SIAM J. DISCRETE MATH.
Vol. 31, No. 1, pp. 101–123

POWER OF PREEMPTION FOR MINIMIZING TOTAL COMPLETION TIME ON UNIFORM PARALLEL MACHINES*

LEAH EPSTEIN†, ASAF LEVIN‡, ALAN J. SOPER§, AND VITALY A. STRUSEVICH§

Abstract. For scheduling problems on parallel machines, the power of preemption is defined as the supremum ratio of the cost of an optimal nonpreemptive schedule over the cost of an optimal preemptive schedule (for the same input), where the cost is defined by a fixed common cost function. We present a tight analysis of the power of preemption for the problem of minimizing the total completion time on $m \geq 2$ uniformly related machines, showing that its value for $m = 2$ is equal to 1.2, and its overall value is approximately 1.39795.

Key words. scheduling, total completion time, power of preemption, uniformly related machines

AMS subject classifications. 90B35, 68R99, 68W40

DOI. 10.1137/16M1066610

1. Introduction.

In parallel machine scheduling, we are given the jobs of a set $N = \{J_1, J_2, \ldots, J_n\}$, all available at time zero (where J_j is also called job j), and $m \geq 2$ parallel machines M_1, $M_2, \ldots, M_m$ (where M_i is also called machine i). If a job $J_j \in N$ is processed on machine M_i completely, then its processing time is defined 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) *uniformly related* (or 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. For the models on identical and uniform machines, the value p_j is referred to as the *size* of job $J_j \in N$.

In a nonpreemptive 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 that or on any other machine, provided that the job is not processed on two or more machines at the same time.

Given a scheduling problem, let S be a feasible schedule and let $C_j(S)$ denote the completion time of job J_j in schedule S. Among the most popular objective functions studied in scheduling literature are the *total completion time* $\sum C_j(S)$, i.e., the sum of completion times, and the *makespan* $C_{\max}(S) = \max\{C_j(S) \,|\, J_j \in N\}$, i.e., the maximum completion time.

Given a schedule S, denote by $\Phi(S)$ the value of the objective function $\Phi \in \{\sum C_j, C_{\max}\}$ computed for S and refer to it as the *cost* of S. For a scheduling problem to minimize an objective Φ on m parallel machines (identical, uniformly related, or unrelated), let S^*_{np} and S^*_p denote, respectively, an optimal nonpreemptive and an optimal preemptive schedule with respect to function Φ.

*Received by the editors March 18, 2016; accepted for publication (in revised form) October 20, 2016; published electronically January 5, 2017.
http://www.siam.org/journals/sidma/31-1/M106661.html

†Department of Mathematics, University of Haifa, Haifa, 3498838, Israel (lea@math.haifa.ac.il).
‡Faculty of Industrial Engineering and Management, The Technion, Haifa, 32000, Israel (levinas@ie.technion.ac.il).
§Department of Mathematical Sciences, University of Greenwich, London, SE10 9LS, UK (A.J.Soper@greenwich.ac.uk, V.Strusevich@greenwich.ac.uk).

Given a problem instance of minimizing an objective Φ, denote $R = \Phi(S^*_{np})/\Phi(S^*_p)$ and call this value the *cost ratio* (for that instance). Define the *power of preemption* as the supremum ratio $R = \Phi(S^*_{np})/\Phi(S^*_p)$ across all instances of the problem at hand. We denote the power of preemption by ρ. If the number of machines is constrained to be at most m, then we denote the power of preemption restricted to such family of instances by ρ_m, and thus $\rho = \sup_{m\geq 2} \rho_m$. Since by definition $\rho_{m+1} \geq \rho_m$ holds for any $m \geq 2$, it follows that $\rho = \lim_{m\to\infty} \rho_m$.

The main interest in studying the power of preemption is that this value determines what can be gained if preemption is allowed. Most preemptive models assume that preemption is free, but in real life every preemption slows the system, in general, and increases the processing time of preempted jobs. Thus, when the power of preemption is small, it is beneficial to use nonpreemptive schedules instead of preemptive ones. From a purely theoretical point of view, this is a clean, basic, and natural combinatorial optimization problem; it does not depend on models of complexity, and it provides an interesting comparison between well-known related scheduling problems.

The purpose of this paper is to establish the value of the power of preemption for the scheduling problem of minimizing total completion time on m uniformly related machines. Specifically, we show that for this problem $\rho \approx 1.39795$ and $\rho_2 = 1.2$.

The remainder of this paper is organized as follows. Section 2 presents an overview of known results on the power of preemption for various models on parallel machines. For the model of minimizing total completion time on uniformly related machines, the well-known algorithms for finding optimal nonpreemptive and preemptive schedules are described in section 3. In section 4, we explain how to transform a given instance of the problem under consideration into an instance that possesses required properties, without decreasing the cost ratio. The value ρ of the power of preemption for the model with uniform related machines is derived in section 5, and a sequence of instances is exhibited for which the cost ratio tends to the established value of ρ. The case of two uniform machines is addressed in section 6, where it is shown that the power of preemption is exactly 1.2.

2. Power of preemption: A review. In order to determine the exact value of ρ for a particular problem of minimizing a given objective the following should be done:

(i) demonstrate that the inequality

$$\frac{\Phi\left(S^*_{np}\right)}{\Phi\left(S^*_p\right)} \leq \rho \tag{1}$$

holds for all instances of the problem;

(ii) exhibit instances of the problem for which (1) holds as equality (possibly in the supremum), i.e., show that the value of ρ is *tight*.

Most of the known results on the power of preemption have been established for the problem of minimizing the makespan.

If preemption is not allowed, the problem of minimizing the makespan is NP-hard, even on two identical parallel machines. By contrast, finding a preemptive schedule that minimizes the makespan can be done in polynomial time, even in the most general settings with unrelated machines. See a focused survey [5] on parallel machine scheduling with the makespan objective for details and references. Thus, in order to give the concept of the power of preemption practical meaning, for the problem of minimizing the objective $\Phi = C_{\max}$, the studies on the power of preemption are normally accompanied by an additional point:

(iii) develop a polynomial-time algorithm that finds a nonpreemptive schedule S_{np} such that the inequalities

$$\frac{\Phi\left(S_{np}^{*}\right)}{\Phi\left(S_{p}^{*}\right)} \leq \frac{\Phi\left(S_{np}\right)}{\Phi\left(S_{p}^{*}\right)} \leq \rho \tag{2}$$

hold for all instances, i.e., the ratio between the cost of a heuristic schedule S_{np} and the cost of an optimal preemptive schedule does not exceed the upper bound ρ that is claimed for the cost ratio.

Without loss of generality, throughout this paper it is assumed that for the models on identical and uniform machines, the jobs are numbered in LPT (Longest Processing Time) order, i.e., in a nondecreasing order of their sizes:

$$p_1 \geq p_2 \geq \cdots \geq p_n. \tag{3}$$

Let $N_u = \{J_1, J_2, \ldots, J_u\}$ denote the set of u longest jobs, $1 \leq u \leq n$. Define $p\left(N_u\right) = \sum_{j=1}^{u} p_j$ and $p\left(N\right) = \sum_{j=1}^{n} p_j$.

In the case of identical machines, for an optimal preemptive schedule S_p^* the equality

$$C_{\max}\left(S_{p}^{*}\right) = \max\left\{T, p_1\right\} \tag{4}$$

holds, where $T = p\left(N\right)/m$ is the average machine load. An optimal schedule S_p^* can be found in $O\left(n\right)$ time by a so-called wrap-around algorithm developed in [17]. As proved in [1], in the case of m identical machines the power of preemption for $\Phi = C_{\max}$ is given by $\rho_m = 2 - 2/\left(m+1\right)$. Moreover, it is demonstrated in [1] that a nonpreemptive schedule S_{np}, for which (2) holds (that is, the second inequality of (2) holds), can be found in $O\left(m + n\log n\right)$ time by applying the famous LPT List Scheduling algorithm, which scans the jobs in the order of their LPT numbering and assigns the next job to the first available machine (see also [15]). For the class of instances with $C_{\max}(S_p^*) = p_1$, the value of the power of preemption is strictly smaller than the global bound $2 - 2/\left(m+1\right)$, as established in [19].

Unless stated otherwise, for the model with m uniform machines, throughout this paper assume that the machines are numbered in nonincreasing order of their speeds, i.e.,

$$s_1 \geq s_2 \geq \cdots \geq s_m. \tag{5}$$

Define $S_u = \sum_{i=1}^{u} s_i$, the total speed of the u fastest machines, $1 \leq u \leq m$, and introduce

$$T_u = \frac{p\left(N_u\right)}{S_u}, \quad 1 \leq u \leq m-1, \quad T_m = \frac{p\left(N\right)}{S_m}.$$

According to [9], for an optimal preemptive schedule S_p^* the equality

$$C_{\max}\left(S_{p}^{*}\right) = \max\left\{T_u | 1 \leq u \leq m\right\} \tag{6}$$

holds, and an optimal schedule S_p^* can be found in $O\left(n + m\log m\right)$ time. It is shown in [25] that $\rho_m = 2 - 1/m$, and a nonpreemptive schedule S_{np} for which (2) holds can be found in $O\left(m + n\log n\right)$ time by a version of the LPT List Scheduling algorithm. As clarified in [24], this bound is tight for the class of instances where $C_{\max}\left(S_p^*\right) = T_m$

holds; however, for the class of instances for which $C_{\max}\left(S_p^*\right) = T_u$, $1 \le u \le m-1$, the value of ρ_m becomes $2 - \min\left\{\frac{1}{u}, \frac{1}{m-u}\right\}$. For $m = 2$, a parametric analysis of the power of preemption with respect to the speed of the faster machine is independently performed in [12] and [23]. For $m = 3$, a similar analysis is contained in [23], provided that the machine speeds take at most two values, 1 and $s \ge 1$.

To complete the discussion of the power of preemption of the problems of minimizing the makespan, consider the model with m unrelated machines. An optimal preemptive schedule S_p^* can be found in polynomial time by solving a linear programming problem that determines the values x_{ij}, equal to the total length of the time intervals during which job J_j is processed on machine M_i; see, for example, [14]. The values x_{ij}/p_{ij} can be rounded to produce a nonpreemptive schedule S_{np}. In particular, a rounding procedure that is attributed to Shmoys and Tardos and reproduced in [16] and [7] finds a nonpreemptive schedule S_{np} such that (2) holds for $\rho = 4$. This bound is tight, as proved in [7].

We now turn to considering the issues of the power of preemption for the objective function $\Phi = \sum C_j$ and its weighted counterpart $\Phi = \sum w_j C_j$. In the latter case, job J_j additionally has a positive weight w_j associated with it, which reflects its relative importance.

For identical machines, it is proved in [18] that allowing preemption does not reduce the optimal value of $\Phi = \sum w_j C_j$, i.e., for that model $\rho = 1$. Notice that the problem of minimizing $\Phi = \sum w_j C_j$ is NP-hard, even for two identical machines [3].

The problem of minimizing $\sum C_j$ on unrelated parallel machines belongs to a group of rare scheduling problems for which solving the preemptive version is harder than its nonpreemptive counterpart: finding schedule S_{np}^* reduces to a rectangular assignment problem and takes strongly polynomial time [3, 10], while the problem of finding schedule S_p^* is NP-hard, as proved in [21]. For the problem of minimizing $\Phi = \sum w_j C_j$ on unrelated parallel machines, approximability results established by Sitters in [22] can be interpreted in terms of the power of preemption, which they imply for that very general model $\rho \le 1.81$, which improves a previously known bound of 2 [20].

This paper focuses on the problems of minimizing the unweighted function $\Phi = \sum C_j$ on uniformly related machines. Unlike the problems of minimizing the makespan discussed above, here both versions of the problem, nonpreemptive and preemptive, are polynomially solvable. Indeed, an optimal nonpreemptive schedule S_{np}^* can be found in $O(m + n \log n)$ time [6], while finding an optimal preemptive schedule S_p^* takes $O(n \log n + nm)$ time; see [4, 8, 13]. Detailed descriptions of the corresponding algorithms are given in section 3.

3. Algorithms on uniform machines. An instance I of the problem with n jobs and m parallel uniformly related machines is defined by the list $\mathcal{L}_n = (p_1, p_2, \ldots, p_n)$ of the sizes of the jobs, and the list $\mathcal{M}_m = (s_1, s_2, \ldots, s_m)$ of the machine speeds. The objective is the total completion time $\Phi = \sum C_j$. As assumed in section 2, the jobs are numbered in the LPT order (3), while the machines are numbered in accordance with (5). Without loss of generality, we may assume that $n \ge m$; otherwise, the $m - n$ slowest machines can be removed since they will not be assigned any jobs in any optimal schedule, preemptive or nonpreemptive. This holds since moving the jobs or parts of jobs of one machine to another empty machine that is not slower does not harm the schedule. The running time for this redistribution step is $O(m + n)$, using methods of median selection and partitioning.

Sometimes, to stress for which instance a schedule S is created, we will write

$S(I)$, explicitly referring to instance I.

The algorithm for finding an optimal nonpreemptive schedule S^*_{np} scans the jobs in the order of their numbering and forms the processing sequence on each machine in a backwards manner, starting from the rear end, so that the next job is assigned to the machine on which it makes the smallest contribution to the objective function. Formally, the algorithm can be stated similarly to [2] (and this is a special case of the algorithm for unrelated machines [10, 3]). In the description of the algorithm, Π_i denotes the sequence of jobs assigned to machine M_i, and $\circ$ is the operation of concatenation, i.e., $J_j \circ \Pi$ denotes the sequence obtained by adding job J_j at the beginning of the current sequence Π.

Algorithm QSumNP

1. If necessary, renumber the jobs in accordance with (3) and the machines in accordance with (5).
2. For each machine $M_i, i = 1, 2, \ldots, m$, define $\Pi_i := \varnothing$ and $\omega_i := 1/s_i$.
3. For each j from 1 to n do
   (a) Find the smallest index v with $\omega_v = \min_{1 \le i \le m} \omega_i$.
   (b) Update $\Pi_v := J_j \circ \Pi_v$ and update $\omega_v := \omega_v + \frac{1}{s_v}$.

As shown in [11], for $n \ge m$ Algorithm QSumNP can be implemented in $O(n \log n)$ time (the values ω_i are stored in a priority queue, where even a simple binary heap allows one to implement all steps except for the initial sorting in time $O(n \log m)$). Notice that in schedule S^*_{np}, on each machine the jobs are processed in the order opposite to their numbering, i.e., in nondecreasing order of their sizes, which is known as the Shortest Processing Time (SPT) rule. Each machine M_i has a a multiplier associated with it, denoted by ω_i, which is updated during the run. For each j, $1 \le j \le n$, the algorithm matches job J_j to the smallest available multiplier ω_v, possible ties being broken in favor of the machine with the smallest index; i.e., the job is assigned to the fastest machine associated with the current smallest multiplier. The contribution of job J_j to the total cost $\Phi\left(S^*_{np}\right) = \sum C_j\left(S^*_{np}\right)$ is defined as $\omega_v p_j$ (where ω_v is the value calculated in step 3(a) for j). Notice also that since the value $\Phi\left(S^*_{np}\right)$ is the sum of all contributions of the jobs, the contribution of job J_j may be different from its completion time.

The algorithm for finding an optimal preemptive schedule S^*_p scans the jobs in the SPT order, i.e., in a nondecreasing order of the sizes, opposite to their numbering. Each job is assigned preemptively in such a way that its completion time is minimized. For example, job J_n is assigned to the fastest machine M_1, so that it completes at time $\Delta t = p_n/s_1$. During the time interval $[0, \Delta t]$, machine M_i, $2 \le i \le m$, processes part of job J_{n-i+1} (assuming $m \le n$). Then the remaining part of job J_{n-1} is assigned to machine M_1, where it will be processed for $(p_{n-1} - \Delta t \cdot s_2)/s_1$ time units, while during that time interval each of the other machines will process part of a job (if such a job exists), etc. A formal description of the algorithm given below follows [2].

Algorithm QSumP

1. If necessary, renumber the jobs in accordance with (3) and the machines in accordance with (5).
2. Define $a := 0$.
3. For $i = n, n-1, \ldots, 1$ do
   (a) Compute $\Delta t = p_i/s_1$ and $k := \min\{m, i\}$.
   (b) For v from 0 to $k-1$ do
      i. Schedule job J_{i-v} on machine M_{v+1} during the time interval $[a, a + \Delta t)$.

ii. Update $p_{i-v} := p_{i-v} - \Delta t \cdot s_{v+1}$.

(c) Update $a := a + \Delta t$.

The running time of Algorithm QSumP is $O(n \log n + nm)$. Notice that in schedule S_p^* each job is completed on the fastest machine M_1 (even if it is processed on M_1 during a time interval of zero length), and the completion order follows the SPT rule.

In what follows, when we discuss optimal schedules, we will assume that when multiple optimal schedules exist, the optimal schedule we consider is the one created by the corresponding algorithm given here.

4. Tight instances. For the problem of scheduling n jobs on m uniformly related machines to minimize total completion time $\Phi(S) = \sum_{j=1}^{n} C_j(S)$, let ρ be the power of preemption, i.e., (1) holds for any instance of the problem. An instance I^* is called *tight* if for that instance the following equality holds:

$$\frac{\Phi\left(S_{np}^*(I^*)\right)}{\Phi\left(S_p^*(I^*)\right)} = \rho.$$

Observe that ρ is defined as a supremum of an infinite set of values, and thus, in general, there need not be a tight instance. In such cases, there is a sequence of instances whose sequence of cost ratios approaches ρ. Such a sequence is called a *tight sequence*.

In this section, we establish the existence of a tight sequence that possesses specific properties.

LEMMA 1. *There exists a tight sequence in which for every element of the sequence all jobs are identical, i.e., p_j, $j = 1, \ldots, n$, are the same.*

Proof. It suffices to show that given a set of m machines with speeds $s_1, s_2, \ldots, s_m$ and an upper bound n on the number of jobs, there exists an instance maximizing the power of preemption that is restricted to instances with this set of machines and at most n jobs such that the vector of sizes is binary. This claim implies that there is a tight sequence of instances with binary job sizes. Since zero-sized jobs can be seen to complete at time zero, given a tight sequence with binary job sizes, we can delete the zero-sized jobs from each instance of the sequence, and this does not affect the cost for both the optimal nonpreemptive schedule and the optimal preemptive schedule.

Clearly, we can disregard the instances in which all n jobs have zero size. Therefore, without loss of generality, we can normalize the job sizes so that the largest size is equal to 1. Thus, an instance is associated with a vector of n variables $p_1, p_2, \ldots, p_n$ such that

$$1 = p_1 \geq p_2 \geq \cdots \geq p_n \geq 0. \tag{7}$$

For schedule S_{np}^*, let the contribution of job $J_j \in N$ to $\Phi\left(S_{np}^*\right)$ be $a_j p_j$, where a_j depends on n and m, as well as on the index of the job in the sorted list and the machine speeds. Indeed, the objective function value $\Phi\left(S_{np}^*\right)$ is a linear function of the variables $p_1, \ldots, p_n$ (given the ordering) since, as follows from Algorithm QSumNP, the contribution $\omega_v p_j$ of job J_j to the objective function is the product of its multiplier and its size, and the objective function value is the sum of contributions of the jobs.

Similarly, $\Phi\left(S_p^*\right)$ is a linear function which can be written as $\sum_{j=1}^{n} b_j p_j$. We prove this by induction on the iterations of Algorithm QSumP. Specifically, we show that in each iteration the values of a and of Δt, as well as the updated values of p_j, $J_j \in N$, are all linear functions of the original sizes $p_1, \ldots, p_n$. Recall that Algorithm QSumP

considers the jobs in the order opposite their numbering given by (7), and that in iteration ϕ, where the value of the index i is $n-\phi+1$, the completion time of job J_i is defined. First, since the values of $p_1, p_2, \ldots, p_n$ are updated in each iteration of the algorithm, we show by induction that (7) holds after every iteration, and moreover, after iteration ϕ, $p_{n-\phi+1} = p_{n-\phi+2} = \cdots = p_n = 0$. Due to the sorting, these properties hold before any iterations are performed.

Assume that the properties hold after iteration $\phi - 1$. In iteration ϕ, step 3(a), a time interval of length Δt is determined such that $\Delta t = \frac{p_{n-\phi+1}}{s_1}$, and for $k = \min\{n-\phi+1, m\}$, job $n-v-\phi+2$ for $1 \le v \le k$ is assigned to machine M_v during this time interval. Since the processing time which can be used on machine M_v during this time interval is $\Delta t \cdot s_v$ and $s_1 \ge s_2 \ge \cdots \ge s_m$, we have $\Delta t \cdot s_1 \ge \Delta t \cdot s_2 \ge \cdots \ge \Delta t \cdot s_m$, and since the value $p_{n-v-\phi+2}$ is decreased by $\Delta t \cdot s_v$ for $1 \le v \le k$, the values $p_n, p_{n-1}, \ldots, p_{n-\phi+2}$ are already equal to zero, so $p_{n-k-\phi+1}, \ldots, p_1$ are unchanged, and the sorted order is kept. Moreover, for $v = 1$, the new value of $p_{n-v-\phi+2}$ is zero by the choice of Δt. Finally, we now demonstrate that $\Phi\left(S_p^*\right)$ is a linear function of the original sizes p_j. To do this we show that after each iteration of the algorithm the current values of $p_1, \ldots, p_n$ linear depend on the original sizes of the jobs. Consider iteration ϕ; then for every job J_j the value of p_j either remains the same or decreases by $\Delta t \cdot s_v$ (for some value of v), where Δt depends linearly on the current values of $p_1, p_2, \ldots, p_n$, which is linear in the original sizes using the induction hypothesis. For each job, the completion time of the job is the sum of all Δt values in a prefix of the list of iterations of the algorithm (all iterations up to and including the iteration in which its completion time is defined), and those are also linear in the original sizes of the jobs.

Thus, we have that the power of preemption of this subclass of instances is the optimal value of the following mathematical program:

$$\begin{aligned} \text{maximize} \quad & \frac{\sum_{j=1}^{n} a_j p_j}{\sum_{j=1}^{n} b_j p_j} \\ \text{subject to} \quad & p_1 = 1, \\ & p_j - p_{j+1} \ge 0, \quad 1 \le j \le n-1, \\ & p_n \ge 0. \end{aligned}$$

For this problem, the matrix of the constraints defines a nonempty bounded polytope, and for any feasible solution, the denominator of the objective function is positive. This implies that the optimal value of the objective function, which we denote by λ^*, is finite. Solving the above mathematical program is equivalent to finding a maximizer of the objective function $\sum_{j=1}^{n} a_j p_j - \lambda^* \cdot (\sum_{j=1}^{n} b_j p_j)$ subject to the same set of constraints. The resulting problem is a linear program over a nonempty polytope. Thus, we know that there exists an optimal solution for this linear program that is an extreme point of the polytope. Observe that the structure of the constraint matrix that defines this polytope is such that in each row all entries are zero, except at most one 1 and at most one -1. Such constraint matrices are known to be totally unimodular, and since the right-hand side of the system of constraints is an integer vector, we deduce that all extreme points of the polytope are integral. Since all feasible integral solutions are in fact binary due to (7), Lemma 1 is proved. ∎

In our search for a tight sequence, Lemma 1 allows us to focus on instances in which all processing times are unit, i.e., $p_j = 1$, $J_j \in N$. For the analysis of ρ, we will consider large inputs, with numbers of jobs growing to infinity as justified by the

following lemma.

LEMMA 2. *There is a tight sequence such that the numbers of unit-sized jobs of the instances along the sequence form a monotonically increasing sequence of integers, growing to infinity.*

Proof. Let I be an instance with n unit-sized jobs $J_1, \ldots, J_n$. We create an instance I' with $2n$ unit-sized jobs $J'_1, \ldots, J'_{2n}$ by duplicating each machine; in other words, instead of using m machines $M_1, M_2, \ldots, M_m$ with the speed vector $(s_1, s_2, \ldots, s_m)$ we will use $2m$ machines $M'_1, M'_2, \ldots, M'_{2m-1}, M'_{2m}$ with the speed vector $(s_1, s_1, s_2, s_2, \ldots, s_m, s_m)$. Observe that a nonpreemptive optimal schedule for instance I' can be formed by running on machines M'_{2i-1} and M'_{2i} the same number of jobs that in schedule $S^*_{np}(I)$ are executed on machine M_i, $1 \le i \le m$, and thus the cost of $S^*_{np}(I')$ is exactly twice the cost of $S^*_{np}(I)$. The last claim can be proved by examining the action of Algorithm QSumNP. If the selected machine for job J_j is M_v for input I, then for instance I' jobs J'_{2j-1} and J'_{2j} will be assigned, respectively, to machines M'_{2v-1} and M'_{2v}. On the other hand, for instance I', we can reproduce the assignment in schedule $S^*_p(I)$ twice: on the machines $M'_1, M'_3, \ldots, M'_{2m-1}$ with the odd indices, and on the machines $M'_2, M'_4, \ldots, M'_{2m}$ with the even indices. This results in a feasible preemptive schedule for instance I' whose cost is twice that of $S^*_p(I)$. Hence, the cost of $S^*_p(I')$ is at most twice that of $S^*_p(I)$.

Thus, for any instance I with n unit-sized jobs and m machines, we can form an instance I' with $2n$ unit-sized jobs and $2m$ machines such that

$$\frac{\Phi\left(S^*_{np}(I)\right)}{\Phi\left(S^*_p(I)\right)} \le \frac{\Phi\left(S^*_{np}(I')\right)}{\Phi\left(S^*_p(I')\right)},$$

i.e., the cost ratio for instance I is no more than that for instance I'. Therefore, given a tight sequence, we conclude that there is a tight sequence that is an infinite sequence of instances whereby the sequence of numbers of unit-sized jobs is (strictly) monotone increasing, and the lemma holds. ∎

Lemma 2 demonstrates the existence of a tight sequence rather than of a finite number of tight instances. It plays an important role in the proof of Theorem 2.

Given an instance of the problem, consider the run of Algorithm QSumNP and the run of Algorithm QSumP. Recall that only instances with $m \le n$ need be considered.

Since we deal with instances that contain only unit-sized jobs, the cost of an optimal nonpreemptive schedule depends on the list of the multipliers generated during the run of Algorithm QSumNP. For an instance I, Algorithm QSumNP generates a list of n *used* multipliers, so that for each job the product of its size and the matched multiplier defines its contribution to the total cost. Additionally, upon the completion of the algorithm, each machine M_i supplies a multiplier ω_i, $1 \le i \le m$, which we call a *ready* multiplier. Should the instance under consideration contain another job J_{n+1}, that job would be matched by Algorithm QSumNP to the smallest ready multiplier. If additional jobs arrive, the list of m ready multipliers will be modified, so that if $\frac{k}{s_i}$ is the smallest ready multiplier (where in the case of ties i is the minimum index of a machine whose current multiplier is the smallest one), then $\frac{k+1}{s_i}$ is inserted into the list as the multiplier of machine M_i, while $\frac{k}{s_i}$ is removed from the list.

Another representation would be to create n multipliers of the form $\frac{k}{s_i}$ for $k = 1, 2, \ldots, n$ for each machine M_i, $1 \le i \le m$, and sort this list in nondecreasing order, provided that the elements of equal value are additionally sorted by increasing order of the machine indices. Let $\Omega(I)$ be this sorted list of length $n \cdot m$, where each multiplier

TABLE 1
Runing AlgorithmQSumNP for eight unit-sized jobs.

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
Machine	M_1	M_2	M_1	M_1	M_2	M_3	M_1	M_2
Used multiplier	1/3	1/2	2/3	1	1	1	4/3	3/2

appears in the list as many times as it occurs. In this list, the first n elements are used multipliers, and the $(n+1)$th element is the first ready multiplier. Notice that Algorithm QSumNP actually assigns job J_j according to the jth element in the list so that if the jth element in the list is $\frac{k}{s_i}$, then job J_j is assigned to be processed on machine M_i in the kth position from the rear.

To illustrate the introduced notions, consider an instance of eight unit-sized jobs J_j, $1 \le j \le 8$, to be processed on machines M_1, M_2, and M_3 with the speeds $s_1 = 3$, $s_2 = 2$, and $s_3 = 1$. Running Algorithm QSumNP and scanning the jobs in the order of their numbering, these jobs are allocated to the machines and associated with the used multipliers as shown in Table 1.

Notice that the ties are broken to give preference to a faster machine with a smaller index. The algorithm terminates while delivering ready multipliers 5/3, 2, and 2 for machines M_1, M_2, and M_3, respectively. The corresponding list $\Omega(I)$ is given by

$$(1/3, 1/2, 2/3, 1, 1, 1, 4/3, 3/2, 5/3, 2, 2, 2, 7/3, 5/2, 8/3, 3, 3, 7/2, 4, 4, 5, 6, 7, 8).$$

The following scaling procedure is crucial for further analysis. If necessary, we scale all speeds of the machines in such a way that the smallest ready multiplier generated upon the completion of Algorithm QSumNP is equal to 1 (that is, we multiply all speeds by the smallest ready multiplier). Notice that such a scaling does not affect the cost ratio. For example, to scale the eight-job instance above, we need to multiply all the speeds of the machines by 5/3.

Due to the performed scaling, the smallest ready multiplier generated upon the completion of Algorithm QSumNP is equal to 1. For an instance with n unit-sized jobs in schedule S^*_{np}, no machine is assigned more jobs than its speed, and in particular, a machine whose speed is strictly below 1 has no jobs assigned to it. A machine of speed s will be assigned at least $\lceil s \rceil - 1$ jobs (as otherwise, a ready multiplier on that machine is less than 1). It is assigned at most $\lfloor s \rfloor$ jobs, since a larger number of jobs means that a multiplier strictly above 1 has been used by the algorithm, which means that all multipliers of value 1 have been used and there cannot be a ready multiplier with that value. In particular, Algorithm QSumNP outputs a schedule in which any machine of speed 1 has at most one job assigned to it.

Note that as a result of scaling, the minimum machine speed may become less than 1, however, this does not happen in instances which we are interested in, as we show now. This will allow us in further analysis to assume that no machine speed is below 1.

LEMMA 3. *Given an instance I of n unit-sized jobs and m machines such that the minimum machine speed is smaller than* 1 *(i.e.,* $\min_i s_i < 1$*), there is an instance I' of n unit-sized jobs and m machines such that the minimum machine speed in I' equals* 1*, and the cost ratio for I' is no less than that for I.*

Proof. Take an instance I for which $\min_i s_i < 1$ and transform it into instance I' as described below. For every machine M whose speed is below 1, change its

speed to 1 (without changing the numbering of the machines). Since instance I is scaled, it follows from the structure of Algorithm QSumNP that in schedule $S^*_{np}(I)$ the machines with speeds below 1 receive no jobs. Then, in the resulting optimal schedule $S^*_{np}(I')$ for instance I', the assignment of jobs to machines remains as in $S^*_{np}(I)$, since the machines are numbered in accordance with (5) and the ties are broken in favor of the machines with smaller indices, and since the smallest ready multiplier is equal to 1. The cost of an optimal nonpreemptive schedule does not change, i.e., $\Phi\left(S^*_{np}(I')\right) = \Phi\left(S^*_{np}(I)\right)$. The described transformation may only decrease the cost of an optimal preemptive schedule, since when speeds increase, it is still possible to use any previous optimal schedule as a (not necessarily optimal) schedule for I', i.e., $\Phi\left(S^*_{p}(I')\right) \le \Phi\left(S^*_{p}(I)\right)$. Thus, the cost ratio for instance I is no larger than that for instance I'. This proves the claim. ∎

Now, we show a stronger property for the speeds. The property is that among instances with n unit-sized jobs and $m \le n$ machines, it suffices to consider instances with at most one machine of speed $s > 1$ and all other $m-1$ machines of speed 1 (by Lemma 3, no machine has a speed below 1).

To show this, we do the following. Consider an appropriately scaled instance I that, in particular, contains a machine $M^{(x)}$ of speed $s^{(x)} = x + \alpha > 1$, and a machine $M^{(y)}$ of speed $s^{(y)} = y + \beta > 1$, where $x, y \ge 1$ are integers and α and β are nonnegative numbers such that $\alpha, \beta \le 1$. If one of the speeds $s^{(x)}$ and $s^{(y)}$ is an integer (or if both are), the values α and β are selected (out of 0 and 1) in such a way that Algorithm QSumNP assigns x jobs to machine $M^{(x)}$ and y jobs to machine $M^{(y)}$.

We examine the effect of changing the speeds of machines $M^{(x)}$ and $M^{(y)}$ to $s^{(x)} + s^{(y)} - 1$ and 1, respectively, keeping the other machines untouched. We show in Lemma 4 that this modification cannot increase the value $\Phi\left(S^*_p\right)$ of the optimal preemptive schedule, and in Lemma 5 we prove that it cannot decrease the value $\Phi\left(S^*_{np}\right)$ of the optimal nonpreemptive schedule. Applying this modification as long as there are at least two machines with speeds strictly larger than 1 results in a new instance I' whose cost ratio is no less than the cost ratio of I, and it has an additional property that there is at most one machine of speed larger than 1. Notice that we may assume that there is exactly one machine of speed larger than 1 in instance I'; otherwise, if no machine has a speed larger than 1, then the machines are identical and the cost ratio for that instance is 1.

LEMMA 4. *Changing the speeds of the two machines $M^{(x)}$ and $M^{(y)}$ from $s^{(x)}$ and $s^{(y)}$ to $s^{(x)} + s^{(y)} - 1$ and 1, respectively, does not increase the cost of an optimal preemptive schedule.*

Proof. Consider an optimal preemptive schedule S^*_p for the given initial instance I. We will emulate the allocation of jobs to $M^{(x)}$ and $M^{(y)}$ of speeds $s^{(x)}$ and $s^{(y)}$, respectively, using the two machines of speeds $s^{(x)}+s^{(y)}-1$ and 1. To do this, consider a (maximal) time interval T of length t such that in schedule S^*_p a part of one job (say, J') runs on machine $M^{(x)}$ of speed $s^{(x)}$, and a part of another job (say, J'') runs on machine $M^{(y)}$ of speed $s^{(y)}$; see Figure 1(a). The proof below is presented for the case when both machines $M^{(x)}$ and $M^{(y)}$ are busy in interval T (if this is not the case, no job will be run instead of running the missing job).

The processing amounts of job J' and of job J'' in interval T (the sizes of parts of these jobs that are processed) are equal to $ts^{(x)}$ and $ts^{(y)}$, respectively.

We show that the same processing amount of each of these jobs in interval T can

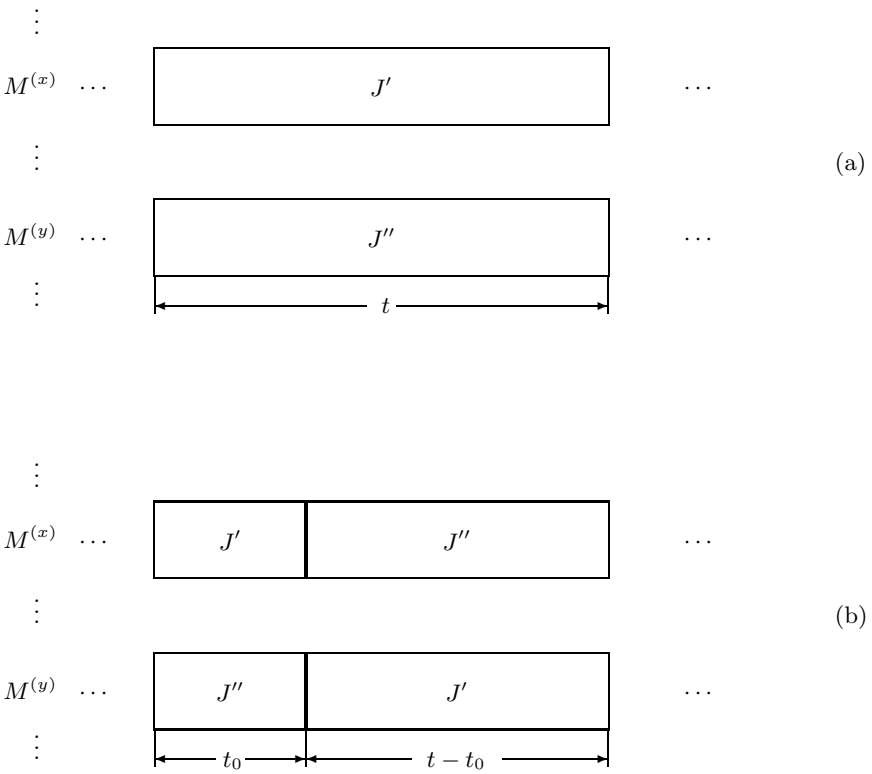

FIG. 1. *Interval T* (a) *in schedule S_p^*;* (b) *in the modified schedule.*

be achieved by the following modification. Assign speed $s^{(x)} + s^{(y)} - 1$ to machine $M^{(x)}$, and speed 1 to machine $M^{(y)}$. Define

$$t_0 := t \cdot \frac{s^{(x)} - 1}{s^{(x)} + s^{(y)} - 2}.$$

Process job J' during the first t_0 time units of interval T on machine $M^{(x)}$, and in the remaining part of interval T on machine $M^{(y)}$. Similarly, run job J'' during the first t_0 time units of interval T on machine $M^{(y)}$ of speed 1, and in the remaining part of interval T on machine $M^{(x)}$; see Figure 1(b). This transformation does not affect other jobs and machines, and none of J' and J'' is processed on two machines simultaneously.

The total processing amount of job J' during T in the modified schedule is

$$t \cdot \frac{s^{(x)} - 1}{s^{(x)} + s^{(y)} - 2} \cdot (s^{(x)} + s^{(y)} - 1) + t \left(1 - \frac{s^{(x)} - 1}{s^{(x)} + s^{(y)} - 2}\right) = t \cdot s^{(x)},$$

i.e., is equal to the processing amount of J' during interval T before the modification.

Similarly, the total processing amount of job J'' during interval T is

$$t \cdot \frac{s^{(x)}-1}{s^{(x)}+s^{(y)}-2} \cdot 1 + t\left(1 - \frac{s^{(x)}-1}{s^{(x)}+s^{(y)}-2}\right) \cdot (s^{(x)}+s^{(y)}-1) = t \cdot s^{(y)},$$

again the same as the total processing amount of this job during T before the modification. Since Algorithm QSumP creates a finite number of intervals like interval T in the above transformation, it follows that this process of dealing with these intervals one by one will terminate. The cost of the constructed preemptive schedule for the modified instance is the same as the cost for the initial instance I. ∎

LEMMA 5. *Changing the speeds of the two machines $M^{(x)}$ and $M^{(y)}$ from $s^{(x)} = x+\alpha$ and $s^{(y)} = y+\beta$ to $s^{(x)}+s^{(y)}-1 = x+y+\alpha+\beta-1$ and 1, respectively, does not decrease the cost of an optimal nonpreemptive schedule.*

Proof. Consider optimal nonpreemptive schedules $S^*_{np}(I)$ for I, and $S^*_{np}(I')$ for I', where I' is the instance with the modified speeds (recall that we assume that the schedules are created by Algorithm QSumNP). According to our notation, for I, Algorithm QSumNP assigns x and y jobs to machines $M^{(x)}$ and $M^{(y)}$, respectively.

For instance I, let $\Omega(I)$ be the sorted list of $n \cdot m$ multipliers that consists of n multipliers for each machine. Recall that in this list, the first n elements are used multipliers, and the $(n+1)$th element is the first ready multiplier, which is equal to 1. For instance I', list $\Omega(I')$ is defined similarly, though the value of the $(n+1)$th multiplier is yet to be evaluated. Since the jobs are unit-sized, the cost of a schedule (and in particular, the costs of $S^*_{np}(I)$ and of $S^*_{np}(I')$) is equal to the sum of all used multipliers in list $\Omega(I)$ (or in list $\Omega(I')$, respectively). For instance I, all fractional multipliers, i.e., those strictly less than 1, appear among the first n elements of $\Omega(I)$ and thus they are used multipliers.

Making the transfer from instance I to instance I', we transform list $\Omega(I)$ into list $\Omega(I')$ by the removal of all multipliers related to machines $M^{(x)}$ and $M^{(y)}$, followed by the insertion of multipliers of the form $\frac{k}{s^{(x)}+s^{(y)}-1}$ and k, where $1 \le k \le n$ is an integer. The later multipliers are, respectively, supplied by machines $M^{(x)}$ and $M^{(y)}$ in instance I'.

By assumption, in list $\Omega(I)$ the number of fractional multipliers is no larger than n, and the number of multipliers no larger than 1 is at least $n+1$. Before we proceed with the analysis for instance I', we show that the value of the $(n+1)$th element of $\Omega(I')$ is 1, since we are currently examining only instances for which the first ready multiplier is equal to 1. For this purpose, we show that the number of fractional multipliers in $\Omega(I')$ is no larger than that of $\Omega(I)$, and that the number of multipliers no larger than 1 in $\Omega(I')$ is no smaller than that of $\Omega(I)$.

For any instance, a machine M of speed s supplies $\lceil s \rceil - 1$ fractional multipliers. The only difference between $\Omega(I)$ and $\Omega(I')$ in terms of fractional multipliers can be the difference between the numbers of such multipliers supplied by $M^{(x)}$ and $M^{(y)}$. Thus, the change in the number of fractional multipliers is $-((\lceil x+\alpha \rceil - 1)+(\lceil y+\beta \rceil - 1))+(\lceil x+\alpha+y+\beta-1 \rceil - 1)$ (note that $M^{(y)}$ no longer supplies a fractional multiplier, as its modified speed is 1). Due to a simple property $\lceil \lambda_1 \rceil + \lceil \lambda_2 \rceil \ge \lceil \lambda_1 + \lambda_2 \rceil$, which holds for any real λ_1 and λ_2, we have that the change is nonpositive.

For any instance, a machine M of speed s supplies $\lfloor s \rfloor$ multipliers no larger than 1. The only difference between $\Omega(I)$ and $\Omega(I')$ in terms of such multipliers can be the difference between the numbers of such multipliers supplied by $M^{(x)}$ and $M^{(y)}$. Thus, the change in the number of multipliers no larger than 1 is $-((\lfloor x+\alpha \rfloor)+(\lfloor y+$

$\beta\rfloor)) + (\lfloor x+\alpha+y+\beta-1\rfloor+1)$ (in this case, $M^{(y)}$ now supplies one multiplier of value 1). Due to a simple property $\lfloor\lambda_1\rfloor+\lfloor\lambda_2\rfloor \le \lfloor\lambda_1+\lambda_2\rfloor$, which holds for any real λ_1 and λ_2, we have that the change is nonnegative.

Thus, I' is a valid input. Compare the prefixes of length n in the lists $\Omega(I)$ and $\Omega(I')$, which are essentially the lists of used multipliers. It follows that the prefix of length n of $\Omega(I')$ contains all fractional multipliers of all machines, and for each machine other than $M^{(x)}$ and $M^{(y)}$, these multipliers are the same multipliers as in $\Omega(I)$. The remaining multipliers in the prefixes under consideration are the fractional multipliers supplied by machines $M^{(x)}$ and $M^{(y)}$, and possibly some multipliers of value 1 (ensuring that the total number of multipliers in each prefix is n).

Let Γ denote the sum of fractional multipliers for all machines excluding machines $M^{(x)}$ and $M^{(y)}$, and let g be the number of such multipliers; both these values are the same for lists $\Omega(I)$ and $\Omega(I')$. The value $\frac{x(x+1)}{2(x+\alpha)}+\frac{y(y+1)}{2(y+\beta)}$ represents the sum of the $x+y$ smallest multipliers associated with machines $M^{(x)}$ and $M^{(y)}$ in list $\Omega\,(I)$. Since in instance I machine $M^{(x)}$ receives x jobs, and machine $M^{(y)}$ receives y jobs, it follows that the used multipliers supplied by machines $M^{(x)}$ and $M^{(y)}$ are of the form $\frac{k}{x+\alpha}$ for $k=1,2,\dots,x$, and of the form $\frac{k}{y+\beta}$ for $k=1,2,\dots,y$, respectively. All remaining used multipliers in $\Omega(I)$ are equal to 1, and there are $n-g-(x+y)$ such multipliers. Thus, the cost for I can be computed as $\Gamma+(n-g-x-y)+\frac{x(x+1)}{2(x+\alpha)}+\frac{y(y+1)}{2(y+\beta)}$.

The proof is split into two cases, depending on the value of $\alpha+\beta$. Notice that the equality

$$\frac{z(z+1)}{z+\gamma} = z+1-\gamma-\frac{\gamma(1-\gamma)}{z+\gamma} \tag{8}$$

holds for any $z>0$ and $\gamma\ge 0$. We will apply it several times, always for a positive integer z and $0\le\gamma\le 1$. In particular, for $z=x$ and $\gamma=\alpha$ we have

$$\frac{x(x+1)}{x+\alpha} = x+1-\alpha-\frac{\alpha(1-\alpha)}{x+\alpha}, \tag{9}$$

and for $z=y$ and $\gamma=\beta$ we have

$$\frac{y(y+1)}{y+\beta} = y+1-\beta-\frac{\beta(1-\beta)}{y+\beta}. \tag{10}$$

Case 1. Assume that $\alpha+\beta\ge 1$. If $\alpha+\beta>1$, we have $x+y+\alpha+\beta-1> x+y$, so $M^{(x)}$ supplies $x+y$ fractional multipliers for I' of values $\frac{k}{x+y+\alpha+\beta-1}$ for $k=1,2,\dots,x+y$. The cost of the corresponding schedule becomes $\Gamma+(n-g-x-y)+\frac{(x+y)(x+y+1)}{2(x+y+\alpha+\beta-1)}$.

If $\alpha+\beta=1$, we have $x+y+\alpha+\beta-1=x+y$, so that machine $M^{(x)}$ supplies $x+y-1$ fractional multipliers of values $\frac{k}{x+y+\alpha+\beta-1}$ for $k=1,2,\dots,x+y-1$. In this case, the cost of the corresponding schedule is $\Gamma+(n-g-x-y+1)+\frac{(x+y-1)(x+y)}{2(x+y+\alpha+\beta-1)}=\Gamma+(n-g-x-y)+\frac{(x+y)(x+y+1)}{2(x+y+\alpha+\beta-1)}$, since in this case $x+y+\alpha+\beta-1=x+y$ holds.

Comparing the cost to $\Gamma+(n-g-x-y)+\frac{x(x+1)}{2(x+\alpha)}+\frac{y(y+1)}{2(y+\beta)}$, in order to show that the cost for I' is no smaller than that of I, we show that the following holds:

$$\frac{(x+y)(x+y+1)}{2(x+y+\alpha+\beta-1)} \ge \frac{x(x+1)}{2(x+\alpha)}+\frac{y(y+1)}{2(y+\beta)}. \tag{11}$$

Applying (8) with $z = x + y$ and $\gamma = \alpha + \beta - 1$, we deduce

$$\begin{aligned}\frac{(x+y)(x+y+1)}{x+y+\alpha+\beta-1} &= x+y+1-(\alpha+\beta-1)-\frac{(\alpha+\beta-1)(2-\alpha-\beta)}{x+y+\alpha+\beta-1}\\ &= x+y+2-(\alpha+\beta)-\frac{(\alpha+\beta-1)(2-\alpha-\beta)}{x+y+\alpha+\beta-1}.\end{aligned}$$

We first prove that

$$\frac{(\alpha+\beta-1)(2-\alpha-\beta)}{x+y+\alpha+\beta-1} \le \frac{\alpha(1-\alpha)}{x+\alpha}+\frac{\beta(1-\beta)}{y+\beta}, \tag{12}$$

which is equivalent to showing that the expression

$$\begin{aligned}G :=\ &(\alpha+\beta-1)(2-\alpha-\beta)(x+\alpha)(y+\beta)-\alpha(1-\alpha)(x+y+\alpha+\beta-1)(y+\beta)\\ &-\beta(1-\beta)(x+y+\alpha+\beta-1)(x+\alpha)\end{aligned}$$

is nonpositive. Using simple algebra we get that

$$\begin{aligned}G &= (x+1)x\beta(\beta-1)+(y+1)y\alpha(\alpha-1)-2(\alpha-1)(\beta-1)xy\\ &= x(\beta-1)((x+1)\beta+(1-\alpha)y)+y(\alpha-1)((y+1)\alpha+(1-\beta)x).\end{aligned}$$

Since $0 \le \alpha \le 1$ and $0 \le \beta \le 1$, we obtain that $G \le 0$, i.e., (12) holds. Using (12) we obtain

$$\begin{aligned}&x+y+2-(\alpha+\beta)-\frac{(\alpha+\beta-1)(2-\alpha-\beta)}{x+y+\alpha+\beta-1}\\ &\quad\ge x+y+2-(\alpha+\beta)-\frac{\alpha(1-\alpha)}{x+\alpha}-\frac{\beta(1-\beta)}{y+\beta}\\ &\quad= \left[(x+1)-\alpha-\frac{\alpha(1-\alpha)}{x+\alpha}\right]+\left[(y+1)-\beta-\frac{\beta(1-\beta)}{y+\beta}\right].\end{aligned}$$

Applying (9) and (10), we finally deduce that

$$x+y+2-(\alpha+\beta)-\frac{(\alpha+\beta-1)(2-\alpha-\beta)}{x+y+\alpha+\beta-1} \ge \frac{x(x+1)}{(x+\alpha)}+\frac{y(y+1)}{(y+\beta)},$$

i.e., (11) holds.

Case 2. Assume that $\alpha+\beta<1$. If $\alpha+\beta>0$, then machine $M^{(x)}$ supplies $x+y-1$ fractional multipliers, and the cost of the corresponding schedule is $\Gamma+(n-g-x-y+1)+\frac{(x+y-1)(x+y)}{2(x+y+\alpha+\beta-1)}$. If $\alpha=\beta=0$, then machine $M^{(x)}$ supplies $x+y-2$ fractional multipliers, and the cost of the corresponding schedule is $\Gamma+(n-g-x-y+2)+\frac{(x+y-2)(x+y-1)}{2(x+y+\alpha+\beta-1)} = \Gamma+(n-g-x-y+1)+\frac{(x+y)(x+y-1)}{2(x+y+\alpha+\beta-1)}$, since in this case $x+y+\alpha+\beta-1 = x+y-1$ holds.

Comparing the cost of the corresponding schedule to $\Gamma+(n-g-x-y)+\frac{x(x+1)}{2(x+\alpha)}+\frac{y(y+1)}{2(y+\beta)}$, in order to show that this cost for I' is no smaller than that of I, we show that the following holds:

$$\frac{(x+y-1)(x+y)}{2(x+y+\alpha+\beta-1)}+1 \ge \frac{x(x+1)}{2(x+\alpha)}+\frac{y(y+1)}{2(y+\beta)}. \tag{13}$$

Applying (8) with $z = x + y - 1$ and $\gamma = \alpha + \beta$, we deduce

$$\frac{(x+y-1)(x+y)}{x+y+\alpha+\beta-1} + 2 = (x+y) - (\alpha+\beta) + 2 - \frac{(\alpha+\beta)(1-\alpha-\beta)}{x+y+\alpha+\beta-1}.$$

Next, we prove that

$$\frac{(\alpha+\beta)(1-\alpha-\beta)}{x+y+\alpha+\beta-1} \leq \frac{\alpha(1-\alpha)}{x+\alpha} + \frac{\beta(1-\beta)}{y+\beta}, \tag{14}$$

which is equivalent to showing that the expression

$$\begin{aligned} H :=\ & (\alpha+\beta)(1-\alpha-\beta)\,(x+\alpha)\,(y+\beta) - \alpha(1-\alpha)\,(x+y+\alpha+\beta-1)\,(y+\beta) \\ & -\beta(1-\beta)\,(x+y+\alpha+\beta-1)\,(x+\alpha) \end{aligned}$$

is nonpositive. Compute

$$\begin{aligned} -H = {} & x^2 \cdot \beta(1-\beta) + x \cdot (2\alpha\beta + \beta^2 - \beta) + y^2 \cdot \alpha(1-\alpha) + y \cdot (2\alpha\beta + \alpha^2 - \alpha) \\ & + xy \cdot [\alpha(1-\alpha) + \beta(1-\beta) - (\alpha+\beta)(1-\alpha-\beta)] - 2\alpha\beta(1-\alpha-\beta). \end{aligned}$$

Since $x, y \geq 1$ and $1 \geq \alpha, \beta \geq 0$, we have

$$\begin{aligned} x^2 \cdot \beta(1-\beta) + x \cdot (2\alpha\beta + \beta^2 - \beta) &\geq x(\beta(1-\beta) + 2\alpha\beta + \beta^2 - \beta) = 2x\alpha\beta \geq 0, \\ y^2 \cdot \alpha(1-\alpha) + y \cdot (2\alpha\beta + \alpha^2 - \alpha) &\geq y(\alpha(1-\alpha) + 2\alpha\beta + \alpha^2 - \alpha) = 2y\alpha\beta \geq 0, \end{aligned}$$

and

$$\begin{aligned} & xy \cdot [\alpha(1-\alpha) + \beta(1-\beta) - (\alpha+\beta)(1-\alpha-\beta)] - 2\alpha\beta(1-\alpha-\beta) \\ & \quad \geq xy(\alpha(1-\alpha) + \beta(1-\beta) - (\alpha+\beta)(1-\alpha-\beta) - 2\alpha\beta(1-\alpha-\beta)) \\ & \quad = 2xy\alpha\beta \cdot (\alpha+\beta) \geq 0, \end{aligned}$$

by $\alpha + \beta < 1$ and $x, y \geq 1$.

Thus, $H \leq 0$ and (14) holds. Using (14), we obtain

$$\begin{aligned} & (x+y) - (\alpha+\beta) + 2 - \frac{(\alpha+\beta)(1-\alpha-\beta)}{x+y+\alpha+\beta-1} \\ & \quad \geq (x+y) - (\alpha+\beta) + 2 - \frac{\alpha(1-\alpha)}{x+\alpha} - \frac{\beta(1-\beta)}{y+\beta} \\ & \quad = \left[x + 1 - \alpha - \frac{\alpha(1-\alpha)}{x+\alpha}\right] + \left[y + 1 - \beta - \frac{\beta(1-\beta)}{y+\beta}\right]. \end{aligned}$$

Applying (9) and (10), we finally deduce that

$$(x+y) - (\alpha+\beta) + 2 - \frac{(\alpha+\beta)(1-\alpha-\beta)}{x+y+\alpha+\beta-1} \geq \frac{x(x+1)}{(x+\alpha)} + \frac{y(y+1)}{(y+\beta)},$$

i.e., (13) holds. ∎

The remaining lemmas of this section will use all properties proved above, and will impose a final restriction on the types of instances that should be analyzed for computing ρ_m or ρ.

DEFINITION 1. *An instance I with n unit-sized jobs and m machines is called* a good input *if it satisfies the following conditions:*

(a) $n \geq m$;
(b) *machine* M_1 *has speed* s, $1 < s \leq n$, *while the speed of each of the remaining machines* $M_2, \ldots, M_m$ *is* 1;
(c) *in schedule* $S^*_{np}(I)$ *found by Algorithm QSumNP, at least one of the unit speed machines is not assigned a job, so that the smallest ready multiplier is equal to* 1.

LEMMA 6. *Any instance* I *can be converted into a good input without decreasing the cost ratio.*

Proof. As proved earlier in this section, it suffices to consider an initial instance I of n unit-sized jobs, and m machines of speeds no smaller than 1, where $n \geq m$. Moreover, in I the speeds are scaled so that the smallest ready multiplier is equal to 1. Further, by Lemmas 4 and 5, at least $m-1$ of these machines are of speed 1. We may assume that instance I contains a faster machine M_1 of speed $s > 1$; otherwise, all machines are identical.

Assume that in schedule $S^*_{np}(I)$ each slower machine of speed 1 is assigned a job. Then the only machine that can supply the smallest ready multiplier equal to 1 is the faster machine M_1. This is, however, impossible since Algorithm QSumNP breaks ties for the smallest current multiplier in favor of a machine with a smaller index.

To prove the lemma, we need only show that the speed s of machine M_1 is at most n.

Assume that $s > n$. Let $s' = n$, and let I' be a good input obtained from I by changing the speed of machine M_1 to s'. We claim that in both schedules $S^*_{np}(I)$ and $S^*_{np}(I')$ all jobs are assigned to the first machine. This last claim holds since the first n multipliers generated on the first machine are no larger than 1, while all the multipliers related to the other machines are no smaller than 1, and Algorithm QSumNP breaks ties in favor of a machine of a smaller index. Considering the jobs in the order of their completion, we get $C_k(S^*_{np}(I)) = \frac{k}{s}$ and $C_k(S^*_{np}(I')) = \frac{k}{s'}$, $1 \leq k \leq n$. Thus, the ratio $\Phi\left(S^*_{np}(I)\right)/\Phi\left(S^*_{np}(I')\right)$ between the costs of optimal nonpreemptive schedules for instances I and I' is $\frac{s'}{s}$.

We show now that the ratio $\Phi\left(S^*_p(I)\right)/\Phi\left(S^*_p(I')\right)$ between the costs of optimal preemptive schedules for I and I' is at least $\frac{s'}{s}$. To show this, consider an optimal preemptive schedule $S^*_p(I)$. Multiply the speed of each machine by a factor of $\frac{s'}{s}$. In the resulting schedule S_p, all completion times increase by a factor of $\frac{s}{s'}$, compared to the completion times in schedule $S^*_p(I)$. Increase the speed of each machine, except machine M_1, which now has a speed of $s' = n$, back to its original speed of 1, as in I. We obtain instance I'. Consider schedule $S_p(I')$, in which the starting times of parts of jobs on all machines are kept as in schedule S_p. Such a schedule is feasible since in instance I' the speeds of machines $M_2, \ldots M_m$ have been increased from $\frac{s'}{s}$ to 1. Thus, the cost of $S_p(I')$ is at most $\frac{s}{s'}$ times the cost of $S^*_p(I)$, i.e., $\Phi\left(S^*_p(I')\right) \leq \Phi\left(S_p(I')\right) \leq \frac{s}{s'}\Phi\left(S^*_p(I)\right)$, as required.

Observe that

$$\frac{\Phi\left(S^*_{np}(I)\right)}{\Phi\left(S^*_p(I)\right)} = \frac{\frac{s'}{s}\Phi\left(S^*_{np}(I')\right)}{\Phi\left(S^*_p(I)\right)} \leq \frac{\frac{s'}{s}\Phi\left(S^*_{np}(I')\right)}{\frac{s'}{s}\Phi\left(S^*_p(I')\right)} = \frac{\Phi\left(S^*_{np}(I')\right)}{\Phi\left(S^*_p(I')\right)},$$

so that the cost ratio for instance I is no more than that for a good input I'. This concludes the proof. ∎

In fact, for analyzing ρ, we may further tighten the conditions of a good input as

shown below.

LEMMA 7. *Let I be a good input with n jobs and $m \leq n$ machines, and a fast machine of speed $s \in (1, n]$. Then, there exists a good input I' with n jobs and n machines, a fast machine of the same speed s, such that the cost ratio for I' is no less than that for I.*

Proof. If $m = n$ in instance I, we are done. Thus, assume $m < n$. Add to I $n-m$ machines of speed 1 to obtain I' (with exactly $n-1$ machines of speed 1). These extra machines will not affect the cost of an optimal nonpreemptive schedule, since Algorithm QSumNP assigns no job to any of them. This implies that $\Phi\left(S^*_{np}(I')\right) = \Phi\left(S^*_{np}(I)\right)$. On the other hand, it follows that $\Phi\left(S^*_p(I)\right) \geq \Phi\left(S^*_p(I')\right)$, since it is possible not to use the added machines in a preemptive schedule. Thus, I' is a good input with $m = n$, and the cost ratio for I' is no less than that for the original instance I. ∎

To summarize the findings of this section, we present the following statement.

THEOREM 1. *For analyzing ρ, there is a tight sequence such that each instance in the sequence is a good input with equal numbers of unit-sized jobs and machines, and furthermore, the sequence of numbers of unit-sized jobs along the tight sequence is monotonically increasing and tends to infinity.*

Each instance I that is described in Theorem 1 can be represented by a pair (n, s), where n is the number of unit-sized jobs and the number of processing machines, while s, $1 \leq s \leq n$, is the speed of the faster machine M_1, with the other machines being of speed 1.

5. The value of the power of preemption. Let I be an instance of a tight sequence that satisfies the conditions of Theorem 1. For the optimal preemptive schedule, the value of the objective function can be found as stated below.

LEMMA 8. *Let S^*_p be an optimal preemptive schedule for I. Then*

$$\Phi\left(S^*_p\right) = s\left(\frac{s-1}{s}\right)^{n+1} + n + 1 - s. \tag{15}$$

Proof. Applying Algorithm QSumP, we find an optimal schedule S^*_p in which the jobs are completed on the fast machine M_1 in accordance with the increasing sequence $C_1\left(S^*_p\right)$, $C_2\left(S^*_p\right), \ldots, C_n\left(S^*_p\right)$. It is clear that $C_1\left(S^*_p\right) = 1/s$, and for k, $1 \leq k \leq n-1$, job J_{k+1} is processed in the time interval $\left[0, C_k\left(S^*_p\right)\right]$ on slow machines (since there are sufficiently many machines to start it at time zero), starts on machine M_1 at time $C_k\left(S^*_p\right)$, and is processed there during $\left(1 - C_k\left(S^*_p\right)\right)/s$ time units, i.e.,

$$C_{k+1}\left(S^*_p\right) = C_k\left(S^*_p\right) + \left(1 - C_k\left(S^*_p\right)\right)/s, \quad 1 \leq k \leq n-1. \tag{16}$$

We use the recursive relation (16) to prove by induction that

$$C_k\left(S^*_p\right) = 1 - \left(\frac{s-1}{s}\right)^k, \quad 1 \leq k \leq n. \tag{17}$$

It is clear that (17) holds for $C_1\left(S^*_p\right) = 1/s$. Assume that it holds for all values of k, $1 \leq k \leq q < n$, and prove that it also holds for $k = q+1$. Using (16) and (17)

for $k = q$, we obtain

$$C_{q+1}\left(S_p^*\right) = C_q\left(S_p^*\right) + \left(1 - C_q\left(S_p^*\right)\right)/s = \frac{1}{s} + \left(\frac{s-1}{s}\right) C_q\left(S_p^*\right)$$
$$= \frac{1}{s} + \left(\frac{s-1}{s}\right)\left(1 - \left(\frac{s-1}{s}\right)^q\right)$$
$$= \frac{1}{s} + \frac{s-1}{s} - \left(\frac{s-1}{s}\right)\left(\frac{s-1}{s}\right)^q = 1 - \left(\frac{s-1}{s}\right)^{q+1},$$

as required.

Then we derive

$$\Phi\left(S_p^*\right) = \sum_{k=1}^{n} C_k\left(S_p^*\right) = \sum_{k=1}^{n}\left(1 - \left(\frac{s-1}{s}\right)^k\right) = s\left(\frac{s-1}{s}\right)^{n+1} + n + 1 - s,$$

which proves the lemma. ∎

Now, we consider the value of the objective function of an optimal nonpreemptive schedule for instance I.

LEMMA 9. *Let S_{np}^* be an optimal nonpreemptive schedule for the instance I. Then*

$$\Phi\left(S_{np}^*\right) \le n - \frac{s}{2} + \frac{1}{2}, \tag{18}$$

and if s is integral, then $\Phi\left(S_{np}^*\right) = n - \frac{s}{2} + \frac{1}{2}$.

Proof. It is clear that Algorithm QSumNP assigns at least $s-1$ jobs to machine M_1.

First, assume that the speed s of the fast machine is an integer. If Algorithm QSumNP assigns exactly $s-1$ jobs to machine M_1, then there are no more jobs left, since for the next job, if it existed, the multiplier on each machine would be 1, and that job would be assigned to M_1 due to the tie-breaking rule. However, the condition $n = s-1$ contradicts the assumption $s \le n$. Thus, in schedule S_{np}^*, Algorithm QSumNP assigns exactly s jobs to the fast machine M_1, while each of the remaining jobs is processed alone on a slow machine, which is always available.

The completion times of the jobs in schedule S_{np}^* are defined by

$$C_k\left(S_{np}^*\right) = \frac{k}{s}, \quad 1 \le k \le s,$$
$$C_k\left(S_{np}^*\right) = 1, \quad s+1 \le k \le n.$$

This implies that

$$\Phi\left(S_{np}^*\right) = \sum_{k=1}^{n} C_k\left(S_{np}^*\right) = \sum_{k=1}^{s} \frac{k}{s} + (n-s) = n - \frac{1}{2}s + \frac{1}{2},$$

i.e., for an integer s the claim holds.

If s is not integral, then Algorithm QSumNP assigns $\lceil s-1 \rceil$ jobs to the fast machine M_1, while each of the remaining jobs is processed alone on a slow machine. Denote $\bar{s} = \lceil s-1 \rceil$. The completion times of jobs are defined by

$$C_k\left(S_{np}^*\right) = \frac{k}{s}, \quad 1 \le k \le \bar{s},$$
$$C_k\left(S_{np}^*\right) = 1, \quad \bar{s}+1 \le k \le n,$$

so that

$$\Phi\left(S^*_{np}\right) = \sum_{k=1}^{n} C_k\left(S^*_{np}\right) = \frac{\bar{s}\cdot(\bar{s}+1)}{2s} + (n-\bar{s}).$$

By $s-1 < \bar{s} < s$, we have

$$0 > \frac{1}{2}(s-\bar{s}-1)\frac{s-\bar{s}}{s} = \frac{\bar{s}\cdot(\bar{s}+1)}{2s} - \bar{s} + \frac{s}{2} - \frac{1}{2},$$

which implies that

$$\Phi\left(S^*_{np}\right) = \frac{\bar{s}\cdot(\bar{s}+1)}{2s} + (n-\bar{s}) < n - \frac{s}{2} + \frac{1}{2},$$

which proves the lemma. ∎

It follows from Lemmas 8 and 9 that for every good input I with n unit-sized jobs and n machines, we have that

$$\frac{\Phi\left(S^*_{np}(I)\right)}{\Phi\left(S^*_{p}(I)\right)} \le \sup_{1\le s\le n} \frac{n-\frac{s}{2}+\frac{1}{2}}{s\left(\frac{s-1}{s}\right)^{n+1}+n+1-s}. \tag{19}$$

For a positive integer n and a real s such that $1 \le s \le n$, we let $\psi(n,s) = \frac{n-\frac{s}{2}+\frac{1}{2}}{s\left(\frac{s-1}{s}\right)^{n+1}+n+1-s}$, and $\psi_n = \sup_{1\le s\le n}\psi(n,s)$. Our next goal is to show that there is a constant upper bound on the sequence $\{\psi_n\}_n$. Here we show a bound of 4.

LEMMA 10. *For every positive integer n and real s such that $1 \le s \le n$, we have $\psi(n,s) \le 4$.*

Proof. Since $s \le n$, we have $n+1-s > 0$, and as $s\left(\frac{s-1}{s}\right)^{n+1} \ge 0$, the denominator of the definition of $\psi(n,s)$ is positive.

First, assume that at least one of the inequalities $n \ge 7s/6$ and $s \le 7$ holds. Since $s\left(\frac{s-1}{s}\right)^{n+1} \ge 0$, it follows that $\psi(n,s) = \frac{n-\frac{s}{2}+\frac{1}{2}}{s\left(\frac{s-1}{s}\right)^{n+1}+n+1-s} \le \frac{n-\frac{s}{2}+\frac{1}{2}}{n+1-s}$, and to prove Lemma 10 it suffices to show that $n - s/2 + 1/2 \le 4(n+1-s)$, which is equivalent to $7s \le 6n+7$. If $n \ge 7s/6$, then $7s \le 6n < 6n+7$, while if $s \le 7$, then due to $s \le n$ we have $6s \le 6n$, proving that $7s \le 6n+7$ as required.

We are left with the case where $s > 7$ and $n < 7s/6$. We prove Lemma 10 by providing a lower bound on $(\frac{s-1}{s})^{n+1}$. Since $\frac{s-1}{s} < 1$, we have that $(\frac{s-1}{s})^{n+1} > (\frac{s-1}{s})^{7s/6+1}$. The function $(\frac{s-1}{s})^{s-1}$ is monotonically nonincreasing, and $\lim_{s\to\infty}(\frac{s-1}{s})^{s-1} = \frac{1}{e} \approx 0.3678$. Thus $(\frac{s-1}{s})^{s-1} \ge 0.367$. Thus $(\frac{s-1}{s})^{7(s-1)/6} \ge 0.31$, and using $(\frac{s-1}{s})^{13/6} \ge 0.7$, which holds by $s \ge 7$, we have $(\frac{s-1}{s})^{7s/6+1} = (\frac{s-1}{s})^{7(s-1)/6+13/6} \ge 0.21 > 1/5$. It is sufficient to show $\frac{n-s/2+1/2}{s/5+n+1-s} \le 4$, or alternatively, $27s \le 30n+35$, which holds as $s \le n$. ∎

Now we are able to establish a tight bound on the power of preemption.

THEOREM 2. *Let $\mu_0 \approx 0.7959\ldots$ be the solution of $2e^{-\frac{1}{\mu}} - \mu + \mu e^{-\frac{1}{\mu}} = 0$. Define $R(\mu_0) = \frac{1-\frac{\mu_0}{2}}{\mu_0 e^{-\frac{1}{\mu_0}}+1-\mu_0} \approx 1.39795\ldots$. Then the inequality*

$$\rho = \sup \frac{\Phi\left(S^*_{np}\right)}{\Phi\left(S^*_{p}\right)} \le R(\mu_0)$$

holds, i.e., $R(\mu_0)$ is an upper bound on the power of preemption, and moreover, this bound is tight.

Proof. For an integer n, let $c_n = \sup_{I \in \mathcal{I}_n} \frac{\Phi\left(S^*_{np}(I)\right)}{\Phi\left(S^*_p(I)\right)}$, where $\mathcal{I}_n$ is the set of inputs consisting of n jobs. We have $\rho = \sup_{n \geq 1} c_n$. Using Lemma 2, we also have $\rho = \limsup_{n \to \infty} c_n$. We have shown in a sequence of lemmas that it is sufficient to consider inputs with unit-size jobs, and moreover, for every n, it is sufficient to consider good inputs with m machines (and n unit-sized jobs). That is, by Lemma 2, the supremum for ρ is achieved by a tight sequence of good inputs, with equal numbers of jobs and machines per input, where the number of unit-sized jobs, n, grows to infinity.

Since the worst-case for ρ is achieved by a tight sequence where the number of unit-sized jobs, n, grows to infinity, consider such a sequence of instances. In a tight sequence, an instance, which is a good input satisfying the properties in Theorem 1, is characterized by a pair (n, s), where n is the number of jobs and machines, while $s \leq n$ is the speed of the faster machine. Consider all pairs (n, s) along a tight sequence. Since $1 \leq s \leq n$, it follows that the sequence of values $\frac{s}{n}$ is bounded. By Lemma 10, the sequence of values ψ_n is also bounded. Thus, there is a subsequence of indices (of n) such that for this subsequence both $\frac{s}{n}$ and ψ_n are converging, and consider the sequence of instances corresponding to this subsequence of values of n. Observe that the instances related to this subsequence form a tight sequence as well, and the number of jobs in these instances grows to infinity. Let μ be the limit of $\frac{s}{n}$ when n grows to infinity (along this subsequence).

Thus, we deduce

$$\rho = \limsup_{n \to \infty} c_n \leq \limsup_{n \to \infty} \psi_n$$

$$\leq \sup_{0 \leq \mu \leq 1} \lim_{n \to \infty} \frac{n - \frac{\mu n}{2} + \frac{1}{2}}{(\mu n)\left(\frac{\mu n - 1}{\mu n}\right)^{n+1} + n + 1 - \mu n} = \sup_{0 \leq \mu \leq 1} \frac{1 - \frac{\mu}{2}}{\mu e^{-\frac{1}{\mu}} - \mu + 1}.$$

The derivative of function

$$R(\mu) = \frac{1 - \frac{\mu}{2}}{\mu e^{-\frac{1}{\mu}} - \mu + 1}$$

is equal to

$$\frac{2e^{-\frac{1}{\mu}} - \mu + \mu e^{-\frac{1}{\mu}}}{2\mu^3 e^{-\frac{2}{\mu}} - 4\mu^3 e^{-\frac{1}{\mu}} + 4\mu^2 e^{-\frac{1}{\mu}} + 2\mu^3 - 4\mu^2 + 2\mu},$$

so that $R(\mu)$ reaches its maximum at a stationary point, which is the solution of the equation $2e^{-\frac{1}{\mu}} - \mu + \mu e^{-\frac{1}{\mu}} = 0$. Numerically, such a solution μ_0 is approximately equal to $0.7959\ldots$, which gives an upper bound $R(\mu_0)$ on the power of preemption approximately equal to $1.39795\ldots$.

To see that $R(\mu_0)$ is also a lower bound on the power of preemption, we can exhibit a tight sequence such that instance I_ℓ is associated with a pair of integers (n_ℓ, s_ℓ). Instance I_ℓ is a good input that contains n_ℓ unit-sized jobs and n_ℓ machines, such that the speed of the fast machine is s_ℓ, $1 < s_\ell \leq n$, while the speed of each remaining machine is equal to 1. Moreover,

$$\lim_{\ell \to \infty} s_\ell = +\infty, \quad \lim_{\ell \to \infty} n_\ell = +\infty, \quad \lim_{\ell \to \infty} \frac{s_\ell}{n_\ell} = \mu_0.$$

For the corresponding sequence of instances the sequence of cost ratios converges to $R(\mu_0)$. ∎

6. The power of preemption for two machines. The upper bound on the power of preemption established in Theorem 2 is a global bound that holds for all instances of the problem of minimizing the total completion time on uniformly related machines. This power of preemption is achieved as a limit for instances with huge numbers of jobs and machines. However, for a fixed number of machines a smaller bound can be derived, as shown below for the case of $m = 2$.

THEOREM 3. *In the case of two machines,*

$$\frac{\Phi\left(S_{np}^{*}\right)}{\Phi\left(S_{p}^{*}\right)} = \rho_2 \leq \frac{6}{5},$$

and this bound is tight.

Proof. If for some instance I

$$\frac{\Phi\left(S_{np}^{*}(I)\right)}{\Phi\left(S_{p}^{*}(I)\right)} = \rho_2,$$

then, as established in section 4, we may assume that I is a good input. Consider such an input that consists of n unit-sized jobs. Assume that machine M_1 has speed s, $1 < s \leq n$, while the speed of machine M_2 is 1. For a schedule S, let $C_k(S)$, $1 \leq k \leq n$, be a nondecreasing sequence of the completion times in S.

Recall that in an optimal nonpreemptive schedule, for a good input at least one slow machine is not assigned any jobs. Thus, in our case, in schedule $S_{np}^{*}(I)$ all jobs are processed on the fast machine M_1, so that

$$C_k\left(S_{np}^{*}\right) = \frac{k}{s}, \quad 1 \leq k \leq n. \tag{20}$$

Notice that since the jobs are unit-sized, Algorithm QSumNP associates the multiplier $\frac{k}{s}$ with the job to be scheduled in the kth position on machine M_1. In particular, in order to assign the last job to machine M_1, the current multiplier $\frac{n}{s}$ on machine M_1 cannot be larger than 1, so we have $n \leq s$. This, together with the condition $s \leq n$, implies that $s = n$ in instance I. Therefore, since $\sum_{k=1}^{n} \frac{k}{s} = \frac{n(n+1)}{2s}$, it follows from $s = n$ that

$$\Phi\left(S_{np}^{*}\right) = \frac{n+1}{2}. \tag{21}$$

Recall that in any preemptive schedule, the makespan, i.e., the completion time of the last job, cannot be smaller than the ratio of all processing times to the sum of speeds, which holds by simple averaging; see (6). This implies (when applied for a prefix of k jobs of the job sequence) that

$$C_k\left(S_{p}^{*}\right) \geq \frac{k}{s+1}, \quad 1 \leq k \leq n.$$

First, assume that $n \geq 5$. Then due to $s = n$, the inequality

$$\frac{C_k\left(S_{np}^{*}\right)}{C_k\left(S_{p}^{*}\right)} \leq \frac{s+1}{s} = \frac{n+1}{n} \leq \frac{6}{5} \tag{22}$$

holds for each k, $1 \le k \le n$, and the cost ratio is therefore no larger than $\frac{6}{5}$ for $n \ge 5$.

We now prove the upper bound for small values of n, $1 \le n \le 4$. If $n = 1$, then obviously $\Phi\left(S_p^*\right) = C_1\left(S_p^*\right) = \Phi\left(S_{np}^*\right) = C_1\left(S_{np}^*\right) = 1/s$. For each value of $n \in \{2, 3, 4\}$, we compute and analyze the ratios

$$F_n = \frac{\sum_{k=1}^{n} C_k\left(S_{np}^*\right)}{\sum_{k=1}^{n} C_k\left(S_p^*\right)}$$

and show that none of them exceeds $\frac{6}{5}$. By direct computation, we deduce that

$$C_2\left(S_p^*\right) = \frac{2}{s} - \frac{1}{s^2}\ , \quad C_3\left(S_p^*\right) = \frac{3}{s} - \frac{2}{s^2} + \frac{1}{s^3}, \quad C_4\left(S_p^*\right) = \frac{4}{s} - \frac{3}{s^2} + \frac{2}{s^3} - \frac{1}{s^4}.$$

Thus, $C_1\left(S_p^*\right) + C_2\left(S_p^*\right) = \frac{3}{s} - \frac{1}{s^2}$, $C_1\left(S_p^*\right) + C_2\left(S_p^*\right) + C_3\left(S_p^*\right) = \frac{6}{s} - \frac{3}{s^2} + \frac{1}{s^3}$, and

$$C_1\left(S_p^*\right) + C_2\left(S_p^*\right) + C_3\left(S_p^*\right) + C_4\left(S_p^*\right) = \frac{10}{s} - \frac{6}{s^2} + \frac{3}{s^3} - \frac{1}{s^4}\ .$$

Applying (21) and the above expressions with $s = n$, we obtain

$$F_2 = \frac{\frac{3}{2}}{\frac{1}{s^2}\left(3s-1\right)} = \frac{\frac{3}{2}}{\frac{5}{4}} = \frac{6}{5},$$

$$F_3 = \frac{\frac{4}{2}}{\frac{1}{s^3}\left(6s^2-3s+1\right)} = \frac{2}{\frac{1}{27}\left(54-9+1\right)} = \frac{27}{23} < \frac{6}{5},$$

$$F_4 = \frac{\frac{5}{2}}{\frac{1}{s^4}\left(10s^3-6s^2+3s-1\right)} = \frac{5/2}{\frac{1}{256}\left(640-96+12-1\right)} = \frac{128}{111} < \frac{6}{5},$$

as required.

To complete the proof, we show that $\frac{6}{5}$ is a tight bound on the power of preemption for $m = 2$. To see this, consider an instance with two unit-sized jobs and $s = 2$. We have that $C_1\left(S_{np}^*\right) = C_1\left(S_p^*\right) = 1/2$, $C_2\left(S_{np}^*\right) = 1$, and $C_2\left(S_p^*\right) = 3/4$, so that $\Phi\left(S_{np}^*\right) = 3/2$ and $\Phi\left(S_p^*\right) = 5/4$, leading to the cost ratio of $\Phi\left(S_{np}^*\right)/\Phi\left(S_p^*\right) = \frac{6}{5}$. ∎

REFERENCES

[1] O. Braun and G. Schmidt, *Parallel processor scheduling with limited number of preemptions,* SIAM J. Comput., 32 (2003), pp. 671–680, https://doi.org/10.1137/S0097539702410697.

[2] P. Brucker, *Scheduling Algorithms*, 5th ed., Springer-Verlag, Berlin, 2007.

[3] J. L. Bruno, E. G. Coffman Jr., and R. Sethi, *Scheduling independent tasks to reduce mean finishing time,* Comm ACM, 17 (1974), pp. 382–387, https://doi.org/10.1145/361011.361064.

[4] J. L. Bruno and T. Gonzalez, *Scheduling Independent Tasks with Release Dates and Due Dates on Parallel Machines,* Technical Report 213, Computer Science Department, Pennsylvania State University, State College, PA, 1976.

[5] B. Chen, *Parallel Machine Scheduling for Early Completion,* in Handbook of Scheduling: Algorithms, Models and Performance Analysis, J. Y.-T. Leung, ed., Chapman & Hall/CRC, London, 2004, pp. 9-175–9-184.

[6] R. W. Conway, W. L. Maxwell, and L. W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, MA, 1967.

[7] J. R. Correa, M. Skutella, and J. Verschae, *The power of preemption on unrelated machines and applications to scheduling orders,* Math. Oper. Res., 37 (2012), pp. 379–398, https://doi.org/10.1287/moor.1110.0520.

[8] T. F. Gonzalez, *Optimal Mean Finish Time Preemptive Schedules,* Technical Report 220, Computer Science Department, Pennsylvania State University, State College, PA, 1977.

[9] T. F. Gonzalez and S. Sahni, *Preemptive scheduling of uniform processor systems,* J. Assoc. Comput. Mach., 25 (1978), pp. 92–101, https://doi.org/10.1145/322047.322055.

[10] W.A. Horn, *Minimizing average flow time with parallel machines,* Oper. Res., 21 (1973), pp. 846–847, https://doi.org/10.1287/opre.21.3.846.

[11] E. Horowitz and S. Sahni, *Exact and approximate algorithms for scheduling nonidentical processors,* J. Assoc. Comput. Mach., 23 (1976), pp. 317–327, https://doi.org/10.1145/321941.321951.

[12] Y. Jiang, Z. Weng, and J. Hu, *Algorithms with limited number of preemptions for scheduling on parallel machines,* J. Comb. Optim., 27 (2014), pp. 711–723, https://doi.org/10.1007/s10878-012-9545-0.

[13] J. Labetoulle, E. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, *Preemptive Scheduling of Uniform Machines Subject to Release Dates,* in Progress in Combinatorial Optimization, Academic Press, Toronto, ON, 1984, pp. 245–261.

[14] E. L. Lawler and J. Labetoulle, *On preemptive scheduling of unrelated parallel processors by linear programming*, J. Assoc. Comput. Mach., 25 (1978), pp. 612–619, https://doi.org/10.1145/322092.322101.

[15] C.-Y. Lee and V. A. Strusevich, *Two-machine shop scheduling with an uncapacitated interstage transporter,* IIE Trans., 37 (2005), pp. 725–736, https://doi.org/10.1080/07408170590918290.

[16] J.-H. Lin and J. S. Vitter, *$\varepsilon-$Approximations with minimum packing constraint violation*, in Proceedings of the 24th Annual ACM Symposium on Theory of Computing (STOC '92), ACM, New York, 1992, pp. 771–782, https://doi.org/10.1145/129712.129787.

[17] R. McNaughton, *Scheduling with deadlines and loss functions,* Management Sci., 6 (1959), pp. 1–12, https://doi.org/10.1287/mnsc.6.1.1.

[18] M. H. Rothkopf, *Scheduling independent tasks on independent processors*, Management Sci., 12 (1966) pp. 437–447, https://doi.org/10.1287/mnsc.12.5.437.

[19] K. Rustogi and V. A. Strusevich, *Parallel machine scheduling: Impact of adding extra machines,* Oper. Res., 61 (2013), pp. 1243–1257, https://doi.org/10.1287/opre.2013.1208.

[20] A. S. Schulz and M. Skutella, *Scheduling unrelated machines by randomized rounding*, SIAM J. Discrete Math., 15 (2002), pp. 450–469, https://doi.org/10.1137/S0895480199357078.

[21] R. A. Sitters, *Two NP-hardness results for preemptive minsum scheduling of unrelated parallel machines*, in Proceedings of the 8th IPCO Conference, Lecture Notes in Comput. Sci. 2081, Springer-Verlag London, 2001, pp. 396–405.

[22] R. A. Sitters, *Approximability of average completion time scheduling on unrelated parallel machines*, in ESA 2008, D. Halperin and K. Mehlhorn, eds., Lecture Notes in Comput. Sci. 5193, Springer, Berlin, Heidelberg, 2008, pp. 768–779, https://doi.org/10.1007/978-3-540-87744-8_64.

[23] A. J. Soper and V. A. Strusevich, *Single parameter analysis of power of preemption on two and three uniform machines*, Discrete Optim., 12 (2014), pp. 26–46, https://doi.org/10.1016/j.disopt.2013.12.004.

[24] A. J. Soper and V. A. Strusevich, *Power of preemption on uniform parallel machines,* in APPROX/RANDOM 2014, Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, K. Jansen et al., eds., Leibniz Int. Proc. Informat. (LIPIcs), 28, Leibniz-Zent. Inform., Wadern, Germany, 2014, pp. 392–402, https://doi.org/10.4230/LIPIcs.APPROX-RANDOM.2014.392.

[25] G. J. Woeginger, *A comment on scheduling on uniform machines under chain-like precedence constraints*, Oper. Res. Lett., 26 (2000), pp. 107–109, https://doi.org/10.1016/S0167-6377(99)00076-0.