Schedules with a Single Preemption on Uniform Parallel Machines

Alan J. Soper and Vitaly A. Strusevich
Department of Mathematical Sciences, University of Greenwich,
Old Royal Naval College, Park Row, Greenwich, London SE10 9LS, U.K.
e-mail: {A.J.Soper,V.Strusevich}@greenwich.ac.uk

Abstract

For a scheduling problem to minimize the makespan on parallel machines, we consider schedules with at most one preemption. We show that in the case of two machines the problem is solvable in polynomial time. For $m \geq 3$ uniform parallel machines, we measure the quality of a single preemption as the worst-case ratio of the makespan of an optimal schedule with at most one preemption over the makespan of an optimal preemptive schedule. We show that the global bound on such a ratio is $2 - 2/m$.

Keywords: Scheduling, uniform parallel machines, power of preemption, quality of a single preemption

1 Introduction

In this paper, we analyze schedules with at most one preemption for scheduling problems on parallel machines.

In parallel machine scheduling, we are given the jobs of set $N = \{J_1, J_2, \dots, J_n\}$ and m parallel machines $M_1, M_2, \dots, M_m$. If a job $J_j \in N$ is processed on machine M_i alone, then its processing time is known to be p_{ij}. There are three main types of scheduling systems with parallel machines: (i) *identical* parallel machines, for which the processing times are machine-independent, i.e., $p_{ij} = p_j$; (ii) *uniform* parallel machines, which have different speeds, so that $p_{ij} = p_j/s_i$, where s_i denotes the *speed* of machine M_i; and (iii) *unrelated* parallel machines, for which the processing time of a job depends on the machine assignment.

In a non-preemptive schedule, each job is processed on the machine it is assigned to without interruption. In a preemptive schedule, the processing of a job on a machine can be interrupted at any time and then resumed either on this or on any other machine, provided that the job is not processed on two or more machines at a time, and the amount of processing assigned to each machine guarantees that the job is fully completed.

In all problems considered in this paper the objective is to minimize the *makespan,* i.e., the maximum completion time across all m machines. For a schedule S, the makespan is denoted by $C_{\max}(S)$. For an instance of a scheduling problem on parallel machines, let $S^*_{(q)}$ and S^*_p denote an optimal schedule with at most q preemptions, and an optimal preemptive schedule which uses an unlimited number of preemptions, respectively. We will refer to schedules with an unlimited number of preemptions as simply preemptive. The case $q = 0$ corresponds to a non-preemptive schedule, and an optimal non-preemptive schedule is denoted either by $S_{(0)}$ or by S^*_{np}.

The number of preemptions in an optimal schedule S_p^* does not exceed $m-1$ in the case of identical machines [12] and $2\left(m-1\right)$ in the case of uniform machines [7].

For $m \geq 3$, the problem of finding an optimal preemptive schedule with at most $q \leq m-2$ preemptions on identical parallel machines is NP-hard [14] and the corresponding problems on uniform or unrelated machines are obviously no easier. If any number of preemptions is allowed, then all these problems are polynomially solvable, even in the most general settings with unrelated machines. See a focused survey [3] on parallel machine scheduling with the makespan objective for details and references.

For $m \geq 3$ uniform machines, the problem of finding an optimal schedule with the number of preemptions q such that q is even and $q \leq 2\left(m-3\right)$ is shown to be NP-hard by [17].

For unrelated machines, it has been shown by [9] that an optimal preemptive schedule requires no more than $O\left(m^2\right)$ preemptions. Clearly finding optimal schedules with a limited number of preemptions is no easier than for the uniform machines case, but no results have yet quantified any difference.

This paper resolves the complexity issue for $q=1$, on two uniform machines and two unrelated machines in Section 3.

For a scheduling problem on parallel machines (parallel, uniform or unrelated), the quality of a schedule with a restricted number of preemptions is defined as the worst-case ratio of the makespan of an optimal schedule with at most q preemptions over the makespan of an optimal preemptive schedule. In Section 4, we show that in the case of $q=1$, the global bound on such a ratio for the problem on uniform parallel machines is $2-2/m$, $m \geq 3$, and this bound is tight.

2 Scheduling with a Restricted Number of Preemptions: Review

Consider an instance of a scheduling problem to minimize the makespan $C_{\max}$ on m parallel machines (identical, uniform or unrelated). For the corresponding problem, we measure the quality of a schedule with at most q preemptions as a tight upper bound $\rho_m^{(q)}$ on the ratio $C_{\max}(S_{(q)}^*)/C_{\max}(S_p^*)$ across all instances of the problem at hand. The value of $\rho_m^{(q)}$ determines what can be gained regarding the maximum completion time if instead of at most q preemptions any number of preemptions is allowed. For $q=0$ this concept coincides with a well-studied notion of the power of preemption.

In order to determine the exact value of $\rho_m^{(q)}$ for a particular problem and to give the concept some practical meaning, the following should be done:

(i) demonstrate that the inequality

$$\frac{C_{\max}\left(S_{(q)}^*\right)}{C_{\max}\left(S_p^*\right)} \leq \rho_m^{(q)} \tag{1}$$

holds for all instances of the problem;

(ii) exhibit instances of the problem for which (1) holds as equality, i.e., show that the value of $\rho_m^{(q)}$ is tight; and

(iii) develop a polynomial-time algorithm that finds a heuristic schedule $S_{(q)}$ with at most q preemptions such that

$$\frac{C_{\max}\left(S_{(q)}^{*}\right)}{C_{\max}\left(S_{p}^{*}\right)} \leq \frac{C_{\max}\left(S_{(q)}\right)}{C_{\max}\left(S_{p}^{*}\right)} \leq \rho_{m}^{(q)}. \tag{2}$$

Most of the known results in this area address the situation of $q = 0$, i.e., are aimed at comparing an optimal non-preemptive schedule with an optimal preemptive schedule.

If the machines are identical parallel, then it is known that $\rho_m^{(0)} = 2 - 2/(m+1)$, as independently proved in [1] and [10]. It is shown in [13] that the value of $\rho_m^{(0)}$ can be reduced for some instances that contain jobs with fairly large processing times, i.e., longer than the average machine load.

According to [18], for m uniform parallel machines $\rho_m^{(0)} = 2 - 1/m$. In [16], the necessary and sufficient conditions under which the global bound of $2 - 1/m$ is tight are given. If the makespan of an optimal preemptive schedule S_p^* is defined by the ratio of the total processing time of $r < m$ longest jobs over the total speed of r fastest machines, it is shown in [16] that the tight bound on the power of preemption $\rho_m^{(0)}$ is $2 - 1/\min\{r, m-r\}$.

For two uniform machines, a parametric analysis of the power of preemption $\rho_2^{(0)}$ with respect to the speed of the faster machine is independently performed in [8] and [15]. For $m = 3$, a similar analysis is contained in [15], provided that the machine speeds take at most two values.

For unrelated parallel machines, a rounding procedure that is attributed to Shmoys and Tardos and reproduced in [11] and [4] finds non-preemptive schedules $S_{(0)}$ such that the bound (2) holds for $\rho_m^{(0)} = 4$. This bound is tight, as proved in [4].

Studies that compare optimal schedules with a limited number of preemptions to optimal preemptive schedules are fewer in number. For identical machines, Braun and Schmidt [1] prove that $\rho_m^{(q)} = (2m)/(m+q+1)$, where $0 \leq q \leq m-1$, and that this bound is tight. Jiang et al. in [8] perform a parametric analysis of a single preemption for two uniform machines with speeds s' and s'', where $s' \geq s''$, from which it follows that

$$\rho_2^{(1)} = \frac{2(s')^2 + s's'' - (s'')^2}{2(s')^2}, \tag{3}$$

This function attains its maximum value of 9/8 when $s' = 2s''$.

Among the results on the power of preemption measured with respect to the objective functions other than the makespan, we mention the recent studies on the single machine problem to minimize the weighted completion time [5] and on the problem on uniform parallel machines to minimize the total completion time [6].

3 Complexity Results

In this section, we study the complexity of the problems on parallel machines, provided that at most one preemption is allowed. Extending standard scheduling notation, we denote the corresponding problems by $\alpha m\left|\#pmtn \leq 1\right| C_{\max}$, where m is the number of machines and $\alpha \in \{P, Q, R\}$ for identical, uniform and unrelated machines, respectively.

In the case of identical machines, it follows from [12] that problem $P2\left|\#pmtn \leq 1\right| C_{\max}$ is solvable in $O(n)$ time, since in the case of two machines an optimal preemptive schedule has at most one preemption. On the other hand, $Pm\left|\#pmtn \leq 1\right| C_{\max}$ is NP-hard for each $m \geq 3$; see [1] and [14]. To clarify the complexity status of the remaining scheduling problems on parallel machines with at most one preemption we address the problems on two uniform and two unrelated machines.

First, we show that problem $Q2\left|\#pmtn \leq 1\right| C_{\max}$ can be solved in polynomial time. We present this result in a generic way, so that the two machines are denoted by M' and M'', while their speeds are s' and s'', respectively, where $s' \geq s''$. Assume that the jobs are numbered in accordance with the LPT rule, i.e., in non-increasing order of their processing times

$$p_1 \geq p_2 \geq \cdots \geq p_n. \tag{4}$$

For a non-empty subset $R \subseteq N$, define

$$p(R) = \sum_{j \in R} p_j,$$

and for completeness define $p(\varnothing) = 0$.

It follows from [7] that problem $Q2\left|pmtn\right| C_{\max}$ is solvable in $O(n)$ time, provided that there is no restriction on the number of preemptions. Besides,

$$C_{\max}(S^*) = \max\left\{p_1/s', T_2\right\},$$

where

$$T_2 = \frac{p(N)}{s' + s''}. \tag{5}$$

Algorithm Q2Pr1

Step 1. Compute T_2 by (5). If

$$p_1 < s'T_2,$$

go to Step 2; otherwise, output a non-preemptive schedule $S_{(1)}$ with job J_1 on machine M' and the other jobs on machine M'' and stop.

Step 2. Scanning the jobs in the order of their numbering, find job J_k such that

$$\sum_{j=1}^{k-1} p_j < s'T_2, \ \sum_{j=1}^{k} p_j \geq s'T_2.$$

Step 3. Compute

$$x_k = s'T_2 - \sum_{j=1}^{k-1} p_j, \ y_k = p_k - x_k.$$

If

$$\frac{y_k}{s''} > \frac{1}{s'} \sum_{j=1}^{k-1} p_j,$$

go to Step 4; otherwise, output the following schedule $S_{(1)}$: on M' the jobs $J_1, \ldots, J_{k-1}$ are processed in any order, followed by a part of job J_k for x_k/s' time units; on M'' process a part of job J_k for y_k/s'' time units, followed by an arbitrary sequence of jobs $J_{k+1}, \ldots, J_n$. Stop.

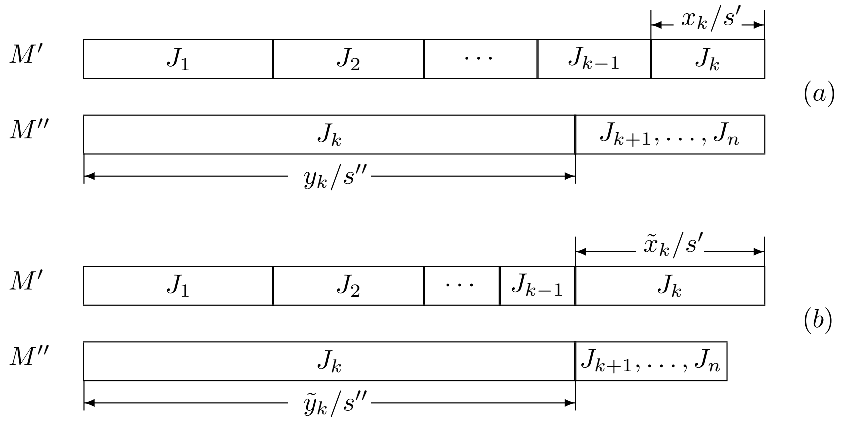

Figure 1: (a) Schedule $S_{(1)}$ in Step 3; (b) schedule $S_{(1)}$ in Step 4

Step 4. For job J_k, compute the values $\tilde{x}_k$ and $\tilde{y}_k$ such that

$$\frac{\tilde{y}_k}{s''} = \frac{1}{s'}\sum_{j=1}^{k-1} p_j, \ \ \tilde{x}_k = p_k - \tilde{y}_k.$$

Output the following schedule $S_{(1)}$: on M' the jobs $J_1, \ldots, J_{k-1}$ are processed in any order, followed by a part of job J_k for $\tilde{x}_k/s'$ time units; on M'' process a part of job J_k for $\tilde{y}_k/s''$ time units, followed by an arbitrary sequence of jobs $J_{k+1}, \ldots, J_n$. Stop.

Theorem 1 *Schedule $S_{(1)}$ found by Algorithm Q2Pr1 is optimal for problem $Q2\,|\#pmtn \le 1|\,C_{\max}$.*

Proof: If $p_1 \ge s'T_2$ then $p_1\,(s' + s'') \ge s'p\,(N)$, i.e.,

$$\frac{p_1}{s'} \ge \frac{p\,(N\backslash\,\{1\})}{s''}.$$

For schedule $S_{(1)}$ found in Step 1 machines M' and M'' terminate at p_1/s' and at $p\,(N\backslash\,\{1\})\,/s''$, respectively. Thus,

$$C_{\max}\left(S_{(1)}\right) = p_1/s',$$

i.e., schedule $S_{(1)}$ is a non-preemptive optimal schedule for problem $Q2\,|pmtn|\,C_{\max}$, and therefore for problem $Q2\,|\#pmtn \le 1|\,C_{\max}$.

We come to Step 2 if

$$p_1 < s'T_2.$$

Since

$$\frac{1}{s'}\sum_{j=1}^{n} p_j > \frac{1}{s'+s''}\sum_{j=1}^{n} p_j = T_2,$$

it follows that job J_k exists.

For schedule $S_{(1)}$ found in Step 3, both machines terminate at time T_2; see Figure 1(a). This schedule is feasible since the part of the preempted job J_k completes on M'' at time

y_k/s'', while this job starts on M' at time $\frac{1}{s'}\sum_{j=1}^{k-1}p_j$, which produces no clash due to $\frac{y_k}{s''} \leq \frac{1}{s'}\sum_{j=1}^{k-1}p_j$. Since $C_{\max}\left(S_{(1)}\right) = T_2$, schedule $S_{(1)}$ is an optimal schedule for problem $Q2\left|pmtn\right|C_{\max}$, and therefore for problem $Q2\left|\#pmtn \leq 1\right|C_{\max}$.

We come to Step 4 if

$$\frac{y_k}{s''} > \frac{1}{s'}\sum_{j=1}^{k-1}p_j,$$

which implies that

$$\frac{y_k}{s''} + \frac{x_k}{s'} > T_2. \tag{6}$$

Combining (6) with $p_k/s' \leq p_1/s' < T_2$ yields $s' > s''$. Define

$$\tilde{T}_2 = \frac{\tilde{x}_k}{s'} + \frac{\tilde{y}_k}{s''}.$$

Since

$$\frac{y_k}{s''} > \frac{1}{s'}\sum_{j=1}^{k-1}p_j = \frac{\tilde{y}_k}{s''},$$

it follows that

$$y_k > \tilde{y}_k,\ x_k < \tilde{x}_k.$$

Then

$$T_2 = \frac{\sum_{j=1}^{k-1}p_j + x_k}{s\prime} = \frac{\tilde{y}_k}{s''} + \frac{x_k}{s'} < \frac{\tilde{x}_k}{s'} + \frac{\tilde{y}_k}{s''} = \tilde{T}_2.$$

Let $S'_{(1)}$ denote the best schedule for problem $Q2\left|\#pmtn \leq 1\right|C_{\max}$ in which the jobs of set $N_k = \{J_1, \cdots, J_k\}$ are processed without preemption. If all these jobs are assigned to machine M' then

$$\begin{aligned} C_{\max}\left(S'_{(1)}\right) &= \frac{1}{s'}p\left(N_k\right) = \frac{p_k}{s'} + \frac{1}{s'}\sum_{j=1}^{k-1}p_j = \frac{p_k}{s'} + \frac{\tilde{y}_k}{s''} \\ &> \frac{\tilde{x}_k}{s'} + \frac{\tilde{y}_k}{s''} = \tilde{T}_2. \end{aligned}$$

If at least one of these jobs is assigned to be processed on machine M'', then due to the the LPT numbering of jobs we deduce that

$$C_{\max}\left(S'_{(1)}\right) \geq \frac{p_k}{s''} = \frac{\tilde{x}_k}{s''} + \frac{\tilde{y}_k}{s''} > \frac{\tilde{x}_k}{s'} + \frac{\tilde{y}_k}{s''} = \tilde{T}_2.$$

Thus, in an optimal schedule $S_{(1)}$ no job of set N_k can be processed in full on machine M'' and exactly one job of set N_k must be processed with preemption.

Take a job $J_\ell \in N_k$ and consider a schedule in which only job J_ℓ is processed with preemption. Define

$$\frac{\tilde{y}_\ell}{s''} = \frac{1}{s'}p\left(N_k \backslash \{\ell\}\right),\ \tilde{x}_\ell = p_\ell - \tilde{y}_\ell.$$

Introduce a schedule $S\left(\ell\right)$, in which on machine M' the jobs of set $N_k \backslash \{J_\ell\}$ are processed without preemption in accordance with an arbitrary sequence and are followed by the processing of J_ℓ for $\tilde{x}_\ell / s'$ time units, while on machine M'' job J_ℓ is processed for $\tilde{y}_\ell / s''$ time units, followed by an arbitrary sequence of jobs of set $N \backslash N_k$.

In schedule $S(\ell)$ machine M' terminates at time

$$\begin{aligned}\frac{\tilde{y}_\ell}{s''} + \frac{\tilde{x}_\ell}{s'} &= \frac{\tilde{y}_\ell}{s''} + \frac{p_\ell - \tilde{y}_\ell}{s'} = \frac{p_\ell}{s'} + \tilde{y}_\ell \left(\frac{1}{s''} - \frac{1}{s'} \right) \\ &= \frac{p_\ell}{s'} + \frac{(p(N_k) - p_\ell)}{s'} \left(1 - \frac{s''}{s'} \right) = \frac{s'' p_\ell + (s' - s'') p(N_k)}{(s')^2}.\end{aligned}$$

The final right-hand side of the above expression increases in p_ℓ, thus it takes its minimum for $\ell = k$. Such a minimum is equal to $\tilde{T}_2 > T_2$.

Notice that in schedule $S(\ell)$ machine M'' must terminate before time T_2. Indeed, assume that the machines terminate at times $C' = P'/s'$ and $C'' = P''/s''$ and $\min\{C', C''\} > T_2$. Then we derive a contradiction $p(N) = P' + P'' > s'T_2 + s''T_2 = p(N)$.

Thus, the makespan $C_{\max}(S(\ell))$ is determined by the time that machine M' terminates. As long as no job of set N_k is processed in full on machine M'', schedule $S(\ell)$ is optimal among all schedules in which job J_ℓ is processed with preemption, since for machine M' it guarantees the smallest possible allocation of the jobs other than J_ℓ and the smallest possible part of job J_ℓ.

The smallest makespan value of $C_{\max}(S(\ell))$ is delivered for $\ell = k$. Since schedule $S_{(1)}$ found in Step 4 is actually schedule $S(\ell)$ for $\ell = k$ (see Figure 1(b)), we deduce that $S_{(1)}$ is optimal for problem $Q2 \left| \#pmtn \le 1 \right| C_{\max}$. ■

Algorithm Q2Pr1 requires $O(n)$ time, provided that the LPT numbering of the jobs is available. Obtaining such a numbering can be seen as a preprocessing stage, which takes $O(n \log n)$ time. Thus, problem $Q2 \left| \#pmtn \le 1 \right| C_{\max}$ is solvable in $O(n \log n)$ time.

Theorem 1 completely resolves the complexity status of the problem of finding an optimal schedule with a single preemption on uniform machines: polynomially solvable on two machines and NP-hard on three or more machines (even identical). For a more general processing system, with two unrelated machines, finding an optimal schedule with at most one preemption becomes NP-hard, as stated below.

For a non-empty index set R, similarly to the earlier introduced notation $p(R)$, for a sequence $e_1, e_2, \dots$ the notation $e(R) = \sum_{i \in R} e_i$ is used. Notation $a(R), b(R)$ is defined analogously.

Theorem 2 *Problem $R2 \left| \#pmtn \le 1 \right| C_{\max}$ is NP-hard in the ordinary sense.*

Proof: We provide reduction from the following well-known NP-complete problem.

Partition: Given r positive integers e_j and an integer E such that $e_j \le E$ and $\sum_{j=1}^{r} e_j = 2E$, does there exist a partition of the index set $R = \{1, 2, \dots, r\}$ into two subsets R_1 and R_2 such that $e(R_1) = e(R_2) = E$?

Consider problem $R2 \left| \#pmtn \le 1 \right| C_{\max}$ with two machines A and B and the set $N = \{J_1, J_2, \dots, J_n\}$ of jobs. If a job J_j is processed on machine A (machine B) alone, then its processing time is equal to a_j time units (to b_j units, respectively).

Given an instance of Partition, define the following instance of the decision version of

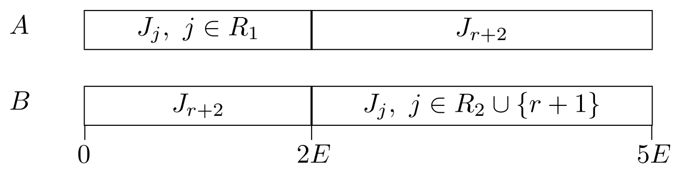

Figure 2: Schedule $S_{(1)}$

problem $R2\,|\#pmtn \le 1|\,C_{\max}$:

$$\begin{aligned} n &= r+2;\ a_j = 2e_j,\ b_j = e_j,\ j \in R; \\ a_{r+1} &= 11E,\ b_{r+1} = 2E;\ a_{r+2} = 6E,\ b_{r+2} = 4E. \end{aligned}$$

It is required to verify whether there exists a schedule $S_{(1)}$ with a single preemption such that $C_{\max}\left(S_{(1)}\right) \le 5E$.

Suppose that Partition has a solution and R_1 and R_2 are the required subsets. Then assign an arbitrary sequence of jobs J_j with $j \in R_1$ to be processed on machine A in the time interval $[0, 2E]$, and an arbitrary sequence of jobs J_j with $j \in R_2$ and job J_{r+1} to be processed on machine B scheduled in the time interval $[2E, 5E]$. Job J_{r+2} is processed with preemption, in the time interval $[0, 2E]$ on machine B and in the time interval $[2E, 5E]$ on machine A. For the resulting schedule $S_{(1)}$ both machines complete their processing at time $5E$. See Figure 2. The preemptive processing of job J_{r+2} is feasible, since

$$\frac{2E}{4E} + \frac{3E}{6E} = 1.$$

Suppose now that schedule $S_{(1)}$ exists. First, notice that in $S_{(1)}$ either job J_{r+1} or job J_{r+2} must be processed with preemption. Indeed, in order to complete by time $5E$ neither of these jobs can be processed without preemption on machine A, and if both are assigned to be processed non-preemptively on machine B they will complete no earlier than time $6E$.

Suppose that job J_{r+2} is processed without preemption, while job J_{r+1} with preemption. To complete by time $5E$, job J_{r+2} must be assigned to machine B. Then job J_{r+1} can be processed on B for at most E time units, which is 50% of its overall processing. No less than 50% of job J_{r+1} must be performed on machine A, which makes at least $5.5E$ units of processing.

Thus, in schedule $S_{(1)}$ job J_{r+2} is processed with preemption and job J_{r+1} is processed on machine B.

Suppose that in $S_{(1)}$ job J_{r+2} is processed on machine A for $3E + x$ time units and on machine B for $2E - y$ time units, where x and y are both positive. To finish this job by time $5E$ we must have that $x \le y$. On the other hand, to guarantee that job J_{r+2} completes in full the equality

$$\frac{2E - y}{4E} + \frac{3E + x}{6E} = 1$$

must hold, which implies that $y = \frac{2}{3}x < x$; a contradiction.

Now suppose that in $S_{(1)}$ job J_{r+2} is processed on machine A for $3E - x$ time units and on machine B for $2E + y$ time units, where x and y are both positive. To finish this job by time $5E$ we must have that $x \geq y$. To guarantee that job J_{r+2} completes in full the equality

$$\frac{2E+y}{4E}+\frac{3E-x}{6E}=1$$

must hold, which implies that $y=\frac{2}{3}x$.

Let N_A and N_B denote the subsets of R such that the jobs J_j with $j \in N_A$ and with $j \in N_B$ are assigned to be processed in schedule $S_{(1)}$ on machine A and on machine B, respectively. Then for machine B to complete its jobs by time $5E$ the inequality

$$\left(2E+\frac{2x}{3}\right)+2E+e\left(N_B\right) \leq 5E,$$

must hold, which implies that $e\left(N_B\right) \leq E-\frac{2x}{3}$, so that $e\left(N_A\right) \geq E+\frac{2x}{3}$. Then machine A completes its jobs no earlier than time

$$2\left(E+\frac{2x}{3}\right)+3E-x=5E+\frac{x}{3}.$$

We deduce that for schedule $S_{(1)}$ to exist, job J_{r+2} must be processed on machine B for $2E$ time units and on machine A for $3E$ time units. This implies that in $S_{(1)}$, $b\left(N_B\right)=e\left(N_B\right)=E$, i.e., the sets N_A and N_B form a solution to PARTITION. ■

4 Single Preemption on m Uniform Parallel Machines

In this section, we establish a global tight bound on how much is lost if at most one preemption is allowed on m uniform parallel machines, compared to the makespan of an optimal schedule with any number of preemptions.

An instance I of the problem with n jobs and m uniform parallel machines is defined by the list $\mathcal{L}_n=(p_1,p_2,\ldots,p_n)$ of the processing times of the jobs and the list $\mathcal{M}_m=(s_1,s_2,\ldots,s_m)$ of the machine speeds. In what follows, we assume that both lists are non-increasing. In other words, the jobs are numbered in accordance with the LPT rule (4) and the machines are numbered in non-increasing order of their speeds, i.e., $s_1 \geq s_2 \geq \cdots \geq s_m$.

Feasible non-preemptive and preemptive schedules for an instance $I=(\mathcal{L}_n,\mathcal{M}_m)$ are denoted by $S_{np}\left(\mathcal{L}_n,\mathcal{M}_m\right)$ or $S_{np}\left(I\right)$, and by $S_p\left(\mathcal{L}_n,\mathcal{M}_m\right)$ or $S_p\left(I\right)$, respectively; the corresponding optimal non-preemptive and preemptive schedules are denoted by $S^*_{np}\left(\mathcal{L}_n,\mathcal{M}_m\right)$ or $S^*_{np}\left(I\right)$ and by $S^*_p\left(\mathcal{L}_n,\mathcal{M}_m\right)$ or $S^*_p\left(I\right)$, respectively. The reference to an instance may be omitted if it is clear which instance is being discussed.

In our analysis of the power of preemption, we will need precise expressions for the makespan of the preemptive schedules. The fastest algorithm for finding an optimal preemptive schedule on uniform parallel machines is due to Gonzalez and Sahni [7] and requires $O(n+m\log m)$ time.

Given an instance $I=(\mathcal{L}_n,\mathcal{M}_m)$, for each v, $1 \leq v \leq m$, define the total speed of the v fastest machines $S_v=\sum_{i=1}^{v} s_i$. Besides, define the set of the v longest jobs $H_v=\{1,2,\ldots,v\}$.

Define $m'=\min\{n,m-1\}$ and

$$T_v=p\left(H_v\right)/S_v,\ 1 \leq v \leq m';\ T_m=p\left(N\right)/S_m. \tag{7}$$

It is well-known (see, e.g., [2]) that for an optimal preemptive schedule $S_p^*(I)$ the makespan is equal to

$$C_{\max}(S_p^*(I)) = \max\left\{\max\left\{T_v | 1 \le v \le m'\right\}, T_m\right\}. \tag{8}$$

Clearly $T_m = p(N)/S_m$ is a lower bound on $C_{\max}(S_p^*(I))$ even when simultaneous execution of jobs is allowed, since it is the average machine load, i.e., the time for processing all jobs, provided all machines are continuously busy.

Definition 1 *For an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$, suppose that in a non-preemptive schedule $S_{np}(I)$ the last completed operation is that of processing job $J_h, 1 \le h \le n$, on machine M_k, $1 \le k \le m$. We call job J_h the* terminal *job and machine M_k the* critical *machine.*

For an instance I, a schedule $S_{(1)}(I)$ with exactly one preemption is defined by

(i) a job $J_\ell \in N$ which is processed with preemption on two machines $M_{\ell'}$ and $M_{\ell''}$ such that $1 \le \ell' < \ell'' \le m$; the actual processing times of job J_ℓ on these machines are equal to $x_\ell/s_{\ell'}$ and $y_\ell/s_{\ell''}$, where $x_\ell + y_\ell = p_\ell$;

(ii) a partition of set $N\backslash\{\ell\}$ into m subsets $N_1, N_2, \ldots, N_m$, where the jobs of set N_i are assigned to be processed on machine M_i, $1 \le i \le m$.

Notice that even in an optimal schedule some of the subsets N_i can be empty, since it may be counterproductive to assign jobs to very slow machines. For a particular instance, it might be optimal not to preempt any job, in which case a schedule $S_{(1)}(I)$ is defined by a partition of set N into m subsets $N_1, N_2, \ldots, N_m$. If there is a preempted job J_ℓ in schedule $S_{(1)}(I)$ then the jobs assigned to machines $M_{\ell'}$ and $M_{\ell''}$ must be arranged in a such a way that the two portions of job J_ℓ do not overlap.

Define

$$\rho_m^{(1)} = 2 - \frac{2}{m}.$$

Below we present an algorithm that creates a schedule $S_{(1)}$ with at most one preemption such that

$$\frac{C_{\max}\left(S_{(1)}\right)}{C_{\max}\left(S_p^*\right)} \le \rho_m^{(1)}.$$

Algorithm LPT1

Step 1. Compute $C_{\max}\left(S_p^*\right)$ in accordance with equation (8) and the completion time bound $B = \rho_m^{(1)} C_{\max}\left(S_p^*\right)$. Form the LPT list $\mathcal{L}_n = (p_1, \ldots, p_n)$. Define

$$N_i := \varnothing,\ p(N_i) := 0,\ 1 \le i \le m.$$

Step 2. For j from 1 to n do

(a) Take job J_j, the first job in the current list $\mathcal{L}_n$. Scanning the machines in the order of their numbering, search for the first machine M_k such that $p(N_k) + p_j \le s_k B$. If such a machine exists, then go to Step 2(b). If $p(N_i) + p_j > s_i B$ for all i, $1 \le i \le m$, then if the preemption has not been used go to Step 2(c); otherwise, go to Step 2(d).

(b) Update
$$N_k := N_k \cup \{j\}\,,\ p(N_k) := p(N_k) + p_j$$
and go to Step 2(e).

(c) First, try to process job J_j as the preempted job J_ℓ. If a pair of machines $M_{\ell'}$ and $M_{\ell''}$ can be selected such that $1 \le \ell' < \ell'' \le m$ and the inequalities
$$\begin{aligned} y_\ell + p(N_{\ell''}) &\le s_{\ell''}B, &(9)\\ \frac{x_\ell}{s_{\ell'}} + \frac{y_\ell}{s_{\ell''}} &\le B, &(10)\end{aligned}$$
hold, where
$$\begin{aligned} x_\ell &= s_{\ell'}B - p(N_{\ell'})\,,\\ y_\ell &= p_\ell - x_\ell;\end{aligned}$$
then choose the pair with smallest value of ℓ'' and assign job J_ℓ to be processed on machine $M_{\ell'}$ for $x_\ell/s_{\ell'}$ time units and on machine $M_{\ell''}$ for $y_\ell/s_{\ell''}$ time units. Define
$$p(N_{\ell'}) := p(N_{\ell'}) + x_\ell,\ p(N_{\ell''}) := p(N_{\ell''}) + y_\ell.$$
If the required pair of machines cannot be found, then go to Step 2(d).

(d) Find machine M_k such that
$$s_kB - p(N_k) = \max\{s_iB - p(N_i)\,|1 \le i \le m\} \tag{11}$$
and return to Step 2(b).

(e) Remove job J_j from list $\mathcal{L}_n$.

Step 3. In the found schedule $S_{(1)}(I)$ machine M_i processes the jobs of set N_i in the LPT order starting from time 0, $1 \le i \le m$. Additionally, if in the loop in Step 2, Step 2(c) has occurred then (i) in the time interval $[0, y_\ell/s_{\ell''}]$ process the jobs of set $N_{\ell'}$ on machine $M_{\ell'}$ and job J_ℓ on machine $M_{\ell''}$, and (ii) at time $y_\ell/s_{\ell''}$ start the jobs of set $N_{\ell''}$ on machine $M_{\ell''}$ and the remaining part of job J_ℓ on machine $M_{\ell'}$.

Algorithm LPT1 scans the jobs in the LPT order and tries to assign each job to the first available machine where it can be completed by time B. The first time that such an assignment is not possible, the corresponding job is assigned to be processed with preemption; see Step 2(c).

To estimate the running time of Algorithm LPT1, notice that forming the list $\mathcal{L}_n$ in Step 1 requires the numbering of the jobs in the LPT order, which takes $O(n \log n)$ time. In Step 2(a) the search for machine M_k requires $O(m)$ time for each j, $1 \le j \le n$. Step 2(c) requires $O(m^2)$ time, but is performed only once. Thus, provided that $n \ge m$, the running time of Algorithm LPT1 is $O(n \log n + nm)$.

Notice that the algorithm iteratively updates the values $p(N_i)$, $1 \le i \le m$, which represent the amount of processing assigned to machine M_i in terms of the original values p_j, $1 \le j \le n$. To maintain that meaning, we go for a slight abuse of notation, when we redefine values $p(N_{\ell'})$ and $p(N_{\ell''})$ in Step 2(c). As a result, in any case $\sum_{i=1}^m p(N_i) = p(N)$.

Theorem 3 *Given an arbitrary instance $I = (\mathcal{L}_n, \mathcal{M}_m)$, let $S_{(1)}(I)$ be a schedule created by Algorithm LPT1 using at most one preemption. Then*

$$\frac{C_{\max}\left(S_{(1)}(I)\right)}{C_{\max}\left(S_p^*(I)\right)} \leq 2 - \frac{2}{m}. \tag{12}$$

Proof: The proof is based on the minimal counterexample technique, often used in worst-case analysis of approximation algorithms. Suppose that the theorem is not true, i.e., there exists an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$, which we call the minimal counterexample, such that

$$\frac{C_{\max}\left(S_{(1)}(\mathcal{L}_n, \mathcal{M}_m)\right)}{C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_m)\right)} > 2 - \frac{2}{m} \tag{13}$$

and no job or machine can be removed from the instance without violating the inequality (13).

For an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$, let $S_{(1)}(I)$ be a schedule found by Algorithm LPT1. As a consequence of choosing the first suitable machine(s) in Steps 2(a) and 2(c) of Algorithm LPT1, if some machines receive no load in schedule $S_{(1)}(I)$, then they will be the slowest machines.

First, consider instances in which the number of jobs is smaller than the number of machines, i.e., the instances $(\mathcal{L}_n, \mathcal{M}_m)$ with $n < m$. Let $\mathcal{M}_n$ be the list of machine speeds obtained from list $\mathcal{M}_m$ by a removal of the $m - n$ slowest machines.

If $n < m$, then in schedule $S_{(1)}(I)$, the jobs are assigned to at most n fastest machines. On the other hand, for each instance I there exists an optimal preemptive schedule $S_p^*(I)$ in which the jobs are assigned to at most n fastest machines. Thus, $C_{\max}\left(S_{(1)}(\mathcal{L}_n, \mathcal{M}_m)\right) = C_{\max}\left(S_{(1)}(\mathcal{L}_n, \mathcal{M}_n)\right)$, while in the preemptive case $C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_m)\right) = \max\{T_v | 1 \leq v \leq n < m\} = C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_n)\right)$.

Since for an instance $(\mathcal{L}_n, \mathcal{M}_m)$ with $n < m$ the removal of the $m - n$ slowest machines does not change the value of the power of preemption, the minimal counterexample cannot be one of these instances. Hence, in our search for the minimal counterexample we only need to consider instances in which there are at least as many jobs as machines.

Suppose that in schedule $S_{(1)}(\mathcal{L}_n, \mathcal{M}_m)$ job J_h is the terminal job and machine M_k is the critical machine. If $h < n$ then Algorithm LPT1 assigns some jobs J_j with $j > h$ after job J_h and they complete (on machines other than M_k) earlier than job J_h. Imagine that these jobs are removed from the instance, so that $\mathcal{L}_h = (p_1, p_2, \ldots, p_h)$ is the corresponding list of the processing times. For the modified instance $(\mathcal{L}_h, \mathcal{M}_m)$, we have

$$C_{\max}\left(S_{(1)}(\mathcal{L}_h, \mathcal{M}_m)\right) = C_{\max}\left(S_{(1)}(\mathcal{L}_n, \mathcal{M}_m)\right); \; C_{\max}\left(S_p^*(\mathcal{L}_h, \mathcal{M}_m)\right) \leq C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_m)\right),$$

so that

$$\frac{C_{\max}\left(S_{(1)}(\mathcal{L}_h, \mathcal{M}_m)\right)}{C_{\max}\left(S_p^*(\mathcal{L}_h, \mathcal{M}_m)\right)} \geq \frac{C_{\max}\left(S_{(1)}(\mathcal{L}_n, \mathcal{M}_m)\right)}{C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_m)\right)} > 2 - \frac{2}{m}.$$

Thus, if $h < n$ we deduce that instance $(\mathcal{L}_n, \mathcal{M}_m)$ cannot be the minimal counterexample, and we must have that $h = n$. In other words, for the minimal counterexample $(\mathcal{L}_n, \mathcal{M}_m)$ Algorithm LPT1 finds a schedule $S_{(1)}(\mathcal{L}_n, \mathcal{M}_m)$ that is terminated by the shortest job J_n. Clearly,

$$p_n \leq \frac{1}{n} p(N) \leq \frac{1}{m} p(N). \tag{14}$$

Suppose that the terminal job J_n is completed on machine M_u, $1 \le u \le m$, and that its completion time exceeds the bound of B, i.e., $p(N_u) > s_u B$.

In schedule $S_{(1)}$, for each machine, find the value G_i such that

$$B = \frac{p(N_i) + G_i}{s_i},\ 1 \le i \le m; i \ne u. \tag{15}$$

Let us call the value G_i the *gap* on machine M_i. We can interpret the gap on a machine as the amount of processing that could be additionally assigned to that machine so that the machine completes at exactly time B. Define $G_u = 0$, i.e., there is no gap on the critical machine M_u.

Summing up the equalities (15) and the inequality $p(N_u) + G_u > s_u B$, we deduce

$$\begin{aligned} \sum_{i=1}^{m} p(N_i) + \sum_{i=1}^{m} G_i &= p(N) + \sum_{i=1}^{m} G_i > B \sum_{i=1}^{m} s_i \\ &= \left(2 - \frac{2}{m}\right) C_{\max}\left(S_p^*\left(\mathcal{L}_n, \mathcal{M}_m\right)\right) S_m. \end{aligned}$$

Since $C_{\max}\left(S_p^*\left(\mathcal{L}_n, \mathcal{M}_m\right)\right) \ge T_m = p\left(N\right)/S_m$, we deduce

$$\sum_{i=1}^{m} G_i > \left(1 - \frac{2}{m}\right) p(N). \tag{16}$$

If there exists a gap G_i such that $G_i \ge p_n$, then job J_n could have been assigned to machine M_i and it would have completed before B. Thus, for each non-zero gap G_i the inequalities $G_i < p_n \le \frac{p(N)}{n} \le \frac{p(N)}{m}$ hold.

If the preemption had been used before the last job is scheduled, i.e., if $J_\ell \ne J_n$, then the number of such non-zero gaps would have been at most $m - 2$, since the assignment of job J_ℓ leaves no gap on machine $M_{\ell'}$; see Step 2(c). It follows from (16) that

$$\left(1 - \frac{2}{m}\right) p(N) < \sum_{i=1}^{m} G_i < (m-2)\, p_n \le (m-2)\, \frac{p\left(N\right)}{m}, \tag{17}$$

a contradiction.

Therefore, no preemption has been used prior to scheduling the last job J_n, and the number of non-zero gaps is at most $m - 1$. We know that job J_n cannot be processed non-preemptively to complete by time B.

Suppose that there are two machines M' and M'' with non-zero gaps G' and G'', respectively, such that the total gap $G' + G''$ is less than p_n. Since each of at most $m - 3$ remaining non-zero gaps is less than p_n, we deduce from (16) that

$$\left(1 - \frac{2}{m}\right) p(N) < \sum_{i=1}^{m} G_i \le (m-3)\, p_n + p_n = (m-2)\, p_n,$$

leading to the same contradiction as above. Hence, in the remaining part of this proof, we only need to consider the situation that the total duration of any two non-zero gaps is at least p_n.

We show that job J_n can be processed with a single preemption, as described in Step 2(c) of the algorithm for $J_\ell = J_n$. Take a pair of machines $M_{\ell'}$ and $M_{\ell''}$ such that $G_{\ell'} + G_{\ell''} \geq p_n$. In accordance with Step 2(c), define

$$x_n = G_{\ell'},\ y_n = p_n - x_n,$$

which implies that

$$y_n + p\left(N_{\ell''}\right) = y_n + s_{\ell''}B - G_{\ell''} = p_n + s_{\ell''}B - \left(G_{\ell'} + G_{\ell''}\right) \leq s_{\ell''}B,$$

i.e., condition (9) holds, as required.

Assume that $p\left(N_{\ell''}\right) > 0$, which means that there is at least one job, say, J_i, assigned to $M_{\ell''}$ such that $p_i \geq p_n$. Compute

$$\frac{x_n}{s_{\ell'}} + \frac{y_n}{s_{\ell''}} \leq \frac{x_n + y_n}{s_{\ell''}} \leq \frac{p_i}{s_{\ell''}} \leq \frac{p\left(N_{\ell''}\right)}{s_{\ell''}} \leq B,$$

i.e., condition (10) also holds.

Now, we only need to consider the case that $p\left(N_v\right) = 0$ for all machines M_v with $v > \ell'$. Notice that for all these machines $G_v = s_v B$.

We can take the fastest machine M_1 as machine $M_{\ell'}$, since this machine has a non-zero gap prior to the assignment of job J_n; otherwise, there are at most $m-2$ non-zero gaps in schedule $S_{(1)}\left(\mathcal{L}_n, \mathcal{M}_m\right)$ and we can use (17) to derive a contradiction.

Thus, we only need to consider the situation that prior to the assignment of job J_n all machines, other than the fastest machine M_1, have no assigned jobs. We show that job J_n can be processed with a single preemption on the two fastest machines, M_1 and M_2. Let us assign the last job preemptively to machines M_1 and M_2 and according to Algorithm LPT1 we make M_1 critical. For job J_n, split its processing time into two parts x_n and y_n, where $x_n = s_1 B - p\left(N_1\right)$ and that part of J_n is assigned to M_1, while the remaining part $y_n = p_n - x_n$ is assigned to M_2.

To guarantee feasibility, we need to prove that (10) holds, i.e.,

$$\frac{x_n}{s_1} + \frac{p_n - x_n}{s_2} = \frac{s_1 B - p\left(N_1\right)}{s_1}\left(1 - \frac{s_1}{s_2}\right) + \frac{p_n}{s_2} \leq B,$$

which after multiplying by a factor of $\frac{s_2}{s_1} > 0$, simplifies to

$$\frac{p\left(N_1\right)}{s_1}\left(1 - \frac{s_2}{s_1}\right) + \frac{p_n}{s_1} - B \leq 0. \tag{18}$$

We use the fact that $p_n \leq \frac{p(N)}{m}$, so that the left-hand side of (18) satisfies

$$\frac{p\left(N_1\right)}{s_1}\left(1 - \frac{s_2}{s_1}\right) + \frac{p_n}{s_1} - B \leq \frac{p\left(N\right)}{s_1}\left(1 - \frac{s_2}{s_1}\left(1 - \frac{1}{m}\right)\right) - B.$$

Now $\rho_m^{(1)} T_m = \frac{2(m-1)}{m}\frac{p(N)}{\sum_{i=1}^m s_i} \leq B$, so that

$$\begin{aligned}\frac{p\left(N\right)}{s_1}\left(1 - \frac{s_2}{s_1}\left(1 - \frac{1}{m}\right)\right) - B \quad &\leq \quad \frac{p\left(N\right)}{s_1}\left(1 - \frac{s_2}{s_1}\left(1 - \frac{1}{m}\right)\right) \\ &\quad - \frac{2\left(m-1\right)}{m}\frac{p\left(N\right)}{\sum_{i=1}^m s_i}.\end{aligned}$$

Since s_1 and s_2 are the two fastest machine speeds, it follows that $\sum_{i=1}^{m} s_i \leq (s_1 + (m-1)\, s_2)$, so that

$$\begin{aligned}
& \frac{p(N)}{s_1}\left(1-\frac{s_2}{s_1}\left(1-\frac{1}{m}\right)\right)-B \\
\leq \; & \frac{p(N)}{s_1}\left(1-\frac{s_2}{s_1}\left(1-\frac{1}{m}\right)\right)-\frac{2(m-1)}{m}\frac{p(N)}{s_1+(m-1)\,s_2} \\
= \; & -\frac{p(N)}{ms_1^2\,(s_1+(m-1)\,s_2)}\left((m-2)\,s_1^2-(m-1)^2\,s_1s_2+(m-1)^2\,s_2^2\right).
\end{aligned}$$

To show that (18) holds, we demonstrate that

$$(m-2)\,s_1^2-(m-1)^2\,s_1s_2+(m-1)^2\,s_2^2 \geq 0. \tag{19}$$

Recall that Algorithm LPT1 assigns all jobs except job J_n to machine M_1, so that these jobs complete at time $(p(N)-p_n)/s_1$. Since $p_n \leq \frac{1}{m}p(N)$, machine M_1 completes no earlier than time $\frac{m-1}{m}\frac{p(N)}{s_1}$. On other other hand, job J_n cannot be assigned to M_2 or to any slower machine to be processed without preemption and to complete by time B, so that $\frac{p_n}{s_2} > B$. Thus, we have that

$$\frac{m-1}{m}\frac{p(N)}{s_1} \leq B < \frac{p_n}{s_2} \leq \frac{p(N)}{ms_2},$$

which implies that $s_1 \geq (m-1)\,s_2$. Then we can express the ratio of the machine speeds as $\frac{s_1}{s_2} = (m-1)\,\alpha$, where $\alpha \geq 1$. Substituting into (19), we deduce

$$\begin{aligned}
& (m-2)\,s_1^2-(m-1)^2\,s_1s_2+(m-1)^2\,s_2^2 \\
= \; & (m-2)\,(m-1)^2\,\alpha^2 s_2^2-(m-1)^3\,\alpha s_2^2+(m-1)^2\,s_2^2 \\
= \; & (m-1)^2\,s_2^2\left((m-2)\,\alpha^2-(m-1)\,\alpha+1\right) \\
= \; & (m-1)^2\,s_2^2\,((m-2)\,\alpha-1)\,(\alpha-1) \geq 0,
\end{aligned}$$

where the last inequality holds since $\alpha \geq 1$ and we are only interested in $m \geq 3$. Thus, Algorithm LPT1 creates a schedule with a single preemption in which both machines complete by time B.

Having considered all possible cases, we conclude that the minimal counterexample does not exist and $C_{\max}\left(S_{(1)}(\mathcal{L}_n,\mathcal{M}_m)\right) \leq B$ for all instances as required. ∎

To see that the established bound is tight, consider an instance of the problem with m machines, all of unit speed except machine M_1, which has speed $m-1$, and m jobs of unit length, i.e., $\mathcal{M}_m = (m-1,1,1,\ldots,1)$ and $\mathcal{L}_n = \mathcal{L}_m = (1,1,\ldots,1)$. It is clear that $C_{\max}\left(S_p^*(\mathcal{L}_m,\mathcal{M}_m)\right) = T_m = \frac{m}{2m-2}$. For this instance, an optimal preemptive schedule requires $2(m-1)$ preemptions, which is the maximum estimate [7].

In an optimal non-preemptive schedule $S_{(0)}^*(\mathcal{L}_n,\mathcal{M}_m)$ at least one of the slower machines will receive one job, so that $C_{\max}\left(S_{(0)}^*(\mathcal{L}_n,\mathcal{M}_m)\right) = 1$, and that makespan cannot be reduced if a single preemption is allowed. Indeed, suppose that there exists a schedule $S_{(1)}^*(\mathcal{L}_n,\mathcal{M}_m)$ with exactly one preemption such that $C_{\max}\left(S_{(1)}^*(\mathcal{L}_n,\mathcal{M}_m)\right) < 1$. Then a preempted job, say job J_ℓ, is shared between a slow machine M' and the fast machine M_1. Apart from the piece of job J_ℓ machine M_1 can process at most $m-2$ jobs; otherwise it would complete after time 1. Thus, a slow machine, other than machine M', must receive one job, i.e., $C_{\max}\left(S_{(1)}^*(\mathcal{L}_n,\mathcal{M}_m)\right) = 1$.

Observation 1 *Theorem 3 still holds if a looser version of Algorithm LPT1 is used. It is sufficient for the preemption to be used in Step 2(c) at any time, as long as any machine of the chosen pair is made critical by the preempted job and the preemption is feasible.*

Observation 2 *We note that if Algorithm LPT1 is modified so that it is run allowing no preemption and B is redefined as* $\rho_m^{(0)} C_{\max}\left(S_p^*\right)$*, where* $\rho_m^{(0)} = 2 - 1/m$*, then the proof of Theorem 3 can be appropriately adjusted to establish that* $\rho_m^{(0)} = 2 - 1/m$ *is the power of preemption. This is achieved by using a less restrictive algorithm than those used for the same purpose in [18] and [16].*

5 Conclusion

We have analyzed the scheduling problem on m uniform parallel machines with the objective of minimizing the makespan, provided that at most one preemption is allowed. We have shown that finding an optimal schedule for $m = 2$ can be done in polynomial time, and have established a global tight bound on how much is lost for an arbitrary m.

For the problem on m uniform parallel machines, it is an interesting research goal to deduce a bound on the ratio $C_{\max}(S^*_{(q)})/C_{\max}(S^*_p)$, provided that at most any fixed number q of preemptions is allowed, where $2 \leq q \leq 2\,(m-1)$. It is of interest to perform a parametric analysis of the quality of schedules with at most one preemption on three uniform machines, similar to a study conducted in [15]. Possible extensions may also include an objective function other than the makespan and/or more general machine environment, e.g., unrelated parallel machines.

Acknowledgement

We are grateful to the editors and anonymous referees for their suggestions aimed at improving the presentation and clarity of the paper.

References

[1] O. Braun, G. Schmidt, Parallel processor scheduling with limited number of preemptions, SIAM J. Comput. 32 (2003) 671–680.

[2] P. Brucker, Scheduling Algorithms, 5th edition, Springer, Berlin, 2007.

[3] B. Chen, Parallel machine scheduling for early completion, in J.Y-T. Leung, (Ed.), Handbook of Scheduling: Algorithms, Models and Performance Analysis, Chapman & Hall/CRC, London, 2004, pp. 9-175–9-184.

[4] J.R. Correa, M.Skutella, J. Verschae, The power of preemption on unrelated machines and applications to scheduling orders, Math. Oper. Res. 37 (2012) 379–398.

[5] L. Epstein, A. Levin, The benefit of preemption for single machine scheduling so as to minimise total weighted completion time, Oper. Res. Letters 44 (2016) 772–774.

[6] L. Epstein, A. Levin, A.J. Soper, V.A. Strusevich, Power of preemption for minimizing total completion time on uniform parallel machines, SIAM J. Discr. Math. 31 (2017) 101–123.

[7] T. F. Gonzalez, S. Sahni, Preemptive scheduling of uniform processor systems, J. ACM 25 (1978) 92–101.

[8] Y. Jiang, Z. Weng, J. Hu, Algorithms with limited number of preemptions for scheduling on parallel machines, J. Comb. Opti. 27 (2014) 711–723.

[9] E. L. Lawler, J. Labetoulle, On preemptive scheduling of unrelated parallel processors by linear programming, J. ACM 25 (1978) 612–619.

[10] C.-Y. Lee, V.A. Strusevich, Two-machine shop scheduling with an uncapacitated interstage transporter, IIE Trans. 37 (2005) 725–736.

[11] J.-H. Lin, J.S. Vitter, $\varepsilon-$approximations with minimum packing constraint violation in: STOC '92 Proceedings of the 24th Annual ACM Symposium on Theory of Computing, ACM: New York, 1992, pp. 771-782.

[12] R. McNaughton, Scheduling with deadlines and loss functions, Manag. Sci. 6 (1959) 1–12.

[13] K. Rustogi, V.A. Strusevich, Parallel machine scheduling: Impact of adding extra machines, Oper. Res. 61 (2013) 1243–1257.

[14] E. Shchepin, N. Vakhania, On the geometry, preemptions and complexity of multiprocessor and shop scheduling, Ann. Oper. Res. 159 (2008) 183-213.

[15] A.J. Soper, V.A. Strusevich, Single parameter analysis of power of preemption on two and three uniform machines, Discr. Opti. 12 (2014a) 26-46.

[16] A.J. Soper, V.A. Strusevich, Power of Preemption on Uniform Parallel Machines, in: 17th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) / 18th International Workshop on Randomization and Computation (RANDOM'14). Leibniz International Proceedings in Informatics (LIPIcs) 28, 2014b, pp. 392-402, doi:10.4230/LIPIcs.APPROX-RANDOM.2014.392.

[17] H. Shachnai, T. Tamir, G.J. Woeginger, Minimizing makespan and preemption costs on a system of uniform machines, Algorithmica 42 (2005) 309–334.

[18] G.J. Woeginger, A comment on scheduling on uniform machines under chain-like precedence constraints, Oper. Res. Letters 26 (2000) 107–109.