same, or at any time on another machine. We are interested in finding a schedule for which a certain optimality criterion is met.

In this paper we consider two optimality criteria: minimizing the sum of completion times $\sum_{j=1}^{n} C_j$, and minimizing the sum of unit penalties $\sum_{j=1}^{n} U_j$. The completion time C_j of a job J_j is the last moment in time that it is processed. If a job J_j has a given due date d_j, that is, the moment in time by which it should be completed, then we say the job is *late* if $d_j < C_j$. The unit penalty U_j is 1 if job J_j is late, and 0 otherwise.

With the two optimality criteria we have defined two scheduling problems. In the notation introduced by Graham et al. [10] they are $R|pmtn|\sum U_j$ and $R|pmtn|\sum C_j$. The 'R' indicates that we have unrelated parallel machines, i.e.,

[*]Department of Mathematics, Technische Universiteit Eindhoven, P.O.Box 513, 5600 MB Eindhoven, The Netherlands. e-mail: {r.a.sitters@tue.nl

no relation between the mn processing times p_{ij} is presumed. The number of machines m is defined as part of the problem instance. If the number of machines m is fixed, the notation 'Rm' is used. The acronym '$pmtn$' indicates that preemption is allowed. The third field indicates which optimality criterion is used.

Lawler [15] proved that the problem $P|pmtn|\sum U_j$, i.e., the problem of minimizing the number of late jobs on identical parallel machines, is already NP-hard in the ordinary sense. We say machines are *identical* if $p_{i1} = p_{i2} = \ldots = p_{im}$ for all i, $1 \leq i \leq n$. Hence $R|pmtn|\sum U_j$ is NP-hard in the ordinary sense as well. Du and Leung [5] show that the problem $R|pmtn, r_j|\sum U_j$, where the given jobs have release dates, is strongly NP-hard.

We show that $R|pmtn|\sum U_j$ is NP-hard in the strong sense.
Little is known about the preemptive minimization of the sum of completion times. If no preemption is allowed $(R||\sum C_j)$, an optimal schedule can be found in polynomial time by solving a weighted matching problem (Horn [11], Bruno et al. [4]). In the case of uniform parallel machines, Gonzalez [9] shows that an optimal preemptive schedule can be found in polynomial time. We say that the machines are uniform if $p_{ij} = p_j/s_i$ for given processing requirement p_j of job J_j, and speed s_i of machine M_i. The problem $P2|pmtn, r_j|\sum C_j$, is NP-hard in the ordinary sense [6], and the problem $P2|chains,\ pmtn|\sum C_j$, where chainlike precedence relations between the jobs are added, is NP-hard in the strong sense [7].

We show that the problem $R|pmtn|\sum C_j$ is NP-hard in the strong sense.
For almost all scheduling problems the preemptive version of the problem is not harder to solve than the non-preemptive version. For at least two scheduling problems this emperical law does not hold true. Brucker, Kravchenko, and Sotskov [2] showed that the preemptive job shop scheduling problem with two machines and three jobs $(J2|n = 3, pmtn|C_{max})$ is NP-hard in the ordinary sense. However, Kravchenko and Sotskov [13] show that the non-preemptive version can be solved in $O(r^4)$ time, where r is the maximum number of operations of a job.

The other exception is that of finding an optimal preemptive schedule for equal length jobs on identical parallel machines. The optimality criterion is the sum of weighted lateness penalties. This problem $(P|p_j = p,\ pmtn|\sum w_jU_j)$ was proven to be NP-hard in the ordinary sense by Brucker and Kravchenko [1]. In the same paper they give a $O(n \log n)$ time algorithm for the non-preemptive version. Recently they even proved strong NP-hardness for the preemptive problem [3].

$R|pmtn|\sum C_j$ is the third problem type on this short list.
A special case of the unrelated machine model is the model in which each job j has a fixed processing time p_j but the job can only be processed on a job-specific subset of the machines. So the processing time of job j on machine i (p_{ij}) is either p_j or infinite. We show that within this model the problem of minimizing total completion time, when preemption is allowed, can be solved in polynomial time by showing that the use of preemptions can not reduce the optimal objective value.

Table 1: Complexity status of some unrelated machine problems

$R\|\|\sum C_j$	$O(n^3)$	[4],[11]	$R\|pmtn\|\sum C_j$	NP-hard*	[◇]
$R\|\|\sum U_j$	NP-hard*	[8]	$R\|pmtn\|\sum U_j$	NP-hard*	[◇]
$R\|p_{ij} \in \{p_j, \infty\}\|\sum C_j$	$O(n^3)$	[4],[11]	$R\|pmtn, p_{ij} \in \{p_j, \infty\}\|\sum C_j$	$O(n^3)$	[◇]
$R\|p_{ij} \in \{p_j, \infty\}\|\sum U_j$	NP-hard*	[8]	$R\|pmtn, p_{ij} \in \{p_j, \infty\}\|\sum U_j$	NP-hard	[15]

[◇] this paper,
* NP-hard in the strong sense.

The complexity of $R|pmtn|\sum C_j$ and $R|pmtn|\sum U_j$ is still open for a fixed number of machines, even for $m = 2$. Another open question is whether either of the problems $P|pmtn|\sum U_j$ and $R|\text{pmtn}, p_{ij} \in \{p_j, \infty\}|\sum U_j$ is solvable in pseudopolynomial time or NP-hard in the strong sense.

2 Minimizing the Number of Late Jobs

$R|pmtn|\sum U_j$

Instance: A number α, a set $\{M_1, M_2, \ldots, M_m\}$ of m unrelated machines, a set $\{J_1, J_2, \ldots, J_n\}$ of n independent jobs, a set $\{p_{ij} \mid i = 1 \ldots m, j = 1 \ldots n\}$ where p_{ij} is the processing time of job J_j on machine M_i, and a due date d_j for each job J_j.
Question: Does there exist a preemptive schedule for which $\sum_{j=1}^{n} U_j \leq \alpha$?

It is not obvious this problem is in the class NP since we have to exclude the possibility that there is a schedule with a superpolynomial number of preemptions that has a strictly smaller objective value than any schedule with a polynomial number of preemptions. Lawler and Labetoulle [14] show that for any monotone, non-decreasing objective function $f(C_1, C_2, \ldots, C_n)$, an optimal schedule can be constructed by solving a linear program, if the completion times of the jobs in an optimal schedule are given. The number of preemptions for this schedule is $O(m^2 n)$. Verifying the feasibility of a schedule with $O(m^2 n)$ preemptions requires polynomial time, and so $R|pmtn|\sum U_j$ is in NP.

We present a reduction from the 3-Dimensional Matching problem to $R|pmtn|\sum U_j$. The former was proven to be NP-complete by Karp [12].

3-Dimensional Matching (3DM)

Instance: Three sets $U = \{u_1, \ldots, u_m\}$, $V = \{v_1, \ldots, v_m\}$, and $W = \{w_1, \ldots, w_m\}$, and a subset $S \subset U \times V \times W$ of size $n \geq m$.
Question: Does S contain a perfect matching, that is, a subset S' of cardinality

m that covers every element in $U \cup V \cup W$?

As a preliminary we define, for each instance of the 3DM problem, some special sets of machines and jobs that we shall use in both NP-hardness proofs in this paper.

Basic sets of machines and jobs

Given an instance of the 3DM problem we define one machine U_i for each element u_i of the set U. The set of these m machines is denoted by U as well. In the same way we define the sets of machines V and W. We use the following notation for the set S: $S = \{(u_{\alpha_j}, v_{\beta_j}, w_{\gamma_j})|\ j = 1, \ldots, n\}$. For each element $s_j = (u_{\alpha_j}, v_{\beta_j}, w_{\gamma_j})$ of S we define one job which we denote by J_j. The set of these n jobs is denoted by J. The processing time of these jobs is small on the three machines that correspond to the related triple, and is large on all other machines. To be specific: let $s_j = (u_{\alpha_j}, v_{\beta_j}, w_{\gamma_j})$ be an element of S, then the processing time of job J_j is $3p$ on machine U_{α_j}, $\frac{3}{2}p$ on machine V_{β_j}, and p on machine W_{γ_j}, where $p \in \mathbb{R},\ p \geq 2$. The processing time is K on any of the other $3m - 3$ machines, where $K \in \mathbb{R},\ K \geq 6p$. The numbers K and p will be chosen appropriately in each of the two NP-hardness proofs.

We use the following terminology. We say that a machine is a 'slow' machine for the job J_j if the processing time of job J_j is K on that machine. In any other case we say that the machine is 'fast' for the job. The following lemma relates the perfect 3-dimensional matching property with properties of the scheduling instance given by the sets U, W, W and J.

Lemma 2.1 *Let I be an instance of 3-DM and let the corresponding sets U, V, W and J be described as above. If we add the restriction that none of the V-machines can process a job before time $t = 1$ and none of the W-machines can be used for processing before time $t = 2$, then for every preemptive schedule the following holds:*
(i) $C_j \geq p + 1$ for any job $J_j \ \in \ J$,
(ii) if there are m jobs for which $C_j < (p + 1) + \frac{1}{6}$, then I contains a perfect matching.

Proof. (i) Schedule job J_j on machine U_{α_j} from $t = 0$ to $t = 1$, on machine V_{β_j} from $t = 1$ to $t = 2$, and on machine W_{γ_j} from time $t = 2$ onwards. Scheduled in this way $C_j = p + 1$. Any other schedule will give a strictly larger completion time.
(ii) Call the schedule of job J_j as described in (i), the optimal schedule of this job. If I does not contain a perfect matching, it is impossible to schedule a set of m jobs such that each job gets its optimal schedule. At least one job of any set of m jobs must diverge from its optimal schedule for a total time of at least 0.5, i.e., during this time that job is not processed on its fastest available

machine. This will delay the best possible completion time for this job be by at least $\frac{1}{6}$. $\square$

We will use this construction with the optimal schedule of the J-jobs to prove Theorems 1 and 2. In the lemma we made the restriction that sets of machines can only be used from some point in time onwards; in the NP-hardness proofs we have to find a way to enforce this. It turns out that this is easy for the $\sum U_j$ objective and more complicated for the $\sum C_j$ objective.

Theorem 1 *$R|pmtn|\sum U_j$ is strongly NP-hard.*

Proof. The problem is in the class NP as we already showed. To complete the proof we reduce the 3-Dimensional Matching problem to $R|pmtn|\sum U_j$. Given an instance of 3-DM of size n and m, we will define a scheduling instance containing a set of $3m$ machines and $2m+n$ jobs, and prove that a 3-dimensional matching exists if and only if there exists a preemptive schedule for which the number of late jobs does not exceed a certain value.

Let I be an instance of 3-DM with the notation as defined before. The scheduling instance consists of the basic sets of machines U, V and W, and the basic set of jobs J, and additionally we define the set A of jobs as follows. For each machine of the set V we introduce one job with processing time 1 on that specific machine, and with processing time K on any of the other $3m-1$ machines. The due date is 1 for all these jobs. For each machine of W we define one job with processing time 2 on that specific machine, and with processing time K on any of the other $3m-1$ machines. The due date is 2 for all these jobs. We define the set J of jobs related to S, with their processing times as before. The due dates of these jobs are $p+1$. We choose the value $p = 3m+3$.

We claim that for any schedule, the number of late jobs is less than or equal to $n-m$, if and only if a perfect matching exists. Notice that, since the number of jobs is $n+2m$, this is the same as claiming that the number of early jobs is at least $3m$ if and only if a perfect matching exists.

If there exists a perfect matching, then it is possible to schedule the jobs such that $3m$ jobs are early: Schedule the $2m$ A-jobs such that they are early (there is only one way to do this), and give the m J-jobs that correspond to the elements in the 3-dimensional matching, their optimal schedule as described in the proof of Lemma 2.1.

Now observe that it is impossible to have more than m early J-jobs. For a J-job to be early, it must be scheduled on its fast W-machine for a time of at least $p-2$. So if there are at least $m+1$ early J-jobs, then the total processing time required on the W-machines is at least $(m+1)(p-2) = (m+1)(3m+1)$. For any m this is strictly larger than $m(3m+4)$, which is the total available processing time on the W-machines before the due date, $t = 3m+4$.

We conclude that if at least $3m$ jobs are early, then these jobs are the $2m$ A-jobs and exactly m of the J-jobs. From Lemma 2.1(ii) it follows that in this case a perfect matching exists. $\square$

3 Minimizing the Sum of Completion Times

$R|pmtn|\sum C_j$

Instance: A number α, a set $\{M_1, M_2, \dots, M_m\}$ of m unrelated machines, a set $\{J_1, J_2, \dots, J_n\}$ of n independent jobs, and a set $\{p_{ij} | i = 1 \dots m, j = 1 \dots n\}$ where p_{ij} is the processing time of job J_j on machine M_i.
Question: Does there exist a preemptive schedule for which $\sum_{j=1}^{n} C_j \leq \alpha$?

Theorem 2 *$R|pmtn|\sum C_j$ is strongly NP-hard.*

Proof. The membership of the class NP follows from the same arguments as given for the problem $R|pmtn|\sum U_j$.
To complete the proof we reduce 3-Dimensional Matching to $R|pmtn|\sum C_j$. Given an instance of the 3-DM problem we define an instance of a scheduling problem and prove that a perfect matching exists if and only if there exists a preemptive schedule for which the sum of completion times does not exceed a certain value.

Let I be an instance of 3-DM with the notation as defined before. The scheduling instance consists of the basic sets of machines U, V and W, and the basic set of jobs J. Additionally we define one machine which we denote by Z. The processing time on the Z-machine is p for any job from J. The value of p is set to $p = 2$. The value of K (which was defined as the processing time on slow machines) will be specified later. Besides the set J two more sets of jobs are defined.

The second set of jobs is the set A which contains many jobs with a small processing time. For each V-machine we define M A-jobs with processing time $\frac{1}{M}$ on that specific machine and processing time K on any other machine. The value of M will be specified later. For each W-machine we define $2M$ A-jobs with processing time $\frac{1}{M}$ on that specific machine and with processing time K on any other machine. For the Z-machine we define $3M$ A-jobs with processing time $\frac{1}{M}$ on the Z-machine and processing time K on any of the other $3m$ machines. The total number of jobs in A is $(3m + 3)M$. The A-jobs are meant to keep the V-machines busy until time 1, the W-machines busy until time 2, and the Z-machine busy until time 3.

The third and last set of jobs is B. For each U-machine we define $\frac{1}{3}(n-m)$ B-jobs with processing time $2n+4$ on that specific machine, and processing time K on any other machine. (Without loss of generality we may assume that $\frac{1}{3}(n-m)$ is integer.) For each V-machine we define $\frac{2}{3}(n-m)$ B-jobs with processing time $2n + 4$ on that specific machine and processing time K on any other machine. For each W-machine we define $(n - m)$ B-jobs with processing time $2n + 4$ on that specific machine and processing time K on any other machine. The B-jobs are introduced to ensure that, in an optimal schedule, a limited part of the J-jobs is scheduled on the U-, V-, and W-machines.
If a perfect matching exists then the jobs can be scheduled as shown in Fig. 1. All A-jobs are scheduled on their fast machines such that their sum of completion

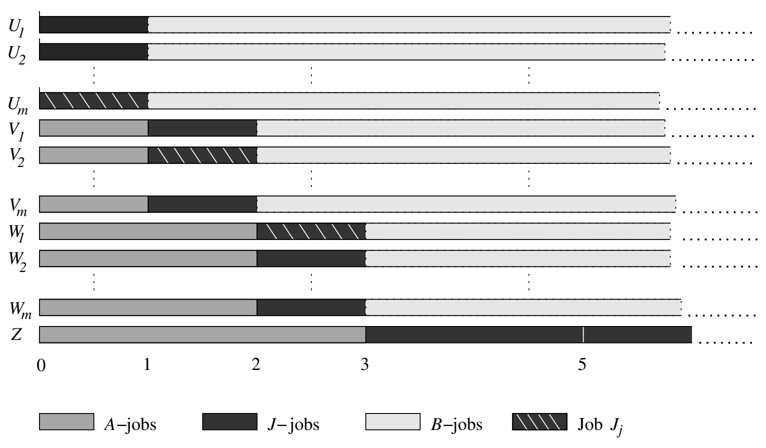

Figure 1: Sketch of the schedule σ_{3DM}.

times is minimized. There is only one way to do this. The n jobs from J that correspond to the perfect matching are scheduled as in the proof of Lemma 2.1. The completion time of these jobs is 3. All other J-jobs are scheduled after the A-jobs on the Z-machine. Each B-job is scheduled on its (unique) fast machine. The B-jobs are placed directly after the other jobs. This schedule is denoted by σ_{3DM}. The sum of completion times in σ_{3DM} is denoted by $C_{\sigma_{3DM}}$. The value of $C_{\sigma_{3DM}}$ is clearly a polynomial in m,n,M and $1/M$. The expression is omitted here.

We use the notation $C_{\tilde{\sigma}}(\tilde{J})$ for the sum of completion times of the jobs in a set $\tilde{J}$ in a schedule $\tilde{\sigma}$, and $C_{\tilde{\sigma}}$ for the sum of completion times of all jobs in schedule $\tilde{\sigma}$.

We now show that if no perfect matching exists, then for any preemptive schedule σ the total completion time is strictly larger than $C_{\sigma_{3DM}}$. In fact we will show that $C_\sigma \geq C_{\sigma_{3DM}} + \frac{1}{24}$.

We introduce two restrictions to the scheduling problem.

- *Restriction 1:* No job may be processed on a slow machine.
- *Restriction 2:* All A-jobs have to be scheduled as in σ_{3DM}.

With these two restriction we define three scheduling problems.

- *Problem 1:* The original problem with Restrictions 1 and 2.
- *Problem 2:* The original problem with Restriction 1.
- *Problem 3:* The original problem.

For each of the three problems we will show a lower bound on the total completion time in case no perfect matching exists for the 3-DM instance I. The essence of the reduction is contained in the proof for Problem l. It is intuitively clear that a lower bound for Problem 2 or 3 can be made arbitrarily close to a lower bound for Problem 1 for appropriately large values of M and K. We give explicit values for the numbers M and K and prove that these values suffice, i.e., we show that processing the A-jobs different or using slow machines will not break the reduction. Notice that it is possible to improve the schedule σ_{3DM} by preempting jobs on slow machines!

Problem 1: Let σ be an optimal schedule for Problem 1. We will prove that $C_\sigma \geq C_{\sigma_{3DM}} + \frac{1}{6}$. The fact that σ is optimal is important; we will use this implicitly.

Let T_{U_i} $(i = 1, \dots, m)$ be the time that is spent on processing J-jobs on machine U_i. Define T_{V_i}, T_{W_i} $(i = 1, \dots, m)$ and T_Z in a similar way. Notice that in schedule σ_{3DM} we have $T_{U_i} = T_{V_i} = T_{W_i} = 1$ and $T_Z = 2(n-m)$. Notice that the J-jobs are the only jobs that can be scheduled on more than one machine. Also notice that, by choosing the processing times of the B-jobs large enough $(2n+4)$, the completion time of any J-job is strictly smaller than the completion time of any B-job. (If all J-jobs are scheduled on the Z-machine, then they are completed at time $2n+3$). Now let $B(U_i)$ $(i = 1, \dots, m)$ be the set of B-jobs that have machine U_i as their fast machine. Define $B(V_i)$ and $B(W_i)$ in a similar way. Now we compare, for each machine, the total completion time of the B-jobs on that machine in schedule σ_{3DM} with those in schedule σ and obtain the following relation.

$$\begin{array}{ll} C_\sigma(B(U_i)) = C_{\sigma_{3DM}}(B(U_i)) + \frac{1}{3}(n-m)(T_{U_i} - 1) & i = 1, \dots, m, \\ C_\sigma(B(V_i)) = C_{\sigma_{3DM}}(B(V_i)) + \frac{2}{3}(n-m)(T_{V_i} - 1) & i = 1, \dots, m, \\ C_\sigma(B(W_i)) = C_{\sigma_{3DM}}(B(W_i)) + (n-m)(T_{U_i} - 1) & i = 1, \dots, m. \end{array}$$

Combining the 3 equations above yields:

$$C_\sigma(B) = C_{\sigma_{3DM}}(B) + (n-m)\left(\frac{1}{3}\sum_{i=1}^{m} T_{U_i} + \frac{2}{3}\sum_{j=i}^{m} T_{V_i} + \sum_{i=1}^{m} T_{W_i} - 2m\right).$$

We substitute the obvious relation

$$T_Z = 2n - \frac{1}{3}\sum_{i=1}^{m} T_{U_i} - \frac{2}{3}\sum_{i=1}^{m} T_{V_i} - \sum_{i=1}^{m} T_{W_i},$$

and obtain an expression for the total completion time of the B-jobs:

$$C_\sigma(B) = C_{\sigma_{3DM}}(B) + (n-m)\left(2(n-m) - T_Z\right). \tag{1}$$

Next we will deduce a lower bound on the total completion time of the J-jobs. Let $C_\sigma(J)_1$ be the sum of the m smallest completion times among the J-jobs in the schedule σ, and let $C_\sigma(J)_2$ be the sum of the $n-m$ largest completion times among the J-jobs. In a similar way we define $C_{\sigma_{3DM}}(J)_1$ and $C_{\sigma_{3DM}}(J)_2$.

Let $c_1 \leq c_2 \leq \ldots \leq c_n$ be an ordering of the completion times of the J-jobs in σ. The largest completion time c_n is at least $3+T_Z$, and the last but one is at least $3+T_Z-2$, and so on. In general we have $c_j \geq \max\{3+T_Z-2(n-j)\ ,\ 3\}$. We obtain:

$$\begin{array}{rcl} C_\sigma(J)_2 & = & \sum_{j=m+1}^{n} c_j \\ & \geq & \sum_{j=m+1}^{n} (3+T_Z-2(n-j)) \\ & = & \sum_{j=m+1}^{n} (3+2(j-m)+T_Z-2(n-m)) \\ & = & C_{\sigma_{3DM}}(J)_2 + (T_Z-2(n-m))(n-m). \end{array} \tag{2}$$

Combining equality (1) and inequality (2), and using $C_{\sigma_{3DM}}(A) = C_\sigma(A)$ and $C_{\sigma_{3DM}}(J)_1 = 3m$ we obtain

$$\begin{array}{rcl} C_\sigma & = & C_\sigma(A)+C_\sigma(B)+C_\sigma(J)_1+C_\sigma(J)_2 \\ & \geq & C_{\sigma_{3DM}}(A)+C_{\sigma_{3DM}}(B)+C_\sigma(J)_1+C_{\sigma_{3DM}}(J)_2 \\ & = & C_{\sigma_{3DM}}-C_{\sigma_{3DM}}(J)_1+C_\sigma(J)_1 \\ & = & C_{\sigma_{3DM}}-3m+C_\sigma(J)_1. \end{array}$$

From Lemma 2.1 we have $C_\sigma(J)_1 \geq 3m+\frac{1}{6}$, and therefore $C_\sigma \geq C_{\sigma_{3DM}}+\frac{1}{6}$.

Problem 2: Let σ be an optimal schedule for Problem 2. We will prove that $C_\sigma \geq C_{\sigma_{3DM}}+\frac{1}{12}$. The numbers M and K, introduced earlier, have not been specified yet. We set $M=(2m+1)C^*(6C^*+1)^2$, where $C^*=C_{\sigma_{3DM}}(J)+C_{\sigma_{3DM}}(B)$, and $K=48(2n+4)NC_{\sigma_{3DM}}$, where $N=n+2m(n-m)+(3m+3)M$, which is simply the total number of jobs. Notice that M and K are well-defined since C^* does not depend on M, and $C_{\sigma_{3DM}}$ does not depend on K. We could have chosen much smaller values for M and K. However, this would make it much harder to verify the correctness of the reduction.

The sum of completion times of the A-jobs in σ is $C_{\sigma_{3DM}}(A)$ if and only if these jobs are scheduled as in σ_{3DM}. Suppose now that some A-jobs are not scheduled according to σ_{3DM}. Let $1-\delta(V_1)$ be the time that machine V_1 spends on processing A-jobs between time $t=0$ and $t=1$. Let $c_1 \leq c_2 \leq \ldots \leq c_M$ be an ordering of the completion times of the A-jobs on machine V_1. The largest completion time, c_M, is at least $1+\delta(V_1)$, and the largest but one is at least $1+\delta(V_1)-\frac{1}{M}$, and so on. Compared to the schedule σ_{3DM} this will increase the sum of completion times for the A-jobs by at least $\lceil\delta(V_1)M\rceil\delta(V_1) \geq \delta(V_1)^2M$. Now let $1-\delta(V_i)$, $1-\delta(W_i)$, and $1-\delta(Z)$ be the time that, respectively, machine V_i, W_i, and Z spends on processing A-jobs between time $t=0$ and respectively $t=1$, $t=2$, and $t=3$ $(1 \leq i \leq m)$. Let $\delta=\delta(V_1)+\ldots+\delta(V_m)+\delta(W_1)+\ldots+\delta(W_m)+\delta(Z)$. For the total completion of the A-jobs we obtain

$$\begin{array}{rcl} C_\sigma(A) & \geq & C_{\sigma_{3DM}}(A)+\sum_{i=1}^{m}\delta(V_i)^2M+\sum_{i=1}^{m}\delta(W_i)^2M+\delta(Z)^2M \\ & \geq & C_{\sigma_{3DM}}(A)+\delta^2M/(2m+1). \end{array} \tag{3}$$

If this $\delta \ll 1$ time is used for processing a J-job, then at most a fraction $\delta/2$ of this job can be processed in this time. For B-jobs this fraction is even smaller. Now consider the problem in which all processing times of J- and B-jobs are multiplied by a factor $1-\delta/2$, slow machines may not be used, and the machines from V, W and Z may not be used until time 1, 2 and 3 respectively. From Problem 1 it follows that the sum of completion times of the J- and B-jobs in this scaled problem, and thus also in the original problem, is at least $(1-\frac{1}{2}\delta)(C^*+\frac{1}{6})$. Together with (3) this gives the following inequality.

$$C_\sigma \geq C_{\sigma_{3DM}}(A) + \delta^2 M/(2m+1) + (1-\frac{1}{2}\delta)(C^* + \frac{1}{6}). \tag{4}$$

If $\delta^2 M/(2m+1) > C^*$, then certainly $C_\sigma \geq C_{\sigma_{3DM}} + \frac{1}{12}$, and we are done. So assume the opposite and substitute the value of M. This gives us $\delta \leq (6C^*+1)^{-1}$. Next we substitute this value for δ in the last term of (4) and again obtain the desired result:

$$\begin{array}{rcl} C_\sigma & \geq & C_{\sigma_{3DM}}(A) + \delta^2 M/(2m+1) + (1-\frac{1}{2}(6C^*+1)^{-1})(C^*+\frac{1}{6}) \\ & = & C_{\sigma_{3DM}}(A) + \delta^2 M/(2m+1) + C_{\sigma_{3DM}}(J) + C_{\sigma_{3DM}}(B) + \frac{1}{12} \\ & > & C_{\sigma_{3DM}} + \frac{1}{12} . \end{array}$$

Problem 3: Let σ be an optimal schedule for Problem 3. We will prove that $C_\sigma \geq C_{\sigma_{3DM}} + 1/24$. Suppose that some parts of jobs are scheduled on slow machines. The sum of the fractions of all jobs scheduled on slow machines can not be more than $C_{\sigma_{3DM}}/K$ since the total processing time of these jobs would already exceed $C_{\sigma_{3DM}}$. From σ we define a new schedule in three steps. First, remove all the work that is scheduled on slow machines. Secondly, shift the remaining schedule to the right over a time $1/(24N)$. That is, all work is postponed by $1/(24N)$. Thirdly, reschedule the removed work on fast machines between $t=0$ and $t=1/(24N)$. This is possible since the total processing time of this work is at most $(2m+4)C_{\sigma_{3DM}}/K = 1/(24N)$ if completely scheduled on fast machines. In this new schedule no job is scheduled on a slow machine, and the increase in the total sum of completion times is at most 1/24. Using the lower bound for Problem 2 we obtain

$$C_\sigma + \frac{1}{24} \geq C_{\sigma_{3DM}} + \frac{1}{12} \quad \Rightarrow \quad C_\sigma \geq C_{\sigma_{3DM}} + \frac{1}{24}.$$

We conclude that a perfect matching exists if and only if there exist a schedule σ for which $C_\sigma \leq C_{\sigma_{3DM}}$. $\square$

4 Identical machines with restricted availability

A special case of the unrelated machine model is the model in which each job j has a fixed processing time p_j but the job can only be processed on a job-specific subset of the machines. So the processing time of job j on machine i (p_{ij}) is either p_j or infinite.

McNaughton proved already in 1959 that for identical machines there is no preemptive schedule with a finite number of preemptions for which the total completion time is strictly less than that of the optimal non-preemptive schedule. As we mentioned earlier there is an optimal schedule with at most $O(m^2 n)$ preemptions [14], so McNaugthon's restriction to a finite number of preemptions may be removed. We generalize this theorem of McNaughton to the restricted unrelated machine model.
$R|\text{pmtn}, p_{ij} \in \{p_j, \infty\}| \sum C_j$

Instance: A number α, a set $\{M_1, M_2, \ldots, M_m\}$ of m parallel machines, a set $\{J_1, J_2, \ldots, J_n\}$ of n independent jobs, a set $\{p_j \mid j = 1 \ldots n\}$, and a set of subsets $\{F_j \subseteq \{M_1, \ldots, M_m\} \mid j = 1 \ldots n\}$. The processing time of job J_j on machine M_i is p_j if $M_i \in F_j$ and is infinite if $M_i \notin F_j$.
Question: Does there exist a preemptive schedule for which $\sum_{j=1}^{n} C_j \leq \alpha$?

Theorem 3 *For the problem $R|pmtn, p_{ij} \in \{p_j, \infty\}| \sum C_j$ there exists an optimal schedule that is non-preemptive.*

Proof. Suppose the theorem is not true. Then there exists an instance with the smallest number of jobs for which its optimal preemptive schedule has a strictly smaller total completion time than any non-preemptive schedule. Let this be instance I with m machines and n jobs. Let the feasible schedule σ^* be optimal among the non-preemptive schedules and let σ be a feasible schedule with $C_\sigma < C_{\sigma^*}$. Since we can assume that the number of preemption is finite [14], we assume that σ has the smallest number of preemptions among all feasible schedules σ for I for which $C_\sigma < C_{\sigma^*}$.

Of course we may assume that all machines are used for processing in σ since otherwise we simply delete an unused machine from the instance. Let T_i ($1 \leq i \leq m$) be the latest moment at which machine i is processing a job. Let this job be J_i and assume $T_1 \leq \cdots \leq T_m$. We distinguish between the case that all jobs J_i ($1 \leq i \leq m$) are different and the case where at least two are equal.

First assume that all jobs J_i ($1 \leq i \leq m$) are different. It is obvious that $C_\sigma(J_i) \geq T_i$ and $\sum_{i=1}^{m} T_i \geq \sum_{j=1}^{n} p_j$, hence $\sum_{i=1}^{m} C_\sigma(J_i) \geq \sum_{j=1}^{n} p_j$. Define the instance I' from I by removing the jobs $J_1, \ldots, J_m$ from the instance. Now we know that for this instance there exists an optimal schedule $\sigma^{'}$ that is non-preemptive. From $\sigma^{'}$ we construct a non-preemptive schedule $\sigma^{''}$ for I by simply adding the job J_i at the end on machine i. Clearly $\sigma^{'}$ and $\sigma^{''}$ do not have idle time so $\sum_{i=1}^{m} C_{\sigma''}(J_i) = \sum_{j=1}^{n} p_j$. Hence the total completion time for this schedule is $C_{\sigma''} = C_{\sigma'} + \sum_{j=1}^{n} p_j$. Since $\sigma^{'}$ is optimal for $I^{'}$, which contains only a subset of the jobs of I, it also holds that $C_\sigma \geq C_{\sigma'} + \sum_{i=1}^{m} C_\sigma(J_i) \geq C_{\sigma''}$. Hence the total completion time of the non-preemptive schedule $\sigma^{''}$ is at most the total completion time of the schedule σ. We obtained a contradiction.

Now assume that there are two jobs, J_j and J_k, $1 \leq j < k \leq m$, that are the same. We make a small change in the schedule as follows. Until time T_j nothing changes and from time T_j job J_j is only processed on machine j. Note that this reduces the number of preemptions of job j while its completion time does not increase. For all other jobs the number of preemptions and completion time remain the same. Again we obtain a contradiction. □

Since the non-preemptive problem can be solved in $O(n^3)$ time even in the more general case of unrelated machines [11],[4] we have the following corollary.

Corollary 4.1 *The problem* $R|\text{pmtn}, p_{ij} \in \{p_j, \infty\}|\sum C_j$ *can be solved in* $O(n^3)$ *time.*

McNaughton proved his theorem for identical machines also for the total weighted completion time objective. So NP-harness of the preemptive problem $P|\text{pmtn}|\sum w_j C_j$ follows from the NP-hardness of the non-preemptive problem $P||\sum w_j C_j$. It follows immediately that the two more general problems $R|p_{ij} \in \{p_j, \infty\}|\sum w_j C_j$ and $R|\text{pmtn}, p_{ij} \in \{p_j, \infty\}|\sum w_j C_j$ are also NP-hard. However, it is not true that preemption is redundant for these problems as illustrated by the next example.

Example 4.1 We define an instance with two machines and three jobs. The processing times are $p_{11} = 1, p_{21} = \infty, p_{12} = \infty, p_{22} = 1, p_{13} = p_{23} = 2$, and the weights are $w_1 = w_2 = 1, w_3 = 2$. Now job 1 starts at time 0 on machine 1, job 2 starts at time 1 on machine 2, and job 3 is scheduled from 0 to 1 on machine 2 and from 1 to 2 on machine 1. The total weighted completion time for this schedule is 7. Any non-preemptive schedule has value at least 8. □

References

[1] P. Brucker and S.A. Kravchenko: Preemption can make parallel machine scheduling problems hard, Osnabruecker *Schriften zur Mathematik*, Reihe P (Preprints), Heft **211**, (1999).

[2] P. Brucker, S.A. Kravchenko and Y.N. Sotskov, Preemptive job-shop scheduling problems with a fixed number of jobs, *Mathematical Methods of Operations Research* **49** (1999), 41–76.

[3] P. Brucker. Personal communication.

[4] J. Bruno, E.G. Coffman Jr, R. Sethi, Scheduling independent tasks to reduce mean finishing time, *Communications of the ACM* **17** (1974), 382–387.

[5] J. Du and J.Y.-T. Leung, Minimizing the number of late jobs on unrelated machines, *Operations Research Letters* **10** (1991), 153–158.

[6] J. Du, J.Y.-T. Leung, Minimizing mean flow time with release time constraints, *Theoretical Computer Science* **75** (1990) 347–355.

[7] J. Du, J.Y.-T. Leung, G.H. Young, Scheduling chain-structured tasks to minimize makespan and mean flow time, *Information and Computation* **92** (1991) 219–236.

[8] M.R. Garey, D.S. Johnson, Strong NP-completeness results: motivation, examples and implications, *Journal of the Association for Computing Machinery* **25** (1978), 499–508.

[9] Gonzalez, Optimal mean finish time preemptive schedules, Technical Report **220**, Computer Science Department, Pennsylvania State University (1977).

[10] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics* **4** (1979), 287–326.

[11] W.A. Horn, Minimizing average flow time with parallel machines, *Operations Research* **21** (1973), 846–847.

[12] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, New York (1972), 85–103.

[13] S.A. Kravchenko, Y.N. Sotskov, Optimal makespan schedule for three jobs on two machines, *Mathematical Methods of Operations Research* **43** (1996), 233–238.

[14] E.L. Lawler, J. Labetoulle, On preemptive scheduling of unrelated parallel processors by linear programming, *Journal of the Association for Computing Machinery* **25** (1978), 612–619.

[15] E.L. Lawler, Recent results in the theory of machine scheduling, in: A. Bachem, M. Grötchel and B. Korte (eds.), *Mathematical Programming: the State of the Art- Bonn 1982*, Springer, Berlin-New York (1983), 202–234.

[16] R. McNaughton, Scheduling with deadlines and loss functions, *Management Science* **6** (1959), 1–12

[17] R.A. Sitters, Two NP-hardness results for preemptive minsum scheduling of unrelated parallel machines, Proc. 8th International IPCO Conference, June 13-15, 2001, Utrecht, the Netherlands, *Lecture Notes in Computer Science* **2081**, Springer, 2001, 396-405.