Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2005-02. Published by the Egrerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres. ISSN 1587–4451.

On scheduling problems with parallel multi-purpose machines

Zsuzsanna Vaik

January 2005

On scheduling problems with parallel multi-purpose machines

Zsuzsanna Vaik*

Abstract

In a multi-purpose machine scheduling problem, jobs can be processed by any machine of a prespecified subset of the machine-set. For the preemptive problem we give a combinatorial algorithm and obtain a min-max theorem for the minimal makespan. We give a combinatorial 2-approximation algorithm based on this algorithm for the non-preemptive case, which is $\mathcal{NP}$-hard. We introduce a new special preemption rule: a job may be interrupted and resumed immediately on another machine. This version is also $\mathcal{NP}$-hard, and a 2-approximation is presented.

1 Introduction

In a multi-purpose parallel machine scheduling problem, a job can be processed by any machine of an associated, prespecified subset of the machine set. Thus, these problems generalize classical parallel machine problems in which a job can be processed by each machine of the machine set.

A scheduling problem with multi-purpose machines (MPM) can be described as follows. We have m machines $M_1, \ldots, M_m$ and n jobs $j = 1, \ldots, n$. Job j has to be processed by one machine of a prescribed set $\mu_j \subseteq \{M_1, \ldots, M_m\}$. The set of machines is denoted by M and the set of jobs is denoted by J. Job j has processing time p_j. The objective is to find a schedule that minimize the makespan. To describe different types of multi-purpose machine scheduling problems, we use the extended $\alpha|\beta|\gamma$-notation of [2], that is in the α-part we add MPM to indicate a scheduling model with multi-purpose machines.

Recently, combinatorial algorithms have been obtained for some parallel multi-purpose machine problems: Pinedo [8] solved $P\ MPM|p_j = 1|C_{\max}$, in the special case, when the sets μ_j are nested. He gave a list-scheduling algorithm, and mentioned that if the sets are not nested, then the list-scheduling does not always work even if there are only 3 machines. (For two machines the sets are always nested.) Brucker, Jurisch, Krämer

*Dept. of Operations Research, Eötvös University, Pázmány Péter sétány 1/C, Budapest, Hungary H-1117. Research is supported by OTKA grant T 037547, by European MCRTN Adonet, Contract Grant No. 504438 and by the Egerváry Research Group of the Hungarian Academy of Sciences. e-mail: zsuzska@cs.elte.hu

[2] showed that $P\ MPM|p_j = 1, r_j|\sum w_jU_j(\sum w_jT_j)$ can be formulated as a minimum cost matching problem in a bipartite graph, and so $P\ MPM|p_j = 1|C_{\max}$ likewise. They gave a combinatorial ε-approximation algorithm for $P\ MPM|pmtn, r_j|L_{\max}$, which is based on network flow concepts. Note, that the parallel multi-purpose machine problem is a special case of the problem of unrelated parallel machine problem, when the processing time of a job depends on the machines, i.e. the job j has processing time of length p_{ij} if it is scheduled on machine M_i. To see this define $p_{ij} = p_j$ if job j can be processed by machine i (i.e. $M_i \in \mu_j$), and let $p_{ij} = \infty$ otherwise. Harvey, Ladner, Lovász, Tamir [6] gave a combinatorial algorithm for $R|p_{ij} = \{1, \infty\}|\sum C_j$, and they proved that this algorithm is optimal for $R|p_{ij} = \{1, \infty\}|C_{\max}$ too.
We consider the problem $P\ MPM|pmtn|C_{\max}$ in Section 2. We give two lower bounds for the optimal makespan and a polynomial time combinatorial algorithm. The schedule obtained has makespan equal to the maximum of the lower bounds, thus a min-max theorem is achieved. In Section 3 we consider the problem $P\ MPM||C_{\max}$, which is $\mathcal{NP}$-hard even if the sets μ_j are equal, see Garey, Johnson [3]. The list-algorithm is well-known to give a 2-approximation algorithm for $P||C_{\max}$ [5]. We show that for the multi-purpose machine case this does not hold. The problem $R||C_{\max}$ admits a 2-approximation algorithm by solving a linear program (LP), then rounding an extremal solution of the LP, Lenstra, Shmoys, Tardos [7]. We give a new combinatorial 2-approximation algorithm for the problem $P\ MPM||C_{\max}$ based on the combinatorial algorithm of Section 2. Finally, in Section 4 we introduce a new type of preemption rule, when a job may be interrupted and resumed on another machine, but it must be resumed immediately. We show that the arising problem is also $\mathcal{NP}$-hard, and a 2-approximation algorithm is presented.

2 $P\ MPM|pmtn|C_{\max}$

Problem $R|pmtn, r_j|L_{\max}$ can be formulated as an LP [1], thus its special case,
$P\ MPM|pmtn, r_j|L_{\max}$ is polynomially solvable. Brucker, Jurisch, Krämer [2] constructed a network, and showed that there exists a feasible solution of
$P\ MPM|pmtn, r_j, L_i \leq L|\cdot$, if and only if there exists a feasible flow with given flow-value. Their algorithm gives a feasible schedule in $\mathcal{O}(n^3(n+m)^3)$ time, and from that we can get by binary search a solution with objective function which differs from the optimal objective value by at most ε time units.
We give an algorithm for $P\ MPM|pmtn|C_{\max}$. Bucker's algorithm is a good choice if a fast approximation is needed, but our algorithm gives the exact solution in polynomial time, and the algorithmic proof gives a new min-max theorem.

Theorem 2.1. *For a given instance of $P\ MPM|pmtn|C_{\max}$ we have :*

$$\min_{S\in\mathcal{S}} C^S_{\max} = \max\left\{ \max_{j\in J} p_j, \max_{N\subseteq M} \frac{\sum\limits_{j:\mu_j\subseteq N} p_j}{|N|} \right\},$$

where $\mathcal{S}$ is the set of the feasible schedules. There is a polynomial algorithm to find an optimal schedule.

Proof. First we will prove that the two quantities are lower bounds for the optimum, thereafter we show that if the makespan of the scheduling is not equal to the maximum of the lower bounds, we can correct the schedule. We also show that the correction steps can be organized to achieve a polynomial time algorithm.

Lemma 2.2. *The following values are lower bounds for the optimal value of the problem P MPM|pmtn|$C_{\max}$:*

$$p_{\max} = \max_{j \in J} p_j \tag{1}$$

$$\max_{N \subseteq M} \frac{\sum_{j:\mu_j \subseteq N} p_j}{|N|} \tag{2}$$

Proof. (1) is a lower bound, since a job cannot be processed by more than one machine. Take a subset of the machine-set and consider those jobs, that can be scheduled only on those machines. The average processing time, (2), is a lower bound too. □

Let us construct a network $G = (V, A)$ as follows. The set V consists of a source-node s, machine-nodes $M_1, M_2, \ldots, M_m$, job-nodes $J_1, J_2, \ldots, J_n$ and a sink-node t. The directed arc from node u to node v is denoted by uv. The set of arcs A consists of arcs sM_i with capacity ∞ $(i = 1, \ldots, m)$, arcs M_iJ_j, if $M_i \in \mu_j$ with capacity p_j $(i = 1, \ldots, m;\ j = 1, \ldots, n)$ and arcs J_jt with capacity p_j $(j = 1, \ldots, n)$. Figure 1 shows a part of this network.

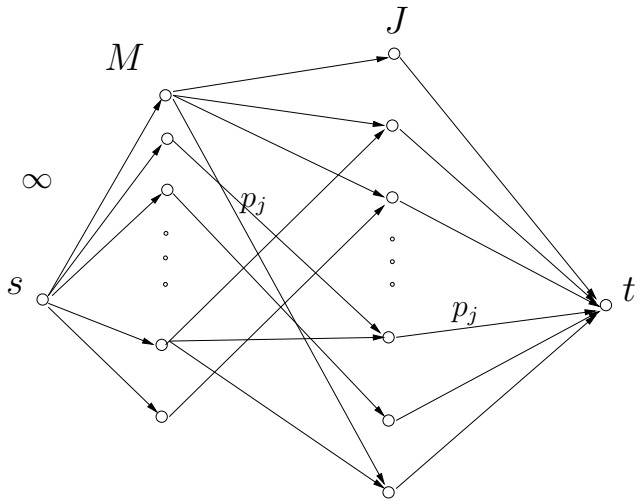


Figure 1: *Network*

Lemma 2.3. *A feasible schedule determines a feasible flow of value $\sum_{j=1}^{n} p_j$ in the corresponding network, and vice versa, a feasible flow of value $\sum_{j=1}^{n} p_j$ defines a feasible schedule.*

Proof. First consider a feasible schedule: let τ_{ij} be the processing time of job j on machine i in the schedule. Then

$$x(e) := \begin{cases} \tau_{ij}, & \text{if } e = M_iJ_j,\ i \in M, j \in J, \\ p_j, & \text{if } e = J_jt,\ j \in J, \\ \sum_{j\in J} \tau_{ij}, & \text{if } e = sM_i,\ i \in M \end{cases}$$

defines a flow with value $\sum_{j=1}^{n} p_j$. For the second part we construct a schedule from the flow values of the arcs going between node-set M and node-set J. Then we schedule $x(M_iJ_j)$ units of time of job j on machine i. This defines an instance of the classical open-shop problem $O|pmtn|C_{\max}$. The optimal value of the open shop problem is given by $\max\{\max_{j=1,\dots,n} p_j, \max_{i=1,\dots,m} \sum_{j=1}^{n} \tau_{ij}\}$, (see [4]). So the makespan of the schedule of the corresponding flow is $\max\{\max_{j=1,\dots,n} p_j, \max_{i=1,\dots,m} \sum_{j=1}^{n} \tau_{ij}\} = \max\{\max_{j=1,\dots,n} x(J_jt), \max_{i=1,\dots,m} x(sM_i)\} = \max\{p_{\max}, b(x)\}$, where $b(x) = \max_{i=1,\dots,m} x(sM_i)$. □

So the goal is to find a flow x of least $b(x)$, and then to show that the makespan of the corresponding schedule is equal to one of the lower bounds: take a feasible schedule S, and the corresponding network. Let us assume it is not an optimal schedule. Then the corresponding $C^S_{\max}$ is equal to neither of the two lower bounds. Let

$$N = \{i \in M : \sum_{e\in\delta^{in}(M_i)} x(e) = b(x)\},$$

where $\delta^{in}(v)$ denotes the set of arcs entering to node v. Let us construct an auxiliary graph $G' = (V', A')$: $V' = V \setminus \{t\}$, and A' consists of two type of arcs: "forward" arc sM_i if and only if $x(sM_i) < b(x)$, "forward" arc M_iJ_j if and only if $x(M_iJ_j) < p_j$, "backward" arc J_jM_i if and only if $x(M_iJ_j) > 0$.

Lemma 2.4. *If there exists a path from s to each node of N in G', then we can construct a flow x' in G with $b(x') < b(x)$.*

Proof. Consider a set of (not necessarily disjoint) paths from s to each node of N: $\mathcal{P} = \{P_1, P_2, \dots, P_{|N|}\}$. The set of arcs of $\mathcal{P}$ is denoted by $A'(\mathcal{P})$. Let us define capacities on the arcs of the paths:

$$c(e) := \begin{cases} \frac{b(x)-x(sM_i)}{l+1} & \text{if } e = sM_i \text{ is in } l \text{ paths} \\ \frac{p_j-x(M_iJ_j)}{l} & \text{if } e = M_iJ_j \text{ is in } l \text{ paths} \\ \frac{x(M_iJ_j)}{l} & \text{if } e = J_jM_i \text{ is in } l \text{ paths.} \end{cases}$$

Let

$$\varepsilon := \min_{e\in A'(\mathcal{P})} c(e).$$

Now modify the flow as follows:

$$x'(e) := \begin{cases} x(e) - \varepsilon & \text{if } e = sM_i,\ i \in N, \\ x(e) + l\varepsilon & \text{if } e = sM_i \in A'(\mathcal{P}) \text{ and } e \text{ is in } l \text{ paths}, \\ x(e) & \text{if } e = sM_i,\ e \notin A'(\mathcal{P}), \\ x(e) + l\varepsilon & \text{if } e = M_iJ_j,\ e \in A'(\mathcal{P}), \text{ and } e \text{ is in } l \text{ paths}, \\ x(e) & \text{if } e = M_iJ_j,\ e \notin A'(\mathcal{P}) \text{ and } J_jM_i \notin A'(\mathcal{P}), \\ x(e) - l\varepsilon & \text{if } e = M_iJ_j,\ J_jM_i \in A'(\mathcal{P}), \text{ and } J_jM_i \text{ is in } l \text{ paths} \\ p_j & \text{if } e = J_jt,\ j \in J. \end{cases}$$

x' is a flow in G, and
$b(x') = \max\limits_{M_i \in M} x'(sM_i) = \max\left\{\max\limits_{M_i \in N} x(sM_i) - \varepsilon,\ \max\limits_{M_i \in M \setminus N} x(sM_i) + \varepsilon\right\} = b(x) - \varepsilon.$ □

So if there exists such a set of paths, then we obtain a flow which corresponds to a schedule with better makespan.

Lemma 2.5. *If there does not exist paths from s to every node of N in G', then the corresponding schedule is optimal.*

Proof. The set of nodes of N for which there does not exist a path from s is denoted by N^*. The set of those job-nodes which are processed on machine of N^* is denoted by J^*, i.e.

$$J^* = \{j \in J : \exists i \in N^* :\ x(M_iJ_j) > 0\}.$$

Since there is no path from s to N^*, there cannot be a path from s to J^*, so there does not exist an arc from $M \setminus N^*$ to J^* in G'. That means that none of the nodes of J^* can be processed on machines from $M \setminus N^*$. [For contradiction, assume that for $j^* \in J^*$ there exists a machine $i \in M \setminus N^*$: $M_i \in \mu_{j^*}$. But for one of the nodes N^* we have $x(M_{i^*}J_{j^*}) > 0$, so $x(M_iJ_{j^*}) < p_{j^*}$. By the definition of the auxiliary graph, there must exist an arc from M_i to J_{j^*}, a contradiction.] So

$$\bigcup_{j \in J^*} \mu_j \subseteq N^*.$$

This implies:

$$b(x) \cdot |N^*| = \sum_{i \in N^*} x(sM_i) = \sum_{j : j \in J^*} p_j \leq \sum_{j : \mu_j \subseteq N^*} p_j,$$

so

$$b(x) \leq \frac{\sum\limits_{j : \mu_j \subseteq N^*} p_j}{|N^*|}.$$

From Lemma 2.2. we know that $b(x) \geq \max\limits_{N \subseteq M} \frac{\sum\limits_{j : \mu_j \subseteq N} p_j}{|N|}$ so $b(x)$ is equal to (2), thus the schedule must be optimal. We also have

$$\min_{S \in \mathcal{S}} C^S_{\max} = \max\left\{\max_{j \in J} p_j, \max_{N \subseteq M} \frac{\sum\limits_{j : \mu_j \subseteq N} p_j}{|N|}\right\}.$$

□

From Lemma 2.4. and 2.5. we can construct an algorithm:
Algorithm I

INPUT: An instance of problem $P\ MPM|pmtn|C_{\max}$, and a feasible schedule S.

OUTPUT: An optimal schedule S^*.

(S1) Construct the corresponding network G.

(S2) Build the auxiliary graph G'.

(S3) If there does not exist a path from s to each node of N in G', then the schedule is optimal. STOP

(S4) Find a set of paths from s to each node of N, $\mathcal{P} = \{P_1, P_2, \ldots, P_{|N|}\}$.

(S5) Modify the flow on the corresponding arcs. Go to (S2).

We prove that this algorithm is polynomial.

Lemma 2.6. *If we choose a shortest s-i ($i \in N$) path in G', then the number of steps of Algorithm I is polynomial.*

Proof. The length of the shortest s-v path is denoted by $\sigma(v)$. We divide the algorithm to phases, while the $\sigma = \max_{v \in N} \sigma(v)$ does not change. An arc is called tight if it is an element of a shortest sv path for some $v \in N$. If we improve the flow on the shortest-path set, then $\sigma(v)$ cannot decrease: by a modification some "forward" arcs will be deleted and some "forward" arcs can get into the graph as "backward" arc too. But new "forward" tight arc cannot arise. During an improvement at least one tight arc is deleted. Since there are at most $|A'|$ tight arcs, there are at most $|A'|$ modification in a phase. Since σ is at most $(|V'| - 1)$, there is at most $(|V'| - 1)$ phase, so if we denote $\max_{j \in J} |\mu_j|$ by $\mu_{\max}$, then in at most $\mathcal{O}(|A'||V'|) = \mathcal{O}(|J||\mu_{\max}| \cdot (|M| + |J|))$ steps the algorithm stops. □

In every step we have to find a shortest path from s to each $M_i \in N$, that takes $\mathcal{O}(|N||A'|^2)$ time, so we obtain a flow which corresponds to an optimal schedule in $\mathcal{O}(n\mu_{\max}(n+m) \cdot m(n\mu_{\max})^2)$ time. To get the schedule, we have to find an optimal solution of the classical open-shop problem $O|pmtn|C_{\max}$, which is solvable in $\mathcal{O}(r^2)$ time [4], where r is the number of the nonzero processing times p_{ij}. Observe that $\mathcal{O}(r^2) = \mathcal{O}((n\mu_{\max})^2)$, the overall complexity of the algorithm is $\mathcal{O}(n^3\mu_{\max}^3 m(n + m))$. □

3 $P\ MPM||C_{\max}$

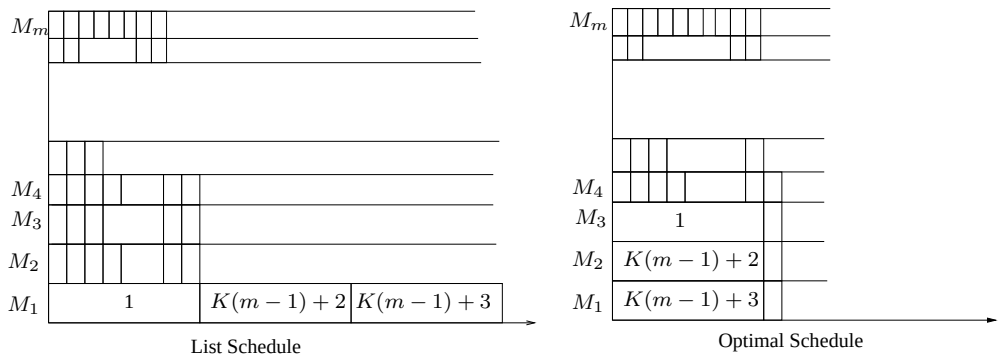
The problem is $\mathcal{NP}$-hard even if the sets μ_j are equal, Garey, Johnson [3]. The list-algorithm gives a 2-approximation for the problem $P||C_{\max}$ [5].

Example 3.1. The general list-scheduling algorithm gives approximate algorithm with factor worse than 2.

Proof. There are $K(m-1)+3$ jobs with the following parameters:

j	p_j	μ_j
1	K	$\{M_1, M_3\}$
2	1	M
$\vdots$	$\vdots$	$\vdots$
$K(m-1)+1$	1	M
$K(m-1)+2$	K	$\{M_1, M_2\}$
$K(m-1)+3$	K	$\{M_1\}$

Consider the following list and the schedule, that the list-scheduling algorithm gives: $L = \{1, 2, \ldots, K(m-1), K(m-1)+1, K(m-1)+2, K(m-1)+3\}$, then on the first machine we schedule the 1st job, job $K(m-1)+2$, and job $K(m-1)+3$ in this sequence, and the other jobs to the other machines. So $C_{\max} = 3K$. In the right side of the figure there is given an optimal schedule with makespan is $K + \frac{2K-3}{m}$.



□

We have a combinatorial 2-approximation algorithm from the algorithm *P MPM|pmtn|*$C_{\max}$. First we show, that a flow, which corresponds to an optimal schedule, may have special structure. Then from that special structure we build up a new 2-approximation algorithm.

Lemma 3.2. *For each instance of the problem P MPM|pmtn|*$C_{\max}$ *there exists an optimal solution, for which the following undirected bipartite graph* $G_+ = (M \cup J, A_+)$ *is a forest, where* $A_+ = \{M_iJ_j : \ x^*(M_iJ_j) > 0\}$,

Proof. Let us assume that there exists an instance of the problem, for which the statement of the lemma does not hold. Choose an optimal schedule, which has minimal number of cycles. Then we modify the corresponding flow as follows. Orient the cycle. Let $\epsilon = \min\limits_{e \in C} x(e)$. Decrease the flow by ϵ on the arcs of the cycle from M to J, and augment the flow by ϵ on the arcs of the cycle from J to M. This flow corresponds to an optimal schedule and the corresponding bipartite graph has less number of cycle which contradicts the assumption. □

Lemma 3.3. *Consider an optimal schedule, for which the corresponding G_+ is a forest. In that schedule there are at most $m-1$ preempted jobs.*

Proof. Consider those subgraph of the forest, which contains only those arcs, which corresponds preempted jobs. Each job-node in this graph has degree at least two, and while it is a forest in the bipartite graph, there must be at most $m-1$ job-node in this graph.

□

Algorithm II

INPUT: an instance of the problem $P\ MPM||C_{\max}$

OUTPUT: a feasible schedule S' for the problem with makespan $C_{\max}^{S'}$ such that if the optimal value of the makespan is $C_{\max}^*$, then $C_{\max}^{S'} \leq 2 \cdot C_{\max}^*$

(A1) Solve the corresponding $P\ MPM|pmtn|C_{\max}$ problem and choose the solution such that G_+ is a forest. (This is possible by Lemma 3.2).

(A2) Match the preempted jobs to the machines.

(A3) Make an arbitrary order of the jobs in the machines.

Theorem 3.4. *This algorithm gives a 2-approximation for the problem $P\ MPM||C_{\max}$.*

Proof. At the end of the algorithm we have a schedule S' with the following:

$$C_{\max}^{S'} \leq C_{\max}^{pmtn} + \max_{j \in J} p_j \leq C_{\max}^* + C_{\max}^* \leq 2C_{\max}^*,$$

where $C_{\max}^*$ is the optimal makespan of the problem, and $C_{\max}^{pmtn}$ is the optimal makespan of the corresponding preemptive scheduling problem. □

4 Special preemption

Preemption of a job means that processing may be interrupted and resumed later and maybe on another machine. Let us modify this rule: special preemption of a job means that processing must be resumed immediately on another machine, i.e. it cannot be resumed later. Certainly a job may be interrupted several times. We denote this special preemption rule by $pmtn^*$. Let us make some observations:

Lemma 4.1. *1. If $\mu_j = M$ for each $j = 1, \ldots, n$, then this special preemption rule does not help: the optimum of $P|pmtn^*|C_{\max}$ is equal to the optimum of $P||C_{\max}$.*

2. Problem $P\ MPM|pmtn^*|C_{\max}$ is $\mathcal{NP}$-hard.

3. The optimal solution of an instance of $P\ MPM|pmtn^*|C_{\max}$ can be strictly larger than the optimal makespan of the corresponding instance of $P\ MPM||C_{\max}$, and can be strictly less than the optimal makespan of the corresponding instance of $P\ MPM|pmtn|C_{\max}$.

Proof. (1) Take an optimal schedule of an instance of $P\ MPM|pmtn^*|C_{\max}$, which has minimal number of interruptions. Let j_1 be the first interrupted job in the schedule, and let it be scheduled first on machine i_1 and second on machine i_2. Then denote by $\mathcal{S}_{i_1}$ and $\mathcal{S}_{i_2}$ the jobs, which are scheduled on the machine i_1 and i_2 after the interruption. Then we exchange the two job-sequences, e.i. let us schedule the jobs of S_{i_1} on machine i_2, and jobs of S_{i_2} on machine i_1. We can do this, since $\mu_j = M$ for all $j \in J$. The obtained schedule S' is optimal, and has less interruption. It is a contradiction.
(2) Follows from (1), since $P||C_{\max}$ is an $\mathcal{NP}$-hard problem.
(3) Consider the following instance with 5 jobs:

j	p_j	μ_j
1	3	$\{M_2\}$
2	2	$\{M_2, M_3\}$
3	$1+\varepsilon$	$\{M_1, M_2\}$
4	2	$\{M_1, M_3\}$
5	1	$\{M_1\}$

For $P\ MPM||C_{\max}$, the optimal schedule has makespan $C^{S_1}_{\max} = 4$. (see Figure 2.)

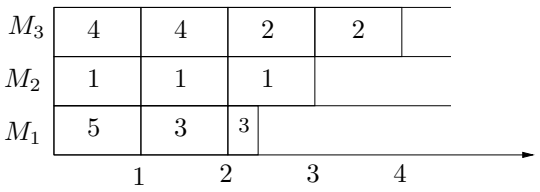


Figure 2: *Optimal schedule for* $P\ MPM||C_{\max}$

For $P\ MPM|pmtn^*|C_{\max}$, the optimal schedule has makespan $C^{S_s}_{\max} = 3 + \varepsilon$. (see Figure 3.)

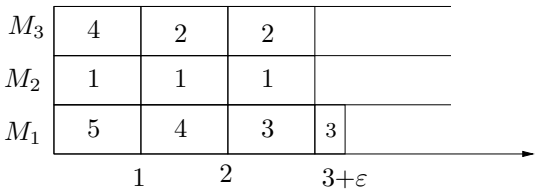


Figure 3: *Optimal schedule for* $P\ MPM|pmtn^*|C_{\max}$

For $P\ MPM|pmtn|C_{\max}$, the optimal schedule has makespan $C^{S_s}_{\max} = 3 + \frac{\varepsilon}{3}$. (See Figure 4.)

□

Theorem 4.2. *Algorithm II gives a 2-approximation for problem* $P\ MPM|pmtn^*|C_{\max}$*:*

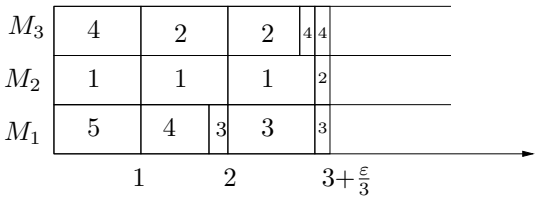


Figure 4: *Optimal schedule for* $P\ MPM|pmtn|C_{\max}$

Proof. The output schedule of the algorithm is a feasible schedule for the problem $P\ MPM|pmtn^*|C_{\max}$ too, and the lower bounds are lower bound for this problem too. □

References

[1] Peter Brucker. *Scheduling algorithms.* Springer-Verlag, Berlin, fourth edition, 2004.

[2] Peter Brucker, Bernd Jurisch, and Andreas Krämer. Complexity of scheduling problems with multi-purpose machines. *Ann. Oper. Res.*, 70:57–73, 1997.

[3] M. R. Garey and D. S. Johnson. "Strong" NP-completeness results: motivation, examples, and implications. *J. Assoc. Comput. Mach.*, 25(3):499–508, 1978.

[4] Teofilo Gonzalez and Sartaj Sahni. Open shop scheduling to minimize finish time. *J. ACM*, 23(4):665–679, 1976.

[5] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Tech. J.*, 45:1563–1581, 1966.

[6] Nicholas J. A. Harvey, Richard E. Ladner, László Lovász, and Tami Tamir. Semi-matchings for bipartite graphs and load balancing. In *Algorithms and data structures*, volume 2748 of *Lecture Notes in Comput. Sci.*, pages 294–306. Springer, Berlin, 2003.

[7] Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming*, 46(3, Ser. A):259–271, 1990.

[8] Michael Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall, New Jersey, second edition, 2002.