NEW TIGHT NP-HARDNESS OF PREEMPTIVE MULTIPROCESSOR AND OPEN SHOP SCHEDULING

Evgeny Shchepin  
Steklov Math. Inst., 117966  
Gubkina 8, Moscow, Russia  
scepin@mian.ras.su

Nodari Vakhania  
Science Faculty, State University of Morelos  
Av. Universidad 1001, Cuernavaca 62210, Morelos Mexico.  
Inst. Computational Math., Akuri 8, Tbilisi 93, Georgia  
nodari@servm.fc.uaem.mx

Abstract  We show that preemptive versions of some NP-hard scheduling problems remain NP-hard, but only for a restricted number of preemptions. If we allow a “sufficient” number of preemptions, then these problems become polynomially solvable. We find, as we call, the critical number of preemptions for our problems, i.e., the minimal number of preemptions for which they become polynomially solvable. First we establish that the critical number of preemptions for scheduling $m$ identical processors is $m - 1$ (we show that scheduling identical processors with at most $m - 2$ preemptions is NP-hard). Then we consider so-called acyclic open-shop scheduling problem with $m$ machines, a strongly restricted version of the classical open-shop scheduling, and show that $m - 2$ is the critical number of preemptions for this problem (the earlier known related well-known result was that non-preemptive open shop scheduling is NP-hard). Finally, we consider a slightly restricted version of scheduling on $m$ unrelated processors. The restriction is that the processing time of any job on any processor is no more than the optimal schedule makespan. We call such processors non-lazy unrelated processors. We show that $2m - 3$ is the critical number of preemptions for scheduling non-lazy unrelated processors.

Keywords: open-shop scheduling, multiprocessor scheduling, preemption, algorithm, NP-hard
Introduction

Two types of scheduling problems distinguished in the literature are preemptive and non-preemptive ones. It is quite typical that a non-preemptive scheduling problem $\mathcal{P}$ is NP-hard whereas its preemptive version $\mathcal{P}_{pmtn}$ is polynomially solvable. Let us call a scheduling problem $\pi$-preemptive if at most $\pi$ preemptions are allowed in it, $\pi$ being any non-negative integer. A question which naturally arises is whether $k$-preemptive and $l$-preemptive problems, for some non-negative integers $k$ and $l$, have the same complexity. If a $0$-preemptive (non-preemptive) problem is NP-hard, does the corresponding $k$-preemptive problem, for $k = 1, 2, ..., \pi$, remain NP-hard? For many scheduling problems this is not the case, i.e., there is some positive integer $\pi$ such that corresponding $k$-preemptive versions, for $k \geq \pi$, are polynomially solvable. Then what is the minimal $\pi$ such that the $\pi$-preemptive version is polynomially solvable (we call such $\pi$ the critical number of preemptions for the problem)?

Traditionally, a preemptive scheduling problem implies an arbitrary number of preemptions: a $k$-preemptive problem for a fixed integer $k > 0$ would not be treated as a preemptive scheduling problem (unless $k$ is “sufficiently” large magnitude not known in advance and changing from problem to problem). So though a $k$-preemptive problem, $k > 0$, is not a non-preemptive problem, it might be a “preemptive problem”. We could refine the rough classification of scheduling problems into preemptive or non-preemptive ones by specifying the maximal number of preemptions which is allowed for a particular problem, i.e., consider $\pi$-preemptive scheduling problems.

A polynomial-time algorithm for a preemptive problem $\mathcal{P}_{pmtn}$ imposes a certain number of preemptions, the maximal number of which can be usually estimated. In most of the real-life problems the number of preemptions is a crucial factor which has to be made as small as possible since each preemption implies additional communication cost and may also yield a forced job migration. How small the overall number of preemptions can be made, i.e., what is the critical number of preemptions for $\mathcal{P}_{pmtn}$? This kind of question has been also posed by Shachnai et al. [7] and the critical number of preemptions $2m - 1$ for scheduling uniform machines $Q_{pmtn}/C_{\text{max}}$ was established (an $O(n + m \log m)$ algorithm by Gonzalez and Sahni [3] yields at most $(2m - 2)$ preemptions). In this paper, we first establish that $m - 1$ is the critical number of preemptions for scheduling identical machines $P_{pmtn}/C_{\text{max}}$. We show that $(m - 2)$-preemptive scheduling on identical processors $P_{pmtn}(m - 2)/C_{\text{max}}$ is NP-hard (Section 3), whereas a venerable linear-time algorithm by McNaughton [6] yields $m - 1$ preemptions.

Multiprocessor and open-shop scheduling problems are intimately related with the so-called acyclic distributions (see Shchepin & Vakhania [8]–[11]). An optimal schedule can be obtained in two stages. On the first stage an op-
optimal distribution is constructed (a distribution assigns jobs or their parts to machines without specifying the start times). Such a distribution can be constructed in polynomial time by linear programming. The preemptions in these distributions can be represented by the so-called preemption graphs: in these graphs nodes represent machines and if two nodes are joined by an edge, then the corresponding machines share the same job which labels this edge. An optimal distribution, obtained by linear programming, has at most \( m - 1 \) preemptions or equivalently, its preemption graph is acyclic. A distribution or an open-shop (which can be seen as a distribution) is acyclic, if its preemption graph is acyclic. As an example, in an acyclic open-shop no two jobs may have two non-dummy operations on the same two machines. Scheduling acyclic distributions turned out to be an efficient tool for the exact solution of \( R/pmtn/C_{\text{max}} \) (see Lawler and Labetoulle [4]) and an approximate solution of its non-preemptive version \( R/C_{\text{max}} \) (see Lenstra et al. [5] and Shchepin & Vakhania [10]). Acyclic shop scheduling problems are also interesting from the other point of view: they can represent “maximal” polynomially solvable cases of the corresponding non-acyclic versions Shchepin & Vakhania [11].

The second problem we deal with is an acyclic \((m - 3)\)-preemptive open-shop scheduling \( O/acyclic,\ pmtn(m - 3)/C_{\text{max}} \). General preemptive open-shop problem \( O/pmtn/C_{\text{max}} \) is well-known to be solvable in polynomial (no worse than \( O(n^4) \)) time, and its non-preemptive version \( O/C_{\text{max}} \) is NP-hard due to the early classical results by Gonzales & Sahni [2]. We show that \((m - 3)\)-preemptive open-shop scheduling, even its acyclic version \( O/acyclic,\ pmtn(m - 3)/C_{\text{max}} \), remains NP-hard (Section 4). Moreover, \( m - 2 \) is the critical number of preemptions due to the recent linear-time algorithm by Shchepin & Vakhania [11] for \( O/acyclic,\ pmtn(m - 2)/C_{\text{max}} \).

Scheduling \( m \) unrelated processors \( R/pmtn/C_{\text{max}} \) is much more difficult than scheduling \( m \) identical processors. No polynomial algorithm with a performance ratio less than 1.5 can exist unless \( P = NP \), see Lenstra et al. [5] (currently best known ratio is \( 2 - 1/m \), see Shchepin & Vakhania [10]). However, it turned out that a slightly restricted version of \( R/pmtn/C_{\text{max}} \) can be efficiently solved. In particular, we consider, as we call, non-lazy unrelated processors, in which the processing time of any job on any machine is no more than the optimal schedule makespan \( C^*_{\text{max}} \); i.e., there is no machine which is too slow for some job (we abbreviate this problem by \( R/p_{ij} \leq C^*_{\text{max}}/C_{\text{max}} \) adopting the standard notation for scheduling problems). Not only \( R/p_{ij} \leq C^*_{\text{max}}/pmtn/C_{\text{max}} \) is polynomially solvable, but its \((2m - 3)\)-preemptive version is polynomially solvable (Section 6), whereas we show that the \((2m - 4)\)-preemptive version is NP-hard (Section 5). Thus \( 2m - 3 \) is the critical number of preemptions for \( R/p_{ij} \leq C^*_{\text{max}}/pmtn/C_{\text{max}} \). We can guarantee only a near-optimal \((2m - 3)\)-preemptive schedule for general unrelated processors. The makespan of such a schedule is no more than either
the corresponding non-preemptive schedule makespan or \( \max \{ C^*_\text{max}, p_{\text{max}} \} \), where \( p_{\text{max}} \) is the maximal job processing time (Section 6). These results are obtained relatively easily from the earlier results from Shchepin & Vakhania [9].

1. Summary of notions and notations

This section contains glossary of notions and notations used further in this paper. The reader may choose to have a brief look on it or skip it at all now and return to it later upon necessity. Not all the introduced notations are widely used in the literature (however, we do find them convenient).

**Jobs and machines.** We will be dealing with \( n \) jobs \( J = \{ J_1, \ldots J_n \} \) and \( m \) machines (processors) \( M = \{ M_1, \ldots M_m \} \). The processing time (length) of job \( J \) on machine \( M \) will be denoted by \( M(J) \). The maximal job processing time will be denoted by \( p_{\text{max}} \).

**Distributions and assignments.** A distribution \( \delta \) of \( n \) jobs from \( J \) on \( m \) machines from \( M \) is a mapping \( \delta: J \times M \rightarrow R^+ \), such that \( \sum_{M \in M} \delta(J, M) = 1 \), for all \( J \in J \). The processing time (or length) of job \( J \) on a machine \( M \) in distribution \( \delta \) is \( |J|_M^\delta = \delta(J, M)M(J) \). The processing time (or length) of a job \( J \) in a distribution \( \delta \) is \( |J|^{\delta} = \sum_{M \in M} |J|_M^\delta \). The maximal job length in \( \delta \) is denoted by \( |\delta|_{\text{max}} \).

\( \sum_{J \in J} |J|_M^\delta \) is called the load of machine \( M \) in distribution \( \delta \) and is denoted by \( |M|_{\delta} \). The maximal machine load in \( \delta \) or the makespan is denoted by \( |\delta|_{\text{max}} \).

A distribution \( \delta \) for \( J, M \) with the minimal \( |\delta|_{\text{max}} \) is called an optimal distribution. A uniform distribution is a distribution, in which all machine loads are equal. The total processing time in a distribution \( \delta \) is the sum of all machine loads in this distribution.

The sequential makespan of \( \delta \), \( ||\delta|| = \max\{ |\delta|_{\text{max}}, |\delta|^{\text{max}} \} \). It is clear that \( ||\delta|| \) is a lower bound on the makespan of any feasible schedule for the corresponding open-shop problem; Gonzales & Sahni [2] (see also Lawler & LaBerteouille [4]) have shown that \( ||\delta|| \) is achievable in a feasible schedule associated with distribution \( \delta \) (a formal definition of a feasible schedule associated with \( \delta \) is given a bit later in this section).

An assignment of jobs of \( J \) on machines of \( M \) is a binary relation on \( J \times M \). Any distribution \( \delta \) of \( J \) on \( M \) generates an assignment \( \{(J, M) \mid \delta(J, M) > 0 \} \). We will not use any special letter for an assignment, instead, we will use \( \delta(J) \) for \( \{M \in M \mid \delta(J, M) > 0 \} \).

A distribution \( \delta \) is non-preemptive if \( \delta(J, M) \) takes value 0 or 1 for any \( J, M \). The number of preemptions of job \( J \) in \( \delta \) is \( \text{pr}_\delta(J) = |\delta(J)| - 1 \). \( \text{pr}(\delta) = \sum_{J \in J} \text{pr}_\delta(J) \) is the (total) number of preemption in \( \delta \).
Multiprocessors. A multiprocessor is a triple constituted by the sets of jobs \( J \) and machines \( M \) and a processing time function \( f \), a mapping from \( J \times M \) to \( R^+ \), where the value of this function for a pair \( J, M \) is the processing requirement of \( J \) on \( M \). As mentioned above, we use \( M(J) \) for the value of this function for the pair \( J, M \). For the notational simplicity, a multiprocessor will be denoted by \( J, M \) (we omit \( f \)).

A multiprocessor without any restriction on its processing time function is called a system of unrelated processors. In a system of identical processors, for each \( P \) and \( Q \) from \( M \) and for each \( J \in J \), \( P(J) = Q(J) \).

Schedules. A schedule indicates which job is in process on each machine at each time moment. Formally, a schedule \( \sigma \) is a subset of \( J \times M \times [0, T) \), for some \( T \geq 0 \). \((J, M, t) \in \sigma \) signifies that job \( J \) is processed by machine \( M \) at the moment \( t \) in \( \sigma \). \( T \) is called the makespan of \( \sigma \) and is denoted by \( ||\sigma|| \).

For a pair \( J, M \in J \times M \), \( \sigma(J, M) = \{ t \in R^+ \mid (J, M, t) \in \sigma \} \); \( \sigma(J, M) \) may contain a single interval, or it can be a multi-interval, i.e., a union of two or more intervals. Each interval is assumed to be a left half-interval. The total length of \( \sigma(J, M) \), \( |\sigma(J, M)| \), is the processing time of \( J \) on \( M \) in \( \sigma \).

A feasible schedule \( \sigma \) must satisfy the following conditions:

1. For every \( M \in M \), the set of jobs \( \sigma(M, t) = \{ J \in J \mid (J, M, t) \in \sigma \} \) contains at most one element;
2. For every \( J \in J \), the set of machines \( \sigma(J, t) = \{ M \in M \mid (J, M, t) \in \sigma \} \) contains at most one element;
3. For every \( J \in J \), \( \sum_{M \in M} \frac{|\sigma(J, M)|}{M(J)} = 1 \);

(1) provides that each machine processes at most one job at any time; (2) provides that each job is processed by at most one machine at any time; (3) provides that each job is completely processed.

A feasible schedule with the minimal makespan is called optimal.

With each distribution is associated an (infinite) set of feasible schedules with this distribution. Conversely, with each feasible schedule \( \sigma \) is associated a distribution \( \delta_\sigma \) defined as follows:

\[
\delta_\sigma(J, M) = \frac{|\sigma(J, M)|}{M(J)}.
\]

A machine is idle at some moment (interval) if it handles no job at that moment (interval). We call a feasible schedule \( \sigma \) tight, if no machine is idle from time 0 to its completion time (that is, the completion time of the latest job scheduled on that machine) in \( \sigma \). A tight non-preemptive schedule, associated with a distribution \( \delta \) can be uniquely given by indicating a sequence of jobs for each machine. The makespan of this (not necessarily feasible) schedule is \( |\delta|_{\text{max}} \).

Switching points, components, splittings and preemptions. Let us define a component of a schedule as the maximal (continuous) time interval, dur-
ing which a machine processes a unique job. A schedule can be completely given by all its components. Formally, a component of a schedule $\sigma$ is a triple $(J, M, [p, q])$, where $J$ is a job, $M$ is a machine and $[p, q]$ is a time interval which is a connectivity component in $\sigma(J, M)$. Further on we will refer to a component $(J, M, [p, q])$ as a $J$-component of $\sigma$ on $M$ or a $(J, M)$-component of $\sigma$.

A boundary point of the set $\sigma(J, M)$ is called a switching point of $J$ on $M$ or a $J$-switching point on $M$. At such a point $M$ interrupts the processing of $J$ or (re)starts its processing.

We will say that a job $J$ is split on machine $M$, if $\sigma(J, M)$ is a multi-interval, i.e., it consists of two or more $J$-components. The number of splittings of job $J$ on machine $M$ in $\sigma$, $\text{sp}(\sigma(J, M))$ is the number or components in $\sigma(J, M)$ minus 1. $\text{sp}(\sigma(M)) = \sum_{J \in \mathcal{J}} \text{sp}(\sigma(J, M))$ is the number of splittings in $\sigma$ on $M$; $\text{sp}(\sigma) = \sum_{M \in \mathcal{M}} \text{sp}(\sigma(M))$ is the total number of splittings in $\sigma$.

The number of preemptions in $\sigma$, $\text{pr}(\sigma) = \text{sp}(\sigma) + \text{pr}(\delta_{\sigma})$: a preemption in $\sigma$ may come either from $\delta_{\sigma}$, or it might be a splitting.

**Preemption and assignment graphs.** A (full) preemption graph of a distribution $\delta$ is a labelled graph, where the nodes correspond to the machines of $\mathcal{M}$ and the edges correspond to the jobs of $\mathcal{J}$. In particular, there is an edge $(M_i, M_j)$ between nodes $M_i$ and $M_j$ labelled by job $J$, iff both $\delta(J, M_i)$ and $\delta(J, M_j)$ are positive. The reduced preemption graph of $\delta$ or the preemption graph for short, is a subgraph $G(\delta)$ of the full preemption graph, in which all redundant edges are eliminated: an edge $(M_i, M_j)$ labelled by $J$ is redundant, if there exists $k$, $i < k < j$, such that $\delta(J, M_k) > 0$ (according to our machine indexing in $\mathcal{M}$). Any subgraph of the full preemption graph of $\delta$ constituted by all nodes representing machines sharing some job $J$ with all edges labelled by $J$, is a complete graph. The corresponding subgraph in the reduced preemption graph $G(\delta)$ is a simple path in $G(\delta)$. Note that to different enumerations in $\mathcal{M}$ different preemption graphs correspond. However, $G(\delta)$ is acyclic if and only if the preemption graph, corresponding to any enumeration in $\mathcal{M}$ is acyclic (Shchepin & Vakhania [8]). Other important properties are given in next propositions which immediately follow.

**Proposition 1** The number of preemptions in a distribution $\delta$ is equal to the number of edges in its preemption graph $G(\delta)$.

**Proposition 2** The preemption graph of a distribution $\delta$ is not connected iff the set of jobs $\mathcal{J}$ can be partitioned into subsets $\mathcal{J}_1 \cup \mathcal{J}_2 = \mathcal{J}$, such that $\delta(J_1) \cap \delta(J_2) = \emptyset$, for each $J_1 \in \mathcal{J}_1$ and $J_2 \in \mathcal{J}_2$.

A distribution $\delta$ is called acyclic if $G(\delta)$ is acyclic. For a given distribution $\delta$ and the corresponding assignment, an assignment graph is defined as a bipartite graph in which the two sets of nodes are formed by the sets of jobs $\mathcal{J}$ and
New tight NP-hardness of preemptive multiprocessor and open shop scheduling

machines $\mathcal{M}$, respectively, such that there is an edge $(J, M)$ iff $\delta(J, M) > 0$ (Lenstra et al. [5]). An acyclic assignment is an assignment with the acyclic assignment graph. There is an obvious similarity between preemptions and assignment graphs. The following proposition immediately follows from the definitions.

**Proposition 3** Two distributions with the same assignment have the same preemption graph. Hence, the assignment graph of any distribution is acyclic iff the preemption graph of this distribution is acyclic.

2. **NP-hardness of $P/\text{pmtn}(m - 2)/C_{\text{max}}$**

In this section we show that $P/\text{pmtn}(m - 2)/C_{\text{max}}$ is NP-hard. We use the reduction from the NP-complete PARTITION problem (see Garey & Johnson [1]) for the decision version of $P/\text{pmtn}(m - 2)/C_{\text{max}}$. In the PARTITION problem we are given a finite set of integer numbers $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ with $S = \sum_{i=1}^{n} c_i$. This decision problem gives a “yes” answer iff there exists a subset of $\mathcal{C}$ which sums up to $S/2$. Given an arbitrary instance of a PARTITION, let us define our scheduling instance with $n + 2m + 2$ jobs with the total length of $2m + \frac{1}{2^m}$ as follows.

There are $m$ pairs of the so-called big jobs denoted by $B_i^{\pm}$, $i = 1, \ldots, m$ with $|B_i^+| = 1 + 1/2^i$, and $|B_i^-| = 1 - 1/2^i$. So the total length of all big jobs is $2m$.

There are two median jobs denoted by $D$ and $D'$, with $|D| = \frac{3}{2^m} - \frac{5}{m2^{m+2}}$ and $|D'| = \frac{3}{m2^{m+2}}$. So total length of the two median jobs is $1 - \frac{1}{2^m}$. There are $n$ small jobs $C_i$ with $|C_i| = \frac{c_i}{m2^{m+2}}$. So the total length of small jobs is $\frac{1}{2^m}$.

This transformation is polynomial as the number of jobs is bounded by the polynomial in $n$ and $m$, and all magnitudes can be represented in binary encoding in $O(m)$ bits.

Now we prove that there exists a feasible schedule with less than $m - 1$ preemptions and with the optimal makespan $2 + \frac{1}{m2^m}$ iff there exists a solution to our PARTITION. In one direction, suppose $\sum_{i=1}^{k} c_i = S/2$, for some $k < n$, i.e. we have a solution to the PARTITION. Then we define a tight schedule $\sigma$ with the makespan $2 + \frac{1}{m2^m}$ as follows. The job sequence on $M_1$ is: $B_1^-, B_1^+, D', C_1, \ldots, C_k$: so the completion time of $M_1$ is $2 + \frac{3}{m2^{m+2}} + \frac{1}{m2^{m+2}} = 2 + \frac{1}{m2^m}$ (the load of $M_1$). The job sequence on $M_2$.

---

1In this and the following section $n$ stands exclusively for the number of elements in PARTITION; in the rest of the paper $n$ denotes the number of jobs.
is: $B^-_2, D, C_{k+1}, \ldots, C_n, B^+_2$, where we have only a part of $D$ with the length $\frac{3}{m_2^{2m+\tau}} = \frac{3}{4} \frac{1}{m_2^{2m}}$ (providing that the load of $M_2$ is $2 + \frac{1}{m_2^{2m}}$). The rest of $D$ is divided into equal parts of the length $\frac{1}{m_2^{2m}}$ and is distributed on the machines $M_i, i > 2$. The schedule on $M_i, i > 2$ is tight and is generated by the sequence $B^-, D, B^+$.  

To see that $\sigma$ is feasible, we need to check the sequentiality of $\sigma$ on $D$, which is the only preempted job in $\sigma$. The completion time of $D$ on $M_i$ is no more than $1 - \frac{1}{2^i} + \frac{1}{m_2^{2m}} \leq 1 - \frac{1}{2^i} + \frac{1}{m_2^{2m+\tau}} \leq 1 - \frac{1}{2^i} + \frac{1}{2^i+\tau} = 1 - \frac{1}{2^i+\tau}$, and its starting time on $M_{i+1}$ is $1 - \frac{1}{2^i+\tau}$. Hence, $\sigma$ is sequential and has the makespan $2 + \frac{1}{m_2^{2m}}$. This completes the proof in one direction. We need the following lemma for the other direction:

**Lemma 1** If there exists a uniform distribution $\delta$ of jobs of $\mathcal{J}$ on $m$ identical processors from $\mathcal{M}$ with less than $m - 1$ preemptions, then there is a subset $\mathcal{J}'$ of $\mathcal{J}$ with the total length of $k|\delta|_{\text{max}},$ for some natural $k < m$.

**Proof.** Since $pr(\delta) < m - 1$, $G(\delta)$ has less than $m - 1$ edges and hence any its connected component contains $k < m$ machines (we may have two or more such components). Let $\mathcal{J}'$ be the set of all jobs distributed on machines in any of the connected components. Since $\delta$ is uniform, the total length of these jobs is $k|\delta|_{\text{max}}$.  

Assume now $\sigma$ is a tight $(m - 2)$-preemptive schedule for our scheduling instance. A distribution, generated by $\sigma$ also has no more than $m - 2$ preemptions and the load on all machines in this distribution is $2 + \frac{1}{m_2^{2m}}$. We will prove that this distribution already gives a solution to our instance of PARTITION.

Suppose $\mathcal{J}'$ is a subset of $\mathcal{J}$ with the total length of $|\mathcal{J}'| = 2k + k \frac{1}{m_2^{2m}}$ (see Lemma 1). First we note that $\mathcal{J}'$ contains exactly $2k$ big jobs. Indeed, $\mathcal{J}'$ cannot contain more than $2k$ big jobs, because the total length of the smallest $2k+1$ big jobs is $2k+1 - \sum_{i=1}^{k+1} \frac{1}{2^i} = 2k + \frac{1}{2^k+\tau} \geq 2k + \frac{1}{2^k} > 2k + k \frac{1}{m_2^{2m}} = |\mathcal{J}'|$. At the same time, the total length of the longest $2k - 1$ big jobs together with all median and small jobs is less than $2k$. Further, the total length $B'$ of all big jobs from $\mathcal{J}'$ is $2k$. Indeed if $B' - 2k$ is not 0, it must have an absolute value of at least $\frac{1}{2^k}$. If $B' < 2k$, then $B'$ plus the total length of all non-big jobs (which is no more than $\frac{1}{2^k}$) will be no more than $2k$. This contradicts our assumption that $|\mathcal{J}'| = 2k + k \frac{1}{m_2^{2m}}$. Similarly, if $B' > 2k$, then the above magnitude is at least $2k + \frac{1}{2^k} > 2k + k \frac{1}{m_2^{2m}} = |\mathcal{J}'|$, which again is a contradiction. Thus, $B' = 2k$.

It follows that the total length of the big jobs from $\mathcal{J}'^c$ is $2(m - k)$, $\mathcal{J}'^c$ being the complement of $\mathcal{J}'$ in $\mathcal{J}$, and hence without loss of generality, we can assume that $k \leq m/2$. In this case, $D$ cannot belong to $\mathcal{J}'$, because $|D| > k \frac{1}{m_2^{2m}}$. On the other hand, $D'$ must belong to $\mathcal{J}'$. Indeed, denote by $C'$
the total length of all small jobs from $J'$. If $J'$ does not contain a median job, then $|J'| = B' + C' \leq 2k + \frac{1}{m^{2m+2}}$, which contradicts our conjecture that $|J'| = 2k + C' + |D'|$. Therefore, $D' \in J'$. In this case $|J'| = B' + C' + |D'|$. This implies that $k\frac{1}{m^{2m}} = C' + \frac{3}{m^{2m+2}}$. But since $C' \leq \frac{1}{m^{2m+2}}$, the only possible value of $k$ is 1. Then $C' = \frac{1}{m^{2m}} - \frac{3}{2m+2m} = \frac{1}{m^{2m+2}}$, which is $S/2$ and we have obtained a solution to the PARTITION. We have proved this section’s main result:

**Theorem 2** \( P/{pmtn}(m - 2)/C_{max} \) is NP-hard.

### 3. The NP-hardness of \( O/acyclic, {pmtn}(m - 3)/C_{max} \)

An open shop $J, M$ consists of a set of $n$ jobs $J$ and a set of $m$ machines $M$. Each job $J^j$ consists of $m$ operations $\{J^j_i\}, i = 1, \ldots, m$. Each $J^j_i$ has to be processed by machine $M_i$ during $|J^j_i| \geq 0$ time units, the latter is called the length of $J^j_i$. Note that we allow $|J^j_i| = 0$; in this case we will say that $J^j_i$ is a dummy operation.

$|J^j| = \sum_{i=1}^{m} |J^j_i|$ is the length of job $J^j$. $|M_i| = \sum_{j=1}^{n} |J^j_i|$ is the load of machine $M_i$.

It is known that the optimal non-preemptive schedule makespan for \( O/{pmtn}/C_{max} \) is the maximum between the maximal machine load and the maximal job length (see Gonzalez & Sahni [2] and Lawler and Labetoulle [4]).

A distribution $\delta$ on $J, M$ defined as $\delta(J^j, M_i) = \frac{|J^j_i|}{|J^j|}$ can obviously be associated with each open shop. An acyclic open-shop is one with an acyclic distribution. A uniform open-shop is defined similarly as a uniform distribution.

In this section we prove the following theorem.

**Theorem 3** \( O/acyclic, {pmtn}(m - 3)/C_{max} \) is NP-hard.

The reduction from PARTITION is described in the next subsection.

### 3.1 The reduction from PARTITION

We transform the PARTITION problem from Section 3 to \( O/{pmtn}/C_{max} \). Let $C = \{c_1, c_2, \ldots, c_n\}$ and $S/2 = \sum_{i=1}^{n} c_i$, form again an arbitrary instance of a PARTITION. We define our open shop instance $O(C, m)$ as follows (assuming without loss of generality that $m \geq 3$). $O(C, m)$ deals with $1 + n(m - 2) + 2m - 2 = (n + 2)m - 2n - 1$ jobs and $m$ machines $\{M_i\}_{i=1}^{m}$.

We divide the set of jobs into the following three categories:
1) There is one **common** job $I$ to be distributed on all machines. The processing requirement of $I$ on $M_i$, $i = 1, 2, ..., m$, is 1. Hence, the total processing time of $I$ is $m$.

2) The jobs from the second category are called **partition jobs**. We introduce $n$ partition jobs for each of the machines, except the first one $M_1$ and the last one $M_m$ (we call these machines extremal, and the rest of machines intermediate). The $j$th, $j \leq n$, partition job to be distributed on machine $M_i$ ($1 < i < m$) is denoted by $P_{i,j}$. The processing time of $P_{i,j}$ is $c_j S_i$. Note that the total processing time of all partition jobs on each $M_i$ ($1 < i < m$) is equal to $1 S_i - 1 S_{i-1}$.

3) The jobs from the third category are called **fixers** and denoted by $F_1, F_2, F_3, \ldots, F_{m-1}, F_m$. As it is evident from this notation, there is only one fixer on machines $M_1$ and $M_m$, and there are two fixers on any intermediate machine. The processing time of the first fixer for machine $M_i$, $F_{i+1}$, $1 < i < m$, is equal to $(i - 1) - 1 S_i$, and the processing time of the second fixer $F_i$ is $(m - i) - 1 S_{i-1}$.

Similarly as in Section 3, our transformation is polynomial. Observe that all machine loads are $m$ and that the operations which we have not explicitly defined are dummy ones providing that $O(C, m)$ is acyclic. The following fact is also obvious:

**Lemma 4** Any tight schedule for $O(C, m)$ with the makespan $m$ is optimal.

Given a solution $\sum_{i \leq k} c_i = S/2$ to our PARTITION instance (we renumber elements in PARTITION correspondingly), a tight schedule for $O(C, m)$ without any splitting and with the optimal makespan $m$ (Lemma 4) can easily be built. On $M_1$ first is scheduled the fixer $F_1$ and then the common job $I$. On the last machine, first $I$ is scheduled and then $F_m$. On each intermediate machine $M_i$, jobs are scheduled in the following order: first the fixer $F_{i+1}$, then all $P_{i,j}$, $j \leq k$ (in any order), then job $I$ followed by the rest of the partition jobs $P_{i,j}$, $j > k$, and finally the second fixer $F_i$.

It will take essentially longer to show that a feasible schedule to our open-shop yields a solution to PARTITION. We need a number of auxiliary notions and lemmas. Let us say that $\sigma$ is a partitioning schedule if there is at least one intermediate machine $M_i$ in $\sigma$ and a time moment $t$, such that no partition job is executed at time $t$ and the total processing time of the partition jobs, completed by time $t$ is equal to the total processing time of the partition jobs, which are started at or after time $t$. We shall refer to $t$ as a partitioning time in $\sigma$.

**Lemma 5** Every feasible schedule for $O(C, m)$ having the makespan $m$ and at most $m - 3$ splittings is a partitioning schedule.
Assume for now that the above lemma is true and that $\sigma$ is a feasible schedule for $O(C, m)$, such that $||\sigma|| = m$ and $sp(\sigma) \leq m - 3$. By Lemma 5, $\sigma$ contains an intermediate machine with a switching point which is a partitioning time. Such a switching point can be found in linear time by checking all switching points, corresponding to the partition jobs on each intermediate machine. Clearly, any partitioning time gives a solution to the PARTITION problem. Thus Theorem 3 is proved under the assumption that Lemma 5 is true. The rest of this section is devoted to the proof of this lemma.

### 3.2 Schedule editing

In this subsection we introduce schedule editing operations which we use later on. It will be useful to note that if $\sigma$ is an optimal schedule for $O(C, m)$, then the common job $I$ has to be processed at any time moment in $\sigma$ (because the processing time of $I$ equals to the makespan of $\sigma$). Next, let us note that the number of switching points is closely related with the number of splittings. The following lemma gives an explicit formula which proof is straightforward:

**Lemma 6** The number of splittings in a tight schedule on a machine $M$ is equal to $s - n - 1$, where $s$ is the number of all switching points on $M$ and $n$ is the number of jobs scheduled on $M$.

The schedule editing operations are cutting, inserting and moving and are defined as follows.

**Cutting.** Let $p$ and $q$, $p < q$, be time moments. A schedule $\sigma'$ is obtained from a schedule $\sigma$ by the $(p, q)$-cutting on a machine $M_i$ if the following conditions hold:

1. $\sigma'(J, M) = \sigma(J, M)$ for all $M \neq M_i$ and all $J$;
2. $\sigma'(M_i, t) = \sigma(M_i, t)$ for $t < p$;
3. $\sigma'(M_i, t) = \sigma(M_i, t + q - p)$ for $t \geq p$.

Note that if $\sigma$ is tight, then $\sigma'$, obtained by the $(p, q)$-cutting on $M_i$, is also tight and the completion time of $M_i$ in $\sigma'$ is $p - q$ less than that in $\sigma$. Further, if $p$ and $q$ are switching points in $\sigma$, then the number of splittings in $\sigma'$ cannot increase. Moreover, this number will be decreased by 1 if $M_i$ performed the same job immediately before time $p$ and immediately after time $q$ in $\sigma$.

**Inserting.** A schedule $\sigma'$ is obtained from a schedule $\sigma$ by the $(p, q)$-inserting of a job $J$ on machine $M_i$ if the following conditions hold:

1. $\sigma'(J, M) = \sigma(J, M)$ for all $M \neq M_i$ and all $J$;
2. $\sigma'(M_i, t) = \sigma(M_i, t)$ for $t \leq p$;
3 $\sigma'(M_i, t) = J$ for $t \in [p, q)$;

4 $\sigma'(M_i, t) = \sigma(M_i, t - (q - p))$ for $t \geq q$.

Again, $\sigma'$ is tight if $\sigma$ is tight and the completion time of $M_i$ in $\sigma'$ is $q - p$ greater than that in $\sigma$. Further, if $p$ is a switching point in $\sigma$ on $M_i$, then the inserting cannot create a splitting of any job, different from the inserted job $J$; it yields a new splitting of job $J$ if $J$ in $\sigma$ was already scheduled on $M_i$ and $p$ is not a boundary of $\sigma(J, M_i)$.

**Moving.** We will use $(p, i, q)$ for the schedule component of a job $J$ on the machine $M_i$ with the time interval $[p, q)$. The $(p, q) \rightarrow (p', q')$-moving cuts a whole $J$-component $(p, i, q)$ of $\sigma$ from the interval $[p, q)$ and inserts it into the interval $[p', q')$ (of the same length) on the same machine $M_i$. Thus the moving is performed into two steps. The first step is the $(p, q)$-cutting on $M_i$; with the second step, in the obtained (after cutting) schedule the $(p', q')$-inserting of $J$ on $M_i$ is carried out.

The moving does not affect the tightness of $\sigma$ and the completion time of machine $M_i$ if $q' \leq |\sigma|_{M_i}$. Note that if $p > p'$ and $p'$ is a switching point in $\sigma$, then the $(p', q')$-inserting will not create a new splitting of any job, scheduled on $M_i$. Consequently, neither the $(p, q) \rightarrow (p', q')$-moving of a $J$-component will cause a new splitting. Moreover, if $M_i$ processed the same job immediately before time $p$ and immediately after time $q$ in $\sigma$, then the number of splittings in the resulted schedule $\sigma'$ will be one less than that in $\sigma$. Now in general, if $p$ is not a switching point, then the moving of a $J$-component can increase the number of splittings at most by 1. If $M_i$ is an extremal machine, then the moving $(p, q) \rightarrow (p', q')$ of a $J$-component does not increase the number of splittings if $0 < p < q < |\sigma|_{M_i}$. Indeed, since there are only two jobs scheduled on $M_i$ (the common job and the fixer), the cutting of the $J$-component will reduce by 1 the number of splittings of the fixer.

The following lemma summarizes the above made observations.

**Lemma 7** Let $\sigma'$ be a schedule produced from $\sigma$ by the $(p, q) \rightarrow (p', q')$-moving of a $J$-component on machine $M_i$, such that $q' \leq |\sigma|_{M_i}$. Then

1 $\sigma'$ is tight if $\sigma$ is tight;

2 $sp(\sigma') \leq sp(\sigma) + 1$;

3 $sp(\sigma') \leq sp(\sigma)$ if $p' < p$ and $p'$ is switching point for $\sigma$ on $M_i$;

4 $sp(\sigma') < sp(\sigma)$ and $sp.J(\sigma') < sp.J(\sigma)$ if $p' < p$ and $p'$ is $J$-switching point in $\sigma$ on $M_i$;

5 $sp(\sigma') \leq sp(\sigma)$ if $i = 1$ or $i = m$. 

617
Lemma 8 A feasible tight schedule for $O(C,m)$ with no more than $m - 2$ splittings can be constructed in linear time.

Proof. First construct a tight schedule for all jobs except the common job $I$. Then for each $M_i$, insert $I$ in this schedule by the $(i-1,i)$-insertion. This insertion may create at most one splitting on an intermediate machine. \hfill\square

3.3 Schedule preprocessing

In this section we introduce procedures based on cutting, inserting and moving, which simplify a schedule structure. A sequence of $J$-components $(p_1, i_1, q_1), (p_2, i_2, q_2), \ldots, (p_k, i_k, q_k)$ is called a $J$-loop, if $q_j = p_{j+1}$ for all $j < k$, and $i_1 = i_k$; we will say that the above $J$-loop passes through machines $M_{i_2}, M_{i_3}, \ldots, M_{i_{k-1}}$.

A $J$-component $(p, i, q)$ is called unit if $q - p = 1$, otherwise it is called non-unit. A $J$-loop is called integral if any of its component, except the first one and the last one are unit, and all machines which this loop passes (the intermediate ones) have at least one splitting.

We will say that a $J$-component $(p, i, q)$ splits a job $J'$, if both $p$ and $q$ are switching points of $J'$, i.e., if $M_i$ processes $J'$ immediately before time $p$ and immediately after time $q$.

Lemma 9 If a $J$-component splits another job, then any its moving will not increase the total number of splittings.

Proof. The cutting of such a $J$-component will decrease the number of splittings, and the insertion can again increase the number of splittings at most by 1. Hence, the total number of splittings will not increase. \hfill\square

Lemma 10 Given any tight schedule $\sigma$, it is possible to construct in linear time another tight schedule $\sigma'$ with $sp(\sigma') \leq sp(\sigma)$, such that any machine $M_i$ without an I-splitting in $\sigma$ with $sp^o(M_i) > 0$ will have at most one splitting in $\sigma'$ (and this splitting will be caused by the unique I-component).

Proof. Let $(p, i, p+1)$ be the unique I-component on $M_i$ and let $sp^o(M_i) > 0$. A tight schedule $\sigma'$ with at most 1 splitting on $M_i$ can be obtained as follows. First construct a tight schedule on $M_i$ including in it all jobs, scheduled in $\sigma$ on $M_i$, but job $I$. This schedule has no splittings. Then carry out the $(p, p+1)$ insertion of $I$ on $M_i$. The resulted schedule satisfies our claim for $M_i$. We carry out the analogous procedure on any other machine without a I-splitting and obtain the resulted schedule $\sigma'$ with $sp(\sigma') \leq sp(\sigma)$. \hfill\square

Lemma 11 Given any tight schedule $\sigma$, it is possible to construct in linear time another tight schedule $\sigma'$ without any integral loop and with $sp(\sigma') \leq sp(\sigma)$.
Proof. Due to the previous Lemma 10, without loss of generality, we can assume that in $\sigma$ all machines with unit $I$-components have at most 1 splitting generated by the corresponding unique $I$-component. If $\sigma$ contains an integral loop $(p, i, q), (q, i_1, q + 1), (q + 1, i_2, q + 2), \ldots, (q + k - 1, i_k, q + k), (q + k, i, r)$, then the machines $M_{i_1}, \ldots, M_{i_k}$ have just one splitting generated by the corresponding $I$-component. Then by Lemma 9, a moving of these components will not increase the number of splittings. Hence, the schedule $\sigma'$, obtained from $\sigma$ by the following sequence of movings will have less splittings than $\sigma$ has: on $M_i$ perform the moving $(q + k, r) \rightarrow (q, r - k)$; on $M_{i_1}$ perform the moving $(q, q + 1) \rightarrow (r - k, r - k + 1)$; on $M_{i_2}$ perform the moving $(q + 1, q + 2) \rightarrow (r - k + 1, r - k + 2)$; and so on, the last moving $(q + k - 1, q + k) \rightarrow (r - 1, r)$ is performed on $M_{i_k}$. The first from the above movings decreases the number of splittings on $M_i$ and any successive moving does not increase the number of splittings. Now if $\sigma'$ still contains an integral loop, then we apply the same procedure again. The process has to terminate in $sp(\sigma)$ steps.

Let us say that an intermediate machine $M_i$ in a schedule $\sigma$ is incorrect, if $\sigma$ has no splitting on $M_i$ and the common job $I$ is scheduled first or the last on $M_i$.

**Lemma 12** Given a tight schedule $\sigma$, associated with $O(C, m)$, it is possible to construct in linear time another tight schedule $\sigma'$ with $sp(\sigma') \leq sp(\sigma)$ and without any incorrect machine.

Proof. Note that $\sigma$ can have no more than two incorrect machines: on one of these machines $I$ is scheduled first and on the other one $I$ is scheduled last. Let $M_i$ be the incorrect machine on which $I$ is scheduled first (the other case will be quite similar). The schedule $\sigma'$ which we construct from $\sigma$ coincides with $\sigma$ on all machines except the machines $M_1$ and $M_i$. We define $\sigma'$ on $M_1$ as follows: $\sigma'(I, M_1) = [0, 1)$ and $\sigma'(F_1, M_1) = [1, m)$.

Now let $(p_1, 1, q_1), (p_2, 1, q_2), \ldots, (p_k, 1, q_k)$ be the increasing sequence of all $I$-components on $M_1$ in $\sigma$. As $p_1 > 0$, the number of splittings in $\sigma$ on $M_1$ is $2k - 2$ if $q_k = m$ and it is $2k - 1$ otherwise. $\sigma'$ on $M_i$ is defined as follows. First we perform the $[0, 1)$-cutting on $M_i$ which leaves 0 splittings on $M_i$. Further, step by step we perform $(p_i, q_i)$-insertions of $I$ on $M_i$. Any such insertion but the last one increases the number of splittings at most by 2. The last insertion increases the number of splittings by one if $q_k = m$. Hence, the number of splittings in $\sigma'$ on $M_i$ is no more than the number of splitting in $\sigma$ on $M_1$, and $sp(\sigma') = sp(\sigma) = 0$. Therefore, $sp(\sigma') \leq sp(\sigma)$.

### 3.4 Proper times

For a real $x$, we denote by $\langle x \rangle$ its **fractionation** defined as the distance from $x$ to the nearest integer number.
A time moment \( t \) is called a \textit{proper time} for a machine \( M_i \) in a schedule \( \sigma \) if it is an integer number plus the sum of the processing times of a proper non-empty subset of the jobs, scheduled in \( \sigma \) on \( M_i \) and different from \( I \). For example, for machines \( M_1 \) and \( M_m \) any integral number is a proper time. For an intermediate machine \( M_i \) the following inequalities hold.

**Lemma 13** The fractionation of a proper time \( t \) of an intermediate machine \( M_i \) satisfies the inequalities

\[
\frac{1}{S_{2i}} \leq \langle t \rangle \leq \frac{1}{S_{2i}-1}
\]

Proof. \( t \) has a form \( f + p \), where \( f \) is the sum of processing times of 0, 1 or 2 fixers and \( p \) is the sum of processing times of a set of partition jobs. (Note that by the definition, \( f + p > 0 \) and since in this sum not all fixers and partition jobs can be presented, it is not integral.) Since in \( f + p \) at least one job processing time is included, it follows immediately from the definition of processing times of partition jobs and fixers that \( \langle f + p \rangle \geq \frac{1}{S_{2i}} \). To see the other inequality, note that \( 0 \leq p \leq \frac{1}{S_{2i}-1} \) and the list of possible values of \( f \) is \( 0, i - 1 - \frac{1}{S_{2i}-1}, m - i - \frac{1}{S_{2i}-1} \) and \( m - 1 - \frac{1}{S_{2i}-1} \). It is straightforward to check that this implies \( \langle f + s \rangle \leq \frac{1}{S_{2i}-1} \).

For a machine \( M_i \), a time moment which is not proper is called \textit{improper}. A proper time for some machine is said to be \textit{strongly improper}, for any other machine in \( \sigma \).

**Lemma 14** A strongly improper time for any intermediate machine is improper for that machine.

Proof. The fractionation of a proper time for an intermediate machine \( M_i \) is from the interval \( \left[ \frac{1}{S_{2i}}, \frac{1}{S_{2i}-1} \right] \) (Lemma 13). If \( t \) is a proper time for an intermediate machine \( M_j, j \neq i \), then \( \langle t \rangle \) is from the interval \( \left[ \frac{1}{S_{2j}}, \frac{1}{S_{2j}-1} \right] \). But the intersection of the two above intervals is empty. The same is true if \( M_j \) is an extremal machine, since a proper time of such a machine is integral.

**Lemma 15** If \( t_1, t_2 \) and \( t_3 \) are proper times for different machines, then \( t_1 \neq t_2 \pm t_3 \).

Proof. Let \( t_1, t_2 \) and \( t_3 \) be proper times for machines \( M_i, M_j \) and \( M_k \), respectively. Without any loss of generality, assume \( i < j < k \). If \( i = 1 \) then \( t_1 \) is integer and \( t_3 \pm t_1 \) are proper times for \( M_k \) and hence strongly improper for \( M_j \). Then by Lemma 14 \( t_2 \neq t_3 \pm t_1 \). The proof is similar for \( k = m \).

Let us assume now that \( 1 < i < j < k < m \), and by a contradiction, suppose \( t_1 = t_2 + t_3 \). Then since \( S_{2j}t_1 \) and \( S_{2j}t_2 \) are integral, \( S_{2j}t_3 \) also has to be integral. But the inequalities \( \frac{1}{S_{2j}} \leq \langle t_3 \rangle \leq \frac{1}{S_{2j}-1} \) imply
Our this sections last lemma easily follows from the definition of proper times.

**Lemma 16** Suppose $\sigma$ is a tight non-partitioning schedule and $M_i$ is any correct machine in $\sigma$ without a splitting. Then any switching point in $M_i$ is proper.

### 3.5 The proof of Lemma 5

From now on we will be dealing exclusively with an associated with $O(C, m)$ tight non-partitioning schedule without any integral loop and any incorrect machine. We denote such a schedule by $\sigma$. By virtue of Lemmas 11 and 12 it is sufficient to prove Lemma 5 for $\sigma$. We wish to prove that $\sigma$ has at least $m - 2$ splittings. If $\sigma$ has a splitting on all intermediate machines, then the number of splitting is at least $m - 2$ and our assertion is true. Hence, it remains to consider the case when the set of machines without a splitting contains an intermediate machine in $\sigma$.

We denote by $s_i$ and $f_i$, respectively, the starting time and finishing time, respectively, of the common job $I$ on machine $M_i$. Note that $I$ is not split on $M_i$ iff $f_i - s_i = 1$. Let $M_i$ and $M_j$ be a pair of machines without any splitting in $\sigma$ such that $f_i \leq s_j$. Then we call such a pair a $\sigma$-successive pair if there is no other machine $M_k$ without splittings, such that $f_i \leq s_k \leq f_k \leq s_j$. Notice that since $\sigma$ is a non-partitioning schedule and there is no splitting on machines $M_i$ and $M_j$, $s_j$ is in fact strongly more than $f_i$. This implies that there must exist at least one machine with a splitting, on which a $I$-component is scheduled within the interval $[f_i, s_j]$ (since job $I$ has to be in process at any moment in $\sigma$). Also observe that either $M_i$ or $M_j$ must be intermediate.

We will say that a $I$-component $(p, k, q)$ precedes another $I$-component $(p', k', q')$, if $q \leq p'$; a $I$-component $(p, k, q)$ is a $(i, j)$-intermediate, if $f_i \leq p \leq q \leq s_j$. A $I$-component $(p, k, q)$ is improper for an extremal machine $M_k$ if $p - q < 1$ and $0 < p < q < m$. This component is improper for an intermediate machine $M_k$ if, besides the above two conditions for extremal machines, both $p$ and $q$ are improper times for $M_k$. An improper $I$-component which has one (two, respectively) strongly improper end(s) is called semi-strongly improper (strongly improper, respectively).

**Lemma 17** Let $M_i$, $M_j$ be a $\sigma$-successive pair. Then

1. the set of $(i, j)$-intermediate improper $I$-components is not empty;
2. if the earliest scheduled $(i, j)$-intermediate improper $I$-component belongs to an intermediate machine, then it has a strongly improper starting time;
3. if the latest scheduled $(i, j)$-intermediate improper $I$-component belongs to an intermediate machine, then it has a strongly improper finishing time.
Proof. The difference \( s_j - f_i \) is not integral since otherwise both, \( s_j \) and \( f_i \) would be proper for both \( M_i \) and \( M_j \) and this cannot be true because at least one of these machines is intermediate (see the definition of a proper time and Lemma 14). Since the length of \( I \) coincides with the makespan of our schedule, this job is processed during the whole interval \([s_j, f_i)\), i.e., there is an \( I \)-component inside this interval. Let \((p_1, k_1, q_1)\) be the earliest non-unit \( I \)-component after \( f_i \). If \( M_{k_1} \) is an extremal machine, then this \( I \) component is improper (since it is non-unit) and (1) is proved. If \( M_{k_1} \) is an intermediate machine, then \( p_1 \) may differ from \( f_i \) only by an integer (the sum of processing times of unit \( I \)-components), hence it is proper for \( M_i \) and strongly improper for \( M_{k_1} \). If \( q_1 \) is improper for \( M_{k_1} \), then this component is improper and statements (1) and (2) are proved. Assume \( q_1 \) is proper for \( M_{k_1} \). Then we continue to search an improper \( I \)-component. The difference \( s_j - q_1 \) is again non-integral as it is a difference of proper times of different machines. Hence, there is at least one non-unit \( I \)-component scheduled between \( q_1 \) and \( s_j \). Let \((p_2, k_2, q_2)\) be the earliest such component. If \( k_2 = 1 \) or \( k_2 = m \), this component is improper and the lemma is proved. Assume \( 1 < k_2 < m \). Because \( \sigma \) has no integral loops, \( k_2 \neq k_1 \). Since \( p_2 \) is the sum of \( f_i \) and an integer number, it is proper for \( M_{k_1} \) and hence strongly improper for \( M_{k_2} \). If \( q_2 \) is improper, \((p_2, k_2, q_2)\) is improper and the statements (1) and (2) are proved. If not, we continue the procedure and look for the next \( I \)-component \((p_3, k_3, q_3)\) and so on. This process cannot take more steps than the number of splittings of \( \sigma \), and we obtain an improper \( I \)-component with strongly improper start time. The statements (1) and (2) are proved.

To prove (3), we merely replace in the above the "first" by the "last".

We use the above lemma to mark intermediate improper \( I \)-components for every \( \sigma \)-successive pair \((M_i, M_j)\). In particular, if there is an \((i, j)\)-intermediate improper \( I \)-component on an extremal machine, then we mark the earliest such component. Otherwise, if there is only one \((i, j)\)-intermediate improper \( I \)-component, we mark this piece (such component will be strongly improper by Lemma 17). If there is more than one \((i, j)\)-intermediate improper \( I \)-component, we mark the earliest and the latest of these \( I \)-components; we call a couple of such components a twin couple.

Let us denote by \( mr(M) \) the number of the marked \( I \)-components, scheduled on machine \( M \); if \( mr(M) > 0 \), then we call \( M \) marked. Let \( M \) be a machine with at least one splitting. Then the number of spare splittings on a \( M \), \( sp^s(M) = sp(M) - 1 \).

**Lemma 18** If \( M \) is an extremal marked machine, then \( sp^s(M) \geq mr(M) \).

Proof. Note that the number of switching points on \( M \) is at least \( 2mr(M) + 2 \). Using Lemma 6 with \( n = 2 \) and \( s = 2mr(M) + 2 \), we obtain that \( sp(M) \geq 2mr(M) + 2 - 2 = 2mr(M) - 1 \). If \( mr(M) > 1 \), then \( 2mr(M) - 1 \geq
mr(M) + 1; if mr(M) = 1, then job $I$ has at least one splitting and the fixer also has at least one splitting because the marked component is scheduled in an intermediate position on $M$ and we have found at least 1 spare splitting. This completes the proof. \(\square\)

**Lemma 19** If an intermediate machine $M_k$ is such that $mr(M_k) > 2$, then $sp^*(M_k) \geq mr(M_k)$.

**Proof.** We first consider the case when there is a non-marked $I$-component on $M_k$, or equivalently $sp_I(M_k) \geq mr(M_k)$. Then to prove our lemma it is sufficient to find some split job on $M_k$, different from job $I$. Let $p_1$ be the starting time of the earliest marked $I$-component on $M_k$. There are scheduled only components of jobs, different from $I$ in the interval $(0, p_1)$, and at least one of these components has to be fractional since the length of this interval is improper.

Thus we can assume that $sp_I(M_k) < mr(M_k)$, i.e., all $I$-components on $M_k$ are marked. Consider next the case when we have three marked $I$-components $(p_1, k, q_1), (p_2, k, q_2)$ and $(p_3, k, q_3)$, $p_1 < p_2 < p_3$ which do not contain a twin couple. Hence all these pieces are intermediate for some three different $d$-successive pairs. In this case each of the following intervals $(0, p_1), (q_1, p_2), (q_2, p_3)$ and $(p_3, m)$ has the length more than 1. Therefore, in each of these intervals has to be scheduled a fixer. If the same fixer is scheduled in two different intervals, then it has a splitting; if a fixer is scheduled in three different intervals, it has at least two splitting. As it is easily seen, in all cases the total number of splittings of a fixer is no less than 2. On the other hand, the number of splittings of job $I$ is at least $mr(M_k) - 1$. Hence the total number of splittings on $M_k$ is at least $mr(M_k) - 1 + 2 = mr(M_k) + 1$.

Now let us consider the case when there are 3 marked components on $M_k$, but first two of them form a twin couple. Since $p_1$ and $q_3$ are improper, the intervals $(0, p_1)$ and $(q_3, m)$ contain a fractional component of some job, say $J_1$ for $(0, p_1)$ and $J_2$ for $(q_3, m)$. If $J_1 \neq J_2$, then $sp(M_k) \geq sp_I(M_k) + 2 \geq mr(M_k) - 1 + 2$ and our lemma holds. Suppose $J_1 = J_2$. Observe that from Lemma 17, both $p_1$ and $q_2$ are strongly improper. Besides, either $p_3$ or $q_3$ is also strongly improper. Assume first that $p_3$ is strongly improper. Then the interval $(q_2, p_3)$ has an improper length by Lemma 15 and hence contains a fractional component of some job, say $J_3$. If $J_3 \neq J_1$, the reasoning, similar to that used for $J_2$ above, proves our lemma. If $J_3 = J_1$, then $J_1$ has at least 2 splittings and so $sp(M_k) \geq sp_I(M_k) + sp_{J_1}(M_k) \geq mr(M_k) - 1 + 2$ and the lemma is also proved.

Now assume that $q_3$ is strongly improper. Since all $I$-components on $M_k$ are marked, the sum of the length of all $I$-intervals $(p_i, q_i)$ is 1, and the sum of the length of the intervals $(q_1, p_2)$ and $(q_2, p_3)$ is respectively $1 + q_3 - p_1$ and it is improper by Lemma 15. Hence the length of one of these intervals is
improper by Lemma 15 and it must contain a fractional job component. The proof now is completed similarly as for the above considered case.

The case when a single marked $I$-component follows after a twin couple is analogous. Finally, the only non-considered case is when there are four marked $I$-components on $M_k$, which form 2 different twin couples. Then the following three intervals $(0, p_1), (q_2, p_3)$ and $(q_4, m)$ have improper length and hence contain a fractional component of a job, different from $I$, and we again have found an additional spare splitting.

**Lemma 20** If $mr(M_k) = 2$ and at least one of the marked components on $M_k$ is strongly improper, then $sp^s(M_k) \geq 2$.

**Proof.** Suppose $(p, k, q)$ is a strongly improper marked $I$-component. The second marked $I$-component $(p', k, q')$ is then a semi-strongly improper. Assume $q < p'$ and $p'$ is strongly improper (and hence $q'$ is improper). The length of the interval $(0, p)$ is improper for $M_k$ (Lemma 17), hence it contains a fractional component of some job, say $J_1$. The same is true for the interval $(q', m)$. If this interval contains a fractional component of some another job $J_2$, then we already have founded three split jobs $I$, $J_1$ and $J_2$ and the lemma holds. Suppose $J_2 = J_1$. If $I$ has more than 2 components on $M_k$, then $I$ has 2 splittings and $J_1$ gives another splitting, so we again have 3 splittings on $M_k$. Suppose $I$ has just 1 splitting on $M_k$. In this case the length of interval $(q, p')$ is exactly 1 less than the length of $(p, q')$, whereas the last interval is improper as the difference of two strongly improper times (Lemma 15). Hence, the length of $(q, p')$ is improper and this interval contains a fractional job component. If this component is of job $J_1$, then $J_1$ has at least two splittings, and we have 3 splittings on $M_k$ together with the splitting of job $I$. If the above component is of some other job, we again have 3 split jobs.

The case when $q'$ is strongly improper is simpler, because then we immediately have that the length of the interval $(p, q')$ is improper as the difference of two strongly improper times. The case $q' < p$ is analogous.

**Lemma 21** Every marked machine $M_k$ has at least 2 splittings.

If $I$ has more than 1 splitting on $M_k$, then we have enough splittings. Assume that $I$ has just 1 splitting and let $(p, k, q)$ be a marked $I$-component. Either the interval $(0, p)$ or the interval $(q, m)$ does not contain a component of job $I$. As that interval, in addition, has an improper length, it must contain a fractional component of some job, different from $I$. So we have found two split jobs on $M_k$ and the lemma is proved.

**The proof of Lemma 5.** Now we are ready to prove the main Lemma 5. Let $k$ and $m - k$ be the number of machines without and with splittings, respectively. The total number of splittings is $m - k$ plus the number of spare splittings. Hence it is sufficient to prove that $sp^s(\sigma) \geq m - 2 - (m - k) = k - 2$. 

624
Denote by \( p \) the number of \( \sigma \)-successive pairs for which a twin couple is marked. Then the total number of the marked components is \( k - 1 + p \). Indeed, the number of \( \sigma \)-successive pairs is \( k - 1 \). Each such pair generates one or two marked components, while the number of the pairs which generate two components is \( p \).

Let us call a marked machine *distinctive*, if it contains exactly two marked components, such that each of these components is from some twin couple (the two components may belong to different twin couples). We denote by \( q \) the number of all distinctive machines. Since the total number of twins is \( 2p \) and the number of twins, scheduled on a distinctive machine is \( 2q \), we have that \( 2q \leq 2p \), or equivalently \( q \leq p \).

By Lemma 21, each distinctive machine has at least 1 spare splitting. Hence, the distinctive machines give in total at least \( q \) spare splittings. The number of the marked components, scheduled on non-distinctive machines is \( (k - 1 + p) - 2q \geq k - 1 - q \). By virtue of Lemmas 20, 19 and 18, each marked component on a non-distinctive machine generates at least one spare splitting. Hence, the total number of spare splittings is at least \( q + (k - 1 - q) = k - 1 \) (this estimation is even better than the claimed one; it holds only if there exists an intermediate machine without splittings).

4. **NP-hardness of** \( R/p_{ij} \leq C^*_\max; \text{pmtn}(2m - 4)/C_\max \)

In this section we apply our previous NP-hardness result for \( O/\text{acyclic}; \text{pmtn}(m - 3)/C_\max \) and show that \( R/p_{ij} \leq C^*_\max; \text{pmtn}(2m - 4)/C_\max \) is also NP-hard. First we prove the following auxiliary lemma:

**Lemma 22** Two distributions \( \delta \) and \( \delta' \) are equal if they have the same acyclic assignment and the load on all machines in both \( \delta \) and \( \delta' \) is the same.

**Proof.** Assume \( \delta \) and \( \delta' \) are two arbitrary distributions satisfying the lemma. The proof is by the induction on the number of machines. Suppose the lemma is proved for distributions with \(< m \) machines.

Let \( \delta \) and \( \delta' \) be two acyclic distributions with \( m \) machines from \( M \). Since \( \delta \) and \( \delta' \) have the same acyclic preemption graph, there is a node of degree one \( N \) in this graph. Let \( I \) be the job, corresponding to the (only) edge of \( M \). Note that the rest of the jobs, distributed in \( \delta \) and \( \delta' \) on \( M \) have no preemption, i.e., they are assigned completely to \( M \). Therefore, the length of all these jobs in \( \delta \) and \( \delta' \) is the same. But since the load of \( M \) in \( \delta \) and \( \delta' \) is the same, the length of the portion of \( I \) on \( M \) in both \( \delta \) and \( \delta' \), must be also the same. Hence \( \delta(J, M) = \delta'(J, M) \) for all \( J \). To apply the induction hypothesis, we define distributions \( \delta_0 \) and \( \delta'_0 \) on \( M \setminus M \) in the following way: \( \delta_0(J, M) = \delta(J, M) \) if \( J \neq I \) and \( \delta(J, M) > 0 \), \( \delta_0(I, M) = \delta(I, M) \frac{1}{1 - \delta(I, M)} \); \( \delta'_0(J, M) = \delta'(J, M) \)
if $J \neq I$ and $\delta'(J, M) > 0$, $\delta'_0(I, M) = \delta'(I, M) \frac{1}{1 - \delta'(I, M)}$. Distributions $\delta_0$ and $\delta'_0$ coincide the by induction hypothesis. This implies that $\delta = \delta'$.

Given a multiprocessor $J, M$, let us say that a distribution $\delta$ on $J, M$ generates an open shop $O$ on $J, M$, if $|J'_i| = \delta(J_i, M_i)M_i(J_i)$.

**Theorem 23** For any uniform acyclic open shop $O$ on $J, M$, there is a processing time function $f$ and a distribution $\delta$ for the multiprocessor $J, M$ with this processing function, such that $\delta$ generates $O$ and $\delta$ is a unique optimal distribution for $J, M$.

Proof. First we define $f$ as follows: $M_i(J_i) = |J_i| + \varepsilon$ if $J_i$ is dummy, and $M_i(J_i) = |J_i|$ otherwise, where $\varepsilon$ is a positive real number. Now we define the distribution $\delta$ as $\delta(J_i, M_i) = |J'_i| / |J_i|$. Note that if $|J'_i| > 0$, then $\delta(J_i, M_i)M_i(J_i) = M_i(J_i)\left(\frac{|J'_i|}{|J_i|}\right) = |J'_i|$. Similarly, $|J'_i| = 0$ implies $\delta(J_i, M_i) = 0$. Hence, $(M_i)\delta(J_i) = |J'_i|$ holds for all $i, j$ and $\delta$ generates $O$. Since $O$ is uniform, $\delta$ is also uniform and the total processing time of $\delta$ is $m|\delta|_{\text{max}}$. Besides, every job in $\delta$ is distributed on its fastest machine (i.e. the machine where this job can be processed in the minimal time). This implies that $\delta$ has the minimal possible total processing time, and since $\delta$ is uniform, it is optimal.

It remains to show that $\delta$ is unique. Assume $\delta'$ is another optimal distribution such that $\delta'$ generates $O$. The total processing time in $\delta'$ is no more than $m|\delta'|_{\text{max}}$, but the latter by our assumption is no more than $m|\delta|_{\text{max}}$; now since $m|\delta|_{\text{max}}$ is the minimal possible total processing time, the total processing time in $\delta'$ is to be equal to $m|\delta|_{\text{max}}$. Hence, $m|\delta'|_{\text{max}} = m|\delta|_{\text{max}}$ and $\delta'$ is also uniform, i.e., loads on all machines in both, $\delta$ and $\delta'$ are equal. At the same time, $\delta'$ has to distribute each job on the fastest machine. But the processing time function is defined in such a way that a machine $M$ is fastest for a job $J$ if $\delta(J, M) > 0$. Hence, $\delta'$ and $\delta$ have the same acyclic assignment and our claim follows from Lemma 22.

**Theorem 24** $R/p_{ij} \leq C^*_\text{max}p\min(2m - 4)/C_{\text{max}}$ is NP-hard.

Proof. We prove that the corresponding decision problem is NP-complete. Recall that the open-shop problem $O(C, m)$ of Section 4 is acyclic and uniform. We apply Theorem 23 with $\varepsilon \leq 1$ to $O(C, m)$ to construct a processing time function $f$ for multiprocessor $J, M$, and a distribution $\delta$, such that $\delta$ generate $O(C, m)$. Note that $|\delta|_{\text{max}} = m$ and $p_{\text{max}} = m$, hence the multiprocessor $J, M$, defined by the processing time function $f$, is non-lazy (i.e., $p_{\text{max}} \leq C_{\text{max}}$).

It is not difficult to see that our decision problem, “does there exist a feasible schedule $\sigma$ for the multiprocessor $J, M$ with $||\sigma|| \leq m$ and with $pr(\sigma)$ ≤
has a “yes” answer if and only if there exists a tight schedule for $O(C, m)$ with the makespan, not exceeding $m$ and with less than $m - 2$ splittings. Indeed, if $\sigma$ is such a schedule for $O(C, m)$, then $\sigma$ is a feasible schedule for the multiprocessor $\mathcal{J}, \mathcal{M}$ with less than $(m - 1) + (m - 2) = 2m - 3$ preemptions.

In the other direction, suppose $\sigma$ is a feasible schedule for $\mathcal{J}, \mathcal{M}$ with $||\sigma|| \leq m$ and $pr(\sigma) < 2m - 3$. Then the makespan of any distribution associated with $\sigma$ is $m$ and it is optimal. Since there is only one such distribution (Lemma 23), it is precisely the distribution $\delta$. $\delta$ has exactly $m - 1$ preemptions. It follows that $\sigma$ is a feasible schedule for $O(C, m)$ with the number of splittings $pr(\sigma) - (m - 1) < m - 2$.

5. Polynomial approximation for little-preemptive multiprocessor scheduling

As mentioned earlier in Section 1, the NP-hardness results for $P/pmtn(m - 2)/C_{\text{max}}$ and $O/acyclic, pmtn(m - 3)/C_{\text{max}}$, respectively, are tight due to the polynomial $(m - 1)$ and $(m - 2)$-preemptive algorithms for $P/pmtn/C_{\text{max}}$ and $O/acyclic, pmtn(m - 2)/C_{\text{max}}$, respectively, from McNaughton [6] and Shchepin & Vakhania [11], respectively. The tightness of the NP-hardness result for $R/p_{ij} \leq C^*_{\text{max}}, pmtn(2m - 4)/C_{\text{max}}$ follows from the following theorem. The proof of this and the next theorem are based on the earlier results from Shchepin & Vakhania [9].

**Theorem 25** $R/p_{ij} \leq C^*_{\text{max}}, pmtn(2m - 3)/C_{\text{max}}$ is polynomially solvable.

**Proof.** An optimal acyclic distribution $\delta$ with $pr(\delta) \leq m - 1$ and $||\delta||_{\text{max}} \leq C^*_{\text{max}}$ can be obtained by linear programming. For each $J \in \mathcal{J}$, we have $|J|^\delta = \sum_{M \in \mathcal{M}} \delta(J, M)M(J) \leq \sum_{M \in \mathcal{M}} \delta(J, M)p_{\text{max}} = p_{\text{max}} \leq C^*_{\text{max}}$. Hence $||\delta|| \leq C^*_{\text{max}}$ and we can apply Lemma 4 from [9] to construct an optimal schedule with less than $m - 2$ splittings for $\delta$. Since $\delta$ has at most $m - 1$ preemptions, the total number of preemptions of the constructed schedule will be no more than $(m - 1) + (m - 2) = 2m - 3$.

For general unrelated processor system we can guarantee a near-optimal approximation as the following theorem shows:

**Theorem 26** There is a polynomial time algorithm, which for any instance of $R/pmtn/C_{\text{max}}$ constructs a feasible schedule $\sigma$ with $pr(\sigma) \leq 2m - 3$ and with

\[ (a) \ ||\sigma|| \leq \max\{C^*_{\text{max}}, p_{\text{max}}\}; \]

\[ (b) \ ||\sigma|| \leq C^0_{\text{max}}, \text{ where } C^0_{\text{max}} \text{ is the optimal non-preemptive schedule makespan.} \]
Proof. Part (a) immediately follows from Theorem 25. We refer the reader to Theorem 2 from [9] for the proof of part (b).

6. Further research

We have shown that $O/\text{acyclic\,pmtn}(m - 3)/C_{\text{max}}$, $P/\text{pmtn}(m - 2)/C_{\text{max}}$, and $R/p_{ij} \leq C_{\text{max}}^*, \text{pmtn}(2m - 4)/C_{\text{max}}$ are NP-hard in the normal sense. Can there exist pseudo-polynomial algorithms for (some of) these problems or are they (some of them) strongly NP-hard? A polynomial algorithm for $R/\text{pmtn}/C_{\text{max}}$ by Lawler and Labetoulle [4] gives up to $(4m^2 - 5m + 2)$ preemptions, whereas our result for $R/p_{ij} \leq C_{\text{max}}^*, \text{pmtn}/C_{\text{max}}$ ensures that the critical number of preemptions for $R/\text{pmtn}/C_{\text{max}}$ is no less than $(2m - 4)$. Can this lapse be reduced?
References


