Power of 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.
`{A.J.Soper,V.Strusevich}@greenwich.ac.uk`

Abstract

For a scheduling problem on parallel machines, the power of preemption is defined as the ratio of the makespan of an optimal non-preemptive schedule over the makespan of an optimal preemptive schedule. For m uniform parallel machines, we give the necessary and sufficient conditions under which the global bound of $2 - 1/m$ is tight. If the makespan of the optimal preemptive schedule is defined by the ratio of the total processing times of $r < m$ longest jobs over the total speed of r fastest machines, we show that the tight bound on the power of preemption is $2 - 1/\min\{r, m - r\}$.

1998 ACM Subject Classification F.2.2 Sequencing and Scheduling, G.2.1 Combinatorial Algorithms, G.1.6.Optimization

Keywords and phrases Machine Scheduling, Uniform Parallel Machines, Power of Preemption

Digital Object Identifier 10.4230/LIPIcs.APPROX-RANDOM.2014.392

1 Introduction

In this paper, we perform an analysis of the power of preemption for scheduling problems on uniform parallel machines.

In parallel machine scheduling, we are given the jobs of the 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 all problems considered in this paper the objective is to minimize the *makespan,* i. e., the maximum completion time. For a schedule S, the makespan is denoted by $C_{\max}(S)$. 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. For an instance of a scheduling problem on parallel machines, let S^*_{np} and S^*_p denote an optimal non-preemptive and an optimal preemptive schedule, respectively.

The problem of finding an optimal non-preemptive schedule on identical parallel machines is NP-hard, and the corresponding problems on uniform or unrelated machines are obviously no easier. The preemptive counterparts of 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.

17th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'14) /
18th Int'l Workshop on Randomization and Computation (RANDOM'14).
Editors: Klaus Jansen, José Rolim, Nikhil Devanur, and Cristopher Moore; pp. 392–402
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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 define the *power of preemption* as the maximum ratio $C_{\max}(S^*_{np})/C_{\max}(S^*_p)$ across all instances of the problem at hand. We denote the power of preemption by ρ_m. The power of preemption determines what can be gained regarding the maximum completion time if preemption is allowed.

In order to determine the exact value of ρ_m for a particular problem and to give that concept some practical meaning, the following should be done:

(i) demonstrate that the inequality

$$\frac{C_{\max}\left(S^*_{np}\right)}{C_{\max}\left(S^*_p\right)} \le \rho_m \tag{1}$$

holds for all instances of the problem;

(ii) exhibit instances of the problem for which (1) holds as equality, i. e., to show that the value of ρ_m is tight; and

(iii) develop a polynomial-time algorithm that finds a heuristic non-preemptive schedule S_{np} such that

$$\frac{C_{\max}\left(S^*_{np}\right)}{C_{\max}\left(S^*_p\right)} \le \frac{C_{\max}\left(S_{np}\right)}{C_{\max}\left(S^*_p\right)} \le \rho_m. \tag{2}$$

If the machines are identical parallel, then it is known that $\rho_m = 2 - 2/(m+1)$, as independently proved in [1] and [9]. It is shown in [11], that the value of ρ_m can be reduced for some instances that contain jobs with fairly large processing times.

For unrelated parallel machines, a rounding procedure that is attributed to Shmoys and Tardos and reproduced in [10] and [4] finds non-preemptive schedules S_{np} such that the bound (2) holds for $\rho_m = 4$. This bound is tight, as proved in [4].

According to [13], for uniform parallel machines $\rho_m = 2 - 1/m$. For $m = 2$ a parametric analysis of the power of preemption with respect to the speed of the faster machine is independently performed in [7] and [12]. For $m = 3$, a similar analysis is contained in [12], provided that the machine speeds take at most two values, 1 and $s \ge 1$.

2 Preliminaries

An instance I of the problem with n jobs and m parallel uniform machines is defined by the list $\mathcal{L}_n = (p_1, p_2, \dots, p_n)$ of the processing times of the jobs and the list $\mathcal{M}_m = (s_1, s_2, \dots, s_m)$ of the machine speeds. The machines are numbered in non-increasing order of their speeds, i. e., $s_1 \ge s_2 \ge \cdots \ge s_m$.The jobs are numbered in accordance with the following truncated LPT rule, i. e., m longest jobs are numbered in non-increasing order of their processing times

$$p_1 \ge p_2 \ge \cdots \ge p_m, \tag{3}$$

while the remaining jobs, all at lest as short as p_m, are numbered arbitrary.

Feasible non-preemptive and preemptive schedules for an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ are denoted by $S_{np}(\mathcal{L}_n, \mathcal{M}_m)$ or $S_{np}(I)$, and by $S_p(\mathcal{L}_n, \mathcal{M}_m)$ or $S_p(I)$, respectively; the corresponding optimal non-preemptive and preemptive schedules are denoted by $S^*_{np}(\mathcal{L}_n, \mathcal{M}_m)$ or $S^*_{np}(I)$ and by $S^*_p(\mathcal{L}_n, \mathcal{M}_m)$ or $S^*_p(I)$, 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 [6] and requires $O(n + m \log m)$ time.

Given an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$, for each u, $1 \le u \le m$, define the total speed of the u fastest machines $S_u = \sum_{i=1}^{u} s_i$. Besides, define the set of u longest jobs $H_u = \{1, 2, \dots, u\}$, and for a set of jobs $Q \subseteq N$, define $p(Q) = \sum_{j \in Q} p_j$, where for completeness $p(\emptyset) = 0$.

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

$$C_{\max}(S_p^*(I)) = \max\{T_u | 1 \le u \le m\}, \tag{4}$$

where

$$T_u = p(H_u)/S_u,\ 1 \le u \le m-1;\ T_m = p(N)/S_m. \tag{5}$$

In our consideration, we classify the instances on m uniform machines as follows.

▶ **Definition 1.** An instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ is said to belong to Class r, $1 \le r \le m$, if $C_{\max}\left(S_p^*(I)\right) = T_r = \max\{T_u | 1 \le u \le m\}$.

Notice that an instance may belong to several classes simultaneously, if there is a tie for the maximum value of T_u, $1 \le u \le m$.

A non-preemptive schedule $S_{np}(I)$ is defined by a partition of set N into m subsets N_1, $N_2, \dots, N_m$, where the jobs of set N_i and only those are assigned to be processed on machine M_i, $1 \le i \le m$. Notice that even in an optimal schedule some of these subsets can be empty.

A popular heuristic for finding a non-preemptive schedule on uniform parallel machines is known as the LPT List Scheduling. According to this algorithm, the jobs are scanned in accordance with the LPT rule, i. e., in non-increasing order of their processing times, the next job is assigned to the machine where it will complete as early as possible. For an instance I on uniform machines, let the LPT algorithm output a schedule $S(I)$. It can be found in $O(nm + n \log n)$ time. The best known results on the worst-case ratio $\rho_{LPT} = C_{\max}(S(I))/C_{\max}\left(S_{np}^*(I)\right)$ are due to Kovacs [8] who proves $1.54 \le \rho_{LPT} \le 1.577$. It is proved in [13] that $C_{\max}(S(I))/C_{\max}\left(S_p^*(I)\right) \le 2 - 1/m$, and this bound is tight. For a preemptive schedule $S_p(I)$ found by a preemptive modification of the LPT algorithm the inequality $C_{\max}(S_p(I))/C_{\max}\left(S_p^*(I)\right) \le 2 - 2/(m+1)$ holds; see [5].

In the subsequent sections, we only consider instances in which the number of jobs is no smaller than the number of machines. Take an instance $(\mathcal{L}_n, \mathcal{M}_m)$ 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.

It is clear that in each schedule $S_{np}^*(\mathcal{L}_n, \mathcal{M}_m)$ and $S_p^*(\mathcal{L}_n, \mathcal{M}_m)$ the jobs are assigned to at most n fastest machines. Thus, in the non-preemptive case, $S_{np}^*(\mathcal{L}_n, \mathcal{M}_m) = S_{np}^*(\mathcal{L}_n, \mathcal{M}_n)$ and $C_{\max}\left(S_{np}^*(\mathcal{L}_n, \mathcal{M}_m)\right) = C_{\max}\left(S_{np}^*(\mathcal{L}_n, \mathcal{M}_n)\right)$, while in the preemptive case $C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_m)\right) = \max\{T_u | 1 \le u \le n < m\} = C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_n)\right)$.

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, for the purpose of studying an upper bound on it we only need to consider instances in which there are at least as many jobs as machines.

We focus on a slightly modified version of the LPT algorithm, which can be stated as follows.

Algorithm LPTm

Step 1. If required, renumber the jobs so that the m longest jobs are numbered in accordance with (3), while the other jobs are numbered arbitrarily.

Step 2. At any time that a machine becomes available, take the first job in the current list $\mathcal{L}_n$ and assign it to the machine on which it will complete as early as possible. Remove the assigned job from the list.

Step 3. Repeat Step 2 until all jobs are assigned.

Compared to the full version of the LPT algorithm, the modified Algorithm LPTm requires only $O(m \log m + nm)$ time, since finding and sorting m longest jobs takes $O(m \log m)$ time. From now on, a schedule created by Algorithm LPTm for an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ will be called $S_{LPT}(I)$.

3 Upper Bounds on the Power of Preemption

In this section, we analyze the performance of Algorithm LPTm from the point of view of the power of preemption.

▶ **Definition 2.** 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 $J_h, 1 \leq h \leq n$, on machine M_k, $1 \leq k \leq m$. We call job J_h the *terminal* job and machine M_k the *critical* machine.

The main result of this section is the following statement.

▶ **Theorem 3.** *Given an arbitrary instance $I = (\mathcal{L}_n, \mathcal{M}_m)$, where $n \geq m$, let $S_{LPT}(I)$ be a schedule created by Algorithm LPTm. Then*

$$\frac{C_{\max}(S_{LPT}(I))}{C_{\max}\left(S_p^*(I)\right)} \leq 2 - \frac{1}{m}. \tag{6}$$

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 $(\mathcal{L}_n, \mathcal{M}_m)$, which we call the minimal counterexample, such that

$$\frac{C_{\max}(S_{LPT}(\mathcal{L}_n, \mathcal{M}_m))}{C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_m)\right)} > 2 - \frac{1}{m} \tag{7}$$

and no job can be removed from the instance without violating the inequality (7).

Suppose that in schedule $S_{LPT}(\mathcal{L}_n, \mathcal{M}_m)$ job J_h is the terminal job and machine M_k the critical machine. If $h < n$ then Algorithm LPTm assigns some jobs J_j with $j > h$ after job J_h and they complete 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

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

so that

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

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 LPTm finds a schedule $S_{LPT}(\mathcal{L}_n, \mathcal{M}_m)$ that is terminated by job J_n. Since $n \geq m$, it follows that

$$p_n \leq \frac{1}{m} p(N). \tag{8}$$

For schedule $S_{LPT}(\mathcal{L}_n, \mathcal{M}_m)$, let N_i denote the set of jobs assigned to machine M_i, $1 \leq i \leq m$. For each machine, find the value G_i such that

$$C_{\max}(S_{LPT}(\mathcal{L}_n, \mathcal{M}_m)) = \frac{p(N_i) + G_i}{s_i}, \ 1 \leq i \leq m. \tag{9}$$

Let us call the value G_i the *gap* on machine M_i. We can interpret the gap on some machine as the amount of processing that could be additionally assigned to that machine so that the machine completes at exactly time $C_{\max}(S_{LPT}(\mathcal{L}_n, \mathcal{M}_m))$. Clearly, $G_k = 0$, i. e., there is no gap on the critical machine M_k. Besides, we must have that

$$p_n \geq \max\{G_i | 1 \leq i \leq m, \ i \neq k\}. \tag{10}$$

If the latter inequality had not been true, then Algorithm LPTm would have assigned job J_n to another machine, producing a schedule with a smaller makespan than $C_{\max}(S_{LPT}(\mathcal{L}_n, \mathcal{M}_m))$.

Summing up the equalities (9) we deduce

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

where the last inequality is due to (7). Since $C_{\max}\left(S_p^*(\mathcal{L}_n, \mathcal{M}_m)\right) \geq T_m = p(N)/S_m$, we deduce $\sum_{i=1}^{m} G_i > \left(1 - \frac{1}{m}\right) p(N)$. On the critical machine the gap is equal to zero, therefore the largest gap on the remaining machines is at least $\frac{1}{m-1} \sum_{i=1}^{m} G_i$. This and (10) yield

$$p_n \geq \frac{1}{m-1} \sum_{i=1}^{m} G_i > \frac{p(N)}{m},$$

which contradicts (8). Thus, the minimal counterexample does not exist and (6) holds. ◀

Notice that Theorem 3 holds for all instances, irrespective of their class. However, below we show that the established upper bound can be reduced for instances $I = (\mathcal{L}_n, \mathcal{M}_m)$ that are known to belong to Class r, $1 \leq r \leq m-1$. If r is not unique, we select the value that is the closest to $m/2$.

For r, $1 \leq r \leq m-1$, define the lists $\mathcal{L}'_r$ and $\mathcal{M}'_r$ obtained from the lists $\mathcal{L}_n$ and $\mathcal{M}_m$ by the removal of the r longest jobs and the r fastest machines, respectively. In other words, $\mathcal{L}'_r = (p_{r+1}, \ldots, p_n)$ and $\mathcal{M}'_r = (s_{r+1}, \ldots, s_m)$. The following algorithm for creating a non-preemptive schedule for an instance I of Class r applies Algorithm LPTm to two instances, $(\mathcal{L}_r, \mathcal{M}_r)$ and $(\mathcal{L}'_r, \mathcal{M}'_r)$.

Algorithm LPTr

Step 1. Given an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ of Class r, $1 \leq r \leq m-1$, split I into two instances $(\mathcal{L}_r, \mathcal{M}_r)$ and $(\mathcal{L}'_r, \mathcal{M}'_r)$.

Step 2. Run Algorithm LPTm twice to find a schedule $S_{LPT}(\mathcal{L}_r, \mathcal{M}_r)$ and a schedule $S_{LPT}(\mathcal{L}'_r, \mathcal{M}'_r)$.

Step 3. Output schedule $S_{LPT(r)}(\mathcal{L}_n, \mathcal{M}_m)$ obtained by combining the schedules $S_{LPT}(\mathcal{L}_r, \mathcal{M}_r)$ and $S_{LPT}(\mathcal{L}'_r, \mathcal{M}'_r)$.

The algorithm requires $O(m\log m + nm)$ time. For its analysis, define

$$T'_r = \frac{\sum_{j=r+1}^{n} p_j}{\sum_{i=r+1}^{m} s_i}.$$

▶ **Lemma 4.** *For an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ of Class r, $1 \le r \le m-1$, the inequality*

$$T_r \ge T'_r \tag{11}$$

holds.

Proof. Since for an For an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ of Class r by definition the inequality $T_r \ge T_m$ holds, we deduce that

$$\begin{aligned}
0 \quad &\le \quad T_r - T_m = \frac{\sum_{j=1}^{r} p_j}{\sum_{i=1}^{r} s_i} - \frac{\sum_{j=1}^{n} p_j}{\sum_{i=1}^{m} s_i} = \frac{\sum_{j=1}^{r} p_j \sum_{i=1}^{m} s_i - \sum_{j=1}^{n} p_j \sum_{i=1}^{r} s_i}{\sum_{i=1}^{r} s_i \sum_{i=1}^{m} s_i} \\
&= \frac{\sum_{j=1}^{r} p_j \left(\sum_{i=1}^{m} s_i - \sum_{i=1}^{r} s_i\right) - \sum_{j=r+1}^{n} p_j \sum_{i=1}^{r} s_i}{\sum_{i=1}^{r} s_i \sum_{i=1}^{m} s_i} = \frac{\sum_{j=1}^{r} p_j \sum_{i=r+1}^{m} s_i - \sum_{j=r+1}^{n} p_j \sum_{i=1}^{r} s_i}{\sum_{i=1}^{r} s_i \sum_{i=1}^{m} s_i},
\end{aligned}$$

which implies that (11) holds. ◀

▶ **Theorem 5.** *Given an arbitrary instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ of Class r, where $n \ge m$ and $1 \le r \le m-1$, let $S_{LPT(r)}(I)$ be a schedule created by Algorithm LPTr. Then*

$$\rho_m = \frac{C_{\max}\left(S^*_{np}(I)\right)}{C_{\max}\left(S^*_p(I)\right)} \le \frac{C_{\max}\left(S_{LPT(r)}(I)\right)}{C_{\max}\left(S^*_p(I)\right)} \le \max\left\{2 - \frac{1}{r}, 2 - \frac{1}{m-r}\right\}. \tag{12}$$

Proof. Applying Theorem 3 to instances $(\mathcal{L}_r, \mathcal{M}_r)$ and $(\mathcal{L}'_r, \mathcal{M}'_r)$, we obtain

$$\begin{aligned}
\frac{C_{\max}\left(S^*_{np}(\mathcal{L}_r, \mathcal{M}_r)\right)}{C_{\max}\left(S^*_p(\mathcal{L}_r, \mathcal{M}_r)\right)} &\le \frac{C_{\max}\left(S_{LPT}(\mathcal{L}_r, \mathcal{M}_r)\right)}{C_{\max}\left(S^*_p(\mathcal{L}_r, \mathcal{M}_r)\right)} = \frac{C_{\max}\left(S_{LPT}(\mathcal{L}_r, \mathcal{M}_r)\right)}{T_r} \le 2 - \frac{1}{r}; \\
\frac{C_{\max}\left(S^*_{np}(\mathcal{L}'_r, \mathcal{M}'_r)\right)}{C_{\max}\left(S^*_p(\mathcal{L}'_r, \mathcal{M}'_r)\right)} &\le \frac{C_{\max}\left(S_{LPT}(\mathcal{L}'_r, \mathcal{M}'_r)\right)}{C_{\max}\left(S^*_p(\mathcal{L}'_r, \mathcal{M}'_r)\right)} \le \frac{C_{\max}\left(S_{LPT}(\mathcal{L}'_r, \mathcal{M}'_r)\right)}{T'_r} \le 2 - \frac{1}{m-r}.
\end{aligned}$$

Due to (11)

$$\begin{aligned}
\frac{C_{\max}\left(S^*_{np}(\mathcal{L}_n, \mathcal{M}_m)\right)}{C_{\max}\left(S^*_p(\mathcal{L}_n, \mathcal{M}_m)\right)} &\le \frac{C_{\max}\left(S_{LPT(r)}(\mathcal{L}_n, \mathcal{M}_m)\right)}{T_r} \\
&= \frac{\max\left\{C_{\max}\left(S_{LPT}(\mathcal{L}_r, \mathcal{M}_r)\right), C_{\max}\left(S_{LPT}(\mathcal{L}'_r, \mathcal{M}'_r)\right)\right\}}{T_r} \\
&\le \max\left\{\frac{C_{\max}\left(S_{LPT}(\mathcal{L}_r, \mathcal{M}_r)\right)}{T_r}, \frac{C_{\max}\left(S_{LPT}(\mathcal{L}'_r, \mathcal{M}'_r)\right)}{T'_r}\right\} \\
&\le \max\left\{2 - \frac{1}{r}, 2 - \frac{1}{m-r}\right\},
\end{aligned}$$

as required. ◀

4 Proofs of Tightness

In this section, we prove that the established bounds on the power of preemption are tight.

4.1 Class m Instances

We start with instances of Class m. A tight instance I of this class satisfies the equality

$$\rho_m = \frac{C_{\max}\left(S^*_{np}(I)\right)}{C_{\max}\left(S^*_p(I)\right)} = 2 - \frac{1}{m}. \tag{13}$$

We exhibit the instances for which (13) holds; moreover, we describe the necessary and sufficient conditions for an instance of Class m to be tight. Let us introduce a special class of instances of the problem that plays a crucial role in establishing tightness of the bounds on the power of preemption.

▶ **Definition 6.** For the problem with m uniform machines, an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ is called *canonical* if for each machine M_k there exists an optimal non-preemptive schedule such that M_k is the only critical machine.

Under the usual assumption that $n \geq m$, let $\mathcal{I}$ be a set of instances $I = (\mathcal{L}_n, \mathcal{M}_m)$ such that

- The processing times satisfy $p_j = p,\ j \in N$;
- The speeds are positive integers that for a positive W satisfy

$$s_1 \geq s_2 \geq \cdots \geq s_m;\ 1 \leq W s_i \leq m;\ \sum_{i=1}^{m} s_i = \frac{n+m-1}{W}.$$

▶ **Lemma 7.** *For any instance $I = (\mathcal{L}_n, \mathcal{M}_m) \in \mathcal{I}$ the equality*

$$\frac{C_{\max}\left(S^*_{np}(I)\right)}{C_{\max}(S^*_p(I))} = 1 + \frac{m-1}{n}$$

holds.

Proof. For an optimal non-preemptive schedule $S^*_{np}(I)$, let n_i denote the number of jobs assigned to machine M_i. If $n_i \leq W s_i - 1$, $1 \leq i \leq m$, then we derive a contradiction:

$$n = \sum_{i=1}^{m} n_i \leq W \sum_{i=1}^{m} s_i - m = (n+m-1) - m = n-1.$$

Thus, in $S^*_{np}(I)$ at least one machine should get $n_i \geq W s_i$ jobs, i. e., $C_{\max}\left(S^*_{np}(I)\right) \geq Wp$. The smallest value of the makespan is achieved if for an arbitrary k, $1 \leq k \leq m$, machine M_k gets exactly $n_k = W s_k$ jobs, so that $C_{\max}\left(S^*_{np}(I)\right) = Wp$, which is the completion time of the last job assigned to machine M_k. To make sure that all other machines complete earlier than time p, assign exactly $n_i = W s_i - 1$ jobs to machine M_i, $1 \leq i \leq m,\ i \neq k$. This allocation is feasible, i. e., all n jobs are distributed, since

$$n = \sum_{i=1}^{m} n_i = W \sum_{i=1}^{m} s_i - (m-1) = n.$$

Thus, we derive that for any instance $I \in \mathcal{I}$ the equality

$$\frac{C_{\max}\left(S^*_{np}(I)\right)}{C_{\max}(S^*_p(I))} = \frac{Wp}{\frac{Wpn}{n+m-1}} = 1 + \frac{m-1}{n},$$

holds, i. e., I is a tight instance. ◀

The lemma below states that set $\mathcal{I}$ consists of instances of Class m.

▶ **Lemma 8.** *Any instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ such that $p_1 = \cdots = p_n = p$ belongs to Class m.*

Proof. For any u, $1 \le u \le m-2$, we have that $T_u = up/S_u$, so that

$$T_u - T_{u+1} = \frac{up}{S_u} - \frac{(u+1)\,p}{S_u + s_{u+1}} = \frac{us_{u+1} - S_u}{S_u\,(S_u + s_{u+1})}p.$$

Since

$$S_u = \sum_{i=1}^{u} s_i \ge us_u \ge us_{u+1},$$

we deduce that the sequence $T_1, T_2, \dots T_{m-1}$ is non-decreasing. Besides,

$$\begin{aligned} T_{m-1} - T_m &= \frac{(m-1)\,p}{S_{m-1}} - \frac{np}{S_{m-1} + s_m} \\ &= \frac{(m-1)\,s_m - (n+1-m)\,S_{m-1}}{S_{m-1}\,(S_{m-1} + s_m)}p \le 0. \end{aligned}$$

This proves the lemma. ◀

Under the assumption that $n \ge m$, the value $1 + (m-1)/n$ reaches its maximum of $2 - 1/m$ if $n = m$. Combining Theorem 3, Lemma 7 and Lemma 8, we derive the following statement.

▶ **Corollary 9.** *For instances of Class m the power of preemption is $2 - \frac{1}{m}$, and this value cannot be reduced for instances of this class.*

As far as the set $\mathcal{I}$ is concerned, a stronger statement can be proved.

▶ **Theorem 10.** *For an instance $I = (\mathcal{L}_n, \mathcal{M}_m)$ of Class m to be tight, it is necessary and sufficient that I is an instance of set $\mathcal{I}$ with $n = m$.*

Proof. Sufficiency of the theorem immediately follows from Lemma 7. To prove necessity, first notice that it follows from the tightness of instance I that is does not belong to Class r for any r, $1 \le r \le m-1$. Due to Theorem 3 we have that

$$2 - \frac{1}{m} = \frac{C_{\max}\left(S^*_{np}(I)\right)}{C_{\max}\left(S^*_p(I)\right)} \le \frac{C_{\max}\left(S_{LPT}(I)\right)}{C_{\max}\left(S^*_p(I)\right)} \le 2 - \frac{1}{m}. \tag{14}$$

This implies that in (14) all inequalities hold as equalities, i. e., for a tight instance I Algorithm LPTm in fact finds an optimal non-preemptive schedule. In the remainder of this proof we can deal with schedule $S_{LPT}(I)$ instead of schedule $S^*_{np}(I)$.

If in schedule $S_{LPT}(I)$ some job J_h with $h < n$ is terminal, then the jobs $h+1, \dots, n$ can be removed from the instance. Since I is a Class m instance and does not belong to any other class, the removal of the jobs reduces the makespan of the optimal preemptive schedule, i. e., for the modified instance I', $C_{\max}\left(S^*_p(I')\right) < C_{\max}\left(S^*_p(I)\right)$. On the other hand we have $C_{\max}\left(S_{LPT}(I')\right) = C_{\max}\left(S_{LPT}(I)\right)$, so that

$$\frac{C_{\max}\left(S_{LPT}(I')\right)}{C_{\max}\left(S^*_p(I')\right)} > \frac{C_{\max}\left(S_{LPT}(I)\right)}{C_{\max}\left(S^*_p(I)\right)} = \frac{C_{\max}\left(S^*_{np}(I)\right)}{C_{\max}\left(S^*_p(I)\right)} = 2 - \frac{1}{m}.$$

However, this implies Theorem 3 does not hold for instance I'. Thus, in what follows we assume that in $S_{LPT}(I)$ the terminal job is job J_n and hence unique. For job J_n (8) holds due to $n \ge m$.

Similarly to the proof of Theorem 3, for schedule $S_{LPT}(I)$ let G_i be the gap on machine M_i that is defined by (9). The gap analysis of schedule $S_{LPT}(I)$ leads to

$$p(N) + \sum_{i=1}^{m} G_i = C_{\max}\left(S_{LPT}\left(I\right)\right) \sum_{i=1}^{m} s_i = \left(2 - \frac{1}{m}\right) C_{\max}\left(S_p^*\left(I\right)\right) S_m.$$

Since I is a Class m instance, we deduce from $C_{\max}\left(S_p^*\left(I\right)\right) = p\left(N\right)/S_m$ that

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

Since in schedule $S_{LPT}\left(I\right)$ the gap on the critical machine is zero, it follows that the largest gap $G_{\max} = \max\left\{G_i | 1 \le i \le m\right\}$ is no smaller than $p\left(N\right)/m$. On the other hand, $G_{\max}$ does not exceed p_n; otherwise Algorithm LPTm would have assigned job J_n to the machine with the largest gap. Combining this with (8), we obtain

$$\frac{p\left(N\right)}{m} \le G_{\max} \le p_n \le \frac{p\left(N\right)}{m}.$$

It follows that for $n \ge m$ in the expression above all inequalities hold as strict equalities, and we deduce that

- $m = n$, i. e., in a tight instance I, the number of jobs is equal to the number of machines.
- in a tight instance I all processing times are equal, i. e., $p_j = p,\ j \in N$, where $p = p(N)/n$.
- the largest gap $G_{\max}$ is equal to p.

Lemma 8 confirms that a tight instance I belongs to Class m, i. e., $C_{\max}\left(S_p^*\left(I\right)\right) = mp/S_m$. The total gap on all machines is equal to

$$\sum_{i=1}^{m} G_i = \left(m-1\right) p. \tag{15}$$

In schedule $S_{LPT}\left(I\right)$ the terminal job is unique, i. e., there are $m-1$ non-critical machines, each with a non-zero gap. Since the largest gap is p, it follows from (15) that in schedule $S_{LPT}\left(I\right)$ at the time of assigning the last job J_n the gaps on all machines are the same and equal to p. This means that any machine can be made critical, while the remaining machines will complete earlier. In other words, I is a canonical instance, and for each machine M_i, there exists an optimal non-preemptive schedule in which machine M_i is critical.

For i, $1 \le i \le m$, let k_i denote the number of jobs on M_i in an optimal schedule in which machine M_i is critical, i. e., $C_{\max}\left(S_{np}^*\left(I\right)\right) = k_i p/s_i,\ i = 1, \ldots, m$. We deduce

$$C_{\max}\left(S_{np}^*\left(I\right)\right) \sum_{i=1}^{m} s_i = p \sum_{i=1}^{m} k_i.$$

On the other hand, since I is a Class m instance, the equalities $C_{\max}\left(S_p^*\left(I\right)\right) = p\left(N\right)/\sum_{i=1}^{m} s_i = mp/\sum_{i=1}^{m} s_i$ hold, and we derive

$$C_{\max}\left(S_{np}^*\left(I\right)\right) \sum_{i=1}^{m} s_i = \left(2 - \frac{1}{m}\right) C_{\max}\left(S_p^*\left(I\right)\right) \sum_{i=1}^{m} s_i = \left(2 - \frac{1}{m}\right) pm.$$

This yields $\sum_{i=1}^{m} k_i = 2m-1$. Notice that all ratios k_i/s_i, $1 \le i \le m$, are equal. Let W be the value $W = \frac{k_1}{s_1} = \frac{k_2}{s_2} = \ldots \frac{k_m}{s_m}$. Then

$$W \sum_{i=1}^{m} s_i = 2m-1.$$

and we conclude that I is an instance of set $\mathcal{I}$ with $n = m$. ◀

4.2 Instances of Other Classes

We now demonstrate that for the instances of Class r, $1 \le r \le m-1$, the bounds on the power of preemption established in Theorem 5 are tight. Our consideration is split into two cases that depend on the sign of the difference $2r - m$.

▶ **Lemma 11.** *For $n \ge m$, and r such that $1 \le r \le m-1$ and $2r \ge m$, there exists an instance $(\mathcal{L}_n, \mathcal{M}_m)$ of Class r such that*

$$\frac{C_{\max}\left(S^*_{np}\left(\mathcal{L}_n, \mathcal{M}_m\right)\right)}{C_{\max}\left(S^*_{p}\left(\mathcal{L}_n, \mathcal{M}_m\right)\right)} = 2 - \frac{1}{r}. \tag{16}$$

Proof. For a given m, take an arbitrary r, such that $1 \le r \le m-1$ and $2r \ge m$. To prove the lemma we exhibit an instance $I = (\mathcal{L}_m, \mathcal{M}_m)$ of Class r with m machines and $n = m$ jobs. The $r-1$ faster machines each have speed 2, while all remaining machines have unit speed, i. e., $s_i = 2,\ 1 \le i \le r-1$, and $s_i = 1,\ r \le i \le m$. The processing times are defined by $p_j = 1,\ 1 \le j \le r$, and $p_j = \frac{r}{2r-1} < 1,\ r+1 \le j \le m$. We have that

$$T_i = \frac{1}{2},\ 1 \le i \le r-1;\ T_r = \frac{r}{2r-1} > \frac{1}{2};\ T_i = \frac{r + (i-r)\,p_i}{2r - 1 + (i-r)} = \frac{r}{2r-1},\ r+1 \le i \le m.$$

Here, $T_r = T_{r+1} = \cdots = T_m$ and r is the index that is closest to $m/2$ due to $r \ge m/2$. Thus, instance I belongs to Class r and and $C_{\max}\left(S^*_p\left(I\right)\right) = r/\left(2r-1\right)$.

On the other hand, it can be verified that $C_{\max}\left(S^*_{np}\left(I\right)\right) = 1$. Indeed, had $C_{\max}\left(S^*_{np}\left(I\right)\right)$ been strictly less than 1 then each of the faster machines of speed 2 should have processed exactly one job of unit duration, and therefore the remaining job of unit duration would have been assigned to a machine of unit speed, a contradiction. Thus, $C_{\max}\left(S^*_{np}\left(I\right)\right)/C_{\max}\left(S^*_p\left(I\right)\right) = 2 - 1/r$, so that (16) holds. ◀

▶ **Lemma 12.** *For $n \ge m$, and r such that $1 \le r \le m-1$ and $2r < m$, there exists an instance $(\mathcal{L}_n, \mathcal{M}_m)$ of Class r such that*

$$\frac{C_{\max}\left(S^*_{np}\left(\mathcal{L}_n, \mathcal{M}_m\right)\right)}{C_{\max}\left(S^*_{p}\left(\mathcal{L}_n, \mathcal{M}_m\right)\right)} = 2 - \frac{1}{m-r}. \tag{17}$$

Proof. For a given m, take an arbitrary r, such that $1 \le r \le m-1$ and $2r < m$. To prove the lemma we exhibit an instance $I = (\mathcal{L}_m, \mathcal{M}_m)$ of Class r with m machines and $n = m$ jobs. The speeds of all machines are equal to 2, except machine M_m, which has unit speed, i. e., $s_i = 2,\ 1 \le i \le m-1$, and $s_m = 1$. Compute

$$Q = \frac{m-r}{2\left(m-r\right)-1}$$

and define the processing times as $p_j = 2Q > 1,\ 1 \le j \le r$, and $p_j = 1,\ r+1 \le j \le m$. We have that

$$T_i = Q,\ 1 \le i \le r;\ T_i = \frac{2rQ + (i-r)}{2r + 2\left(i-r\right)} < Q,\ r+1 \le i < m;\ T_m = \frac{2rQ + (m-r)}{2m-1} = Q.$$

Here, $T_1 = \cdots = T_r$ and r is the index that is closest to $m/2$ due to $r < m/2$. Thus, instance I belongs to Class r and and $C_{\max}\left(S^*_p\left(I\right)\right) = Q$.

In an optimal non-preemptive schedule a longer job of duration $2Q$ and any other job cannot be completed before time 1 on any machine, since $2Q + 1 > 2$. Thus, in any optimal schedule there are r faster machines of speed 2 each processing exactly one longer job of duration $2Q$ and completing at time Q. If the slow machine M_m is assigned a job, then it completes it at time 1; otherwise, there exists a faster machine of speed 2 that processes at least two shorter jobs. Thus, $C_{\max}\left(S_{np}^{*}(I)\right)=1$, and $C_{\max}\left(S_{np}^{*}(I)\right)/C_{\max}\left(S_{p}^{*}(I)\right)=2-1/(m-r)$, so that (17) holds. ◀

Thus, for instances of Class r the bound $2-\min\{1/r,1/(m-r)\}$ on the power of preemption is tight.

References

1 O. Braun and G. Schmidt. Parallel processor scheduling with limited number of preemptions. *SIAM Journal on Computing*, 32:671–680, 2003.

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, pages 9-175–9-184, 2004.

4 J. R. Correa, M. Skutella and J. Verschae. The power of preemption on unrelated machines and applications to scheduling orders. *Mathematics of Operations Research*, 37:379–398, 2012.

5 T. Ebenlendr and J. Sgall. Optimal and online preemptive scheduling on uniformly related machines. *Journal of Scheduling*, 12:517–527, 2009.

6 T. F. Gonzalez and S. Sahni. Preemptive scheduling of uniform processor systems. *Journal of ACM*, 25:92–101, 1978.

7 Y. Jiang, Z. Weng and J. Hu. Algorithms with limited number of preemptions for scheduling on parallel machines. *Journal of Combinatorial Optimization*, 27:711–723, 2014.

8 A. Kovács. New approximation bounds for LPT scheduling. *Algorithmica*, 57:413–433, 2010.

9 C.-Y. Lee and V. A. Strusevich. Two-machine shop scheduling with an uncapacitated interstage transporter. *IIE Transactions*, 37:725–736, 2005.

10 J.-H. Lin and 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, pages 771–782, 1992.

11 K. Rustogi and V. A. Strusevich. Parallel machine scheduling: Impact of adding extra machines. *Operations Research,* 61:1243–1257, 2013.

12 A. J. Soper and V. A. Strusevich. Single parameter analysis of power of preemption on two and three uniform machines. *Discrete Optimization,* 12:26–46, 2014.

13 G. J. Woeginger. A comment on scheduling on uniform machines under chain-like precedence constraints. *Operations Research Letters*, 26:107–109, 2000.