

# Scheduling with Special Case of Multipurpose Machines

Mourad Boudhar, Hamza Tchikou

Faculty of Mathematics, University USTHB, BP 32 Bab-Ezzouar, El-Alia 16111, Algiers, Algeria,  
{mboudhar, tch\_hamza}@yahoo.fr

We consider the problem of minimizing the makespan on  $m$  special multipurpose machines (called ordered parallel machines)  $M_1, \dots, M_m$  in which the execution of each job  $J_i$  ( $1 \leq i \leq n$ ) requires a time  $p_i$  and a machine among a subset  $M_{h_i}, \dots, M_m$  of machines. We prove the NP-hardness of the general problem and present some polynomial subproblems. Heuristics with an exact algorithm of branch and bound type are also presented with numerical experimentations.

*Keywords:* Complexity of Scheduling Problems, Heuristic Search, Machine Scheduling.

## 1 Introduction

We consider the problem of scheduling a set of jobs by allowing ordered machines (special case of multipurpose machines) to the jobs and fixing their starting times. The set of machines is denoted by  $M = \{M_1, M_2, \dots, M_m\}$ , where  $m$  represents the number of machines and the set of jobs is denoted by  $J = \{J_1, J_2, \dots, J_n\}$ , where  $n$  represents the number of jobs. The execution of each job  $J_i$  ( $1 \leq i \leq n$ ) requires a processing time  $p_i$  and a machine among a subset of machines  $\{M_{h_i}, \dots, M_m\}$  ( $h_i \in \{1, \dots, m\}$ ). The objective is to minimize the length of the scheduling (makespan) denoted by  $C_{max} = \max_{1 \leq i \leq N} \{C_i\}$  where  $C_i$  is the completion time of the job  $J_i$ .

As applications of the multipurpose machines cover a large class of industrial problems, our case considers one particular type of these problems which is the execution of a job requiring a machine among a subset of machines, such as if the job  $J_i$  ( $1 \leq i \leq n$ ) is candidate to be processed on the machine  $M_k$  it is also candidate to be processed on all the subset  $\{M_k, M_{k+1}, \dots, M_m\}$ . Example: Consider  $m$  machines  $M_1, M_2, \dots, M_m$  of guillotine type, where a machine  $M_j$  can cut only sheets having a given maximum length (called a threshold). Let us suppose that the machines are arranged in non decreasing order of their thresholds, therefore if a sheet can be cut on the third machine it can be also processed on the fourth, the fifth,  $\dots$ , the  $m$ -th machine.

The type of machines considered here is a subclass of the multipurpose machines. In a model of scheduling with multipurpose machines, denoted in the literature by "MPM", there exists a subset of machines  $\mu_i \subseteq \{M_1, \dots, M_m\}$  associated to the job  $J_i$  such as this one is processed on one of the machines of the subset  $\mu_i$ . If the multipurpose machines are parallel identical (resp. uniform), the corresponding situation is denoted by *PMPM* (resp. *QMPM*). One denotes *P* (resp. *Q*) the special case of *PMPM* (resp. *QMPM*) where all the subsets  $\mu_i$  contain all the  $m$  machines.

This article is organized as follows. In section 2 we give an analysis of the complexity of the problem. The section 3 gives some polynomial subproblems. An exact method is given in section 4 and some heuristics are given in section 5. Numerical experimentations are realized in section 6. A conclusion ends this paper.

## 2 Formulation and complexity

We start by proposing a mathematical formulation of the problem in the form of a linear program in binary variables. With this intention, we define the following variables and mathematical constraints: Let  $x_{ij}$  be the variables which indicate if the job  $J_i$  is processed on the machine  $M_j$ , therefore

$$x_{ij} = \begin{cases} 1 & \text{if } J_i \text{ is processed on } M_j \\ 0 & \text{if } \text{no} \end{cases}$$

for  $i = 1, \dots, n$  and  $j = h_i, \dots, m$

And let  $y$  be the variable which indicates the completion time of the set of the jobs ( $y = C_{max}$ ).

The mathematical model is written as follows:

$$\begin{array}{l} \min y \\ \text{s.t.} \left\{ \begin{array}{l} \sum_{i: h_i \leq j} x_{ij} p_i \leq y \quad \text{for } j = 1, \dots, m \\ \sum_{h_i \leq j \leq m} x_{ij} = 1 \quad \text{for } i = 1, \dots, n \\ x_{ij} \in \{0, 1\} \quad \text{for } i = 1, \dots, n ; j = h_i, \dots, m \\ y \geq 0 \end{array} \right. \end{array}$$

Let us denote by *MPMO2* (parallel multi-purpose machines ordered), the problem of two multi-purpose parallel machines, such that if a job  $J_i$  ( $i = 1, \dots, n$ ) can be processed on a machine  $M_1$ , it can also be processed on the machine  $M_2$ ; the inverse is not true. The problem *MPMO2*// $C_{max}$  is NP-hard because the problem *P2*// $C_{max}$ , which is a particular case ( $h_i = 1$  for all the jobs) is NP-hard. We will now show that the problem remains NP-hard even if the number of jobs to be processed on the second machine is constant and known in advance and all these jobs have an identical processing time.

**Theorem 1** *The problem *MPMO2*// $C_{max}$  is NP-hard even if the number of jobs processed on the second machine is fixed, and all these jobs have an identical processing time.*

**Remark 1** *As the problem *MPMO2*// $C_{max}$  (with only two machines) is NP-hard, then the problem *MPMO*// $C_{max}$  in its general form is also NP-hard.*

## 3 Some polynomial problems

In this section we will give some polynomial subproblems and some algorithms of resolution in polynomial time.

**Problem *PMPMO***  $p_i = 1/C_{max}$

For this problem the fact that the processing time of the jobs is equal to 1, allows us to propose a polynomial time algorithm to solve it.

Algorithm A1 ;

begin  $y := 0 ; nb := 0 ;$

for  $j := m$  down 1

do -  $X_j := \{J_i \in J / h_i = j\} ;$

- if  $X_j \neq \emptyset$

then -  $J := J \setminus X_j ; nb := nb + |X_j| ; y := \max\{y, \lceil nb / (m - j + 1) \rceil\} ;$

- schedule the jobs of  $X_j$  on the machines  $M_j, \dots, M_m$  with a finish time equal to  $y$  ;

endif

enddo ;

$$C_{max} := y$$

end.

At each iteration,  $X_j$  contains the jobs where  $h_i = j$ ;  $nb$  is the number of scheduled jobs and  $y$  is the completion time of processing of the scheduled jobs.

**Example 1** Let us process 9 jobs  $J_1, \dots, J_9$  with processing time equal to 1 on three machines  $M_1, M_2$  and  $M_3$  with  $h_1 = h_2 = h_3 = h_4 = h_5 = 1$ ,  $h_6 = h_7 = h_8 = 2$  and  $h_9 = 3$  in minimal time.

The execution of this algorithm gives:

First iteration:  $j = 3$ ;  $X_3 = \{J_9\}$ ;  $nb = 1$ ;  $y = \max\{0, \lceil 1/1 \rceil\} = 1$

Second iteration:  $j = 2$ ;  $X_2 = \{J_6, J_7, J_8\}$ ;  $nb = 4$ ;  $y = \max\{1, \lceil 4/2 \rceil\} = 2$

Third iteration:  $j = 1$ ;  $X_1 = \{J_1, J_2, J_3, J_4, J_5\}$ ;  $nb = 9$ ;  $y = \max\{2, \lceil 9/3 \rceil\} = 3$

This gives  $C_{max} = 3$ . An optimal schedule is given on figure 1.

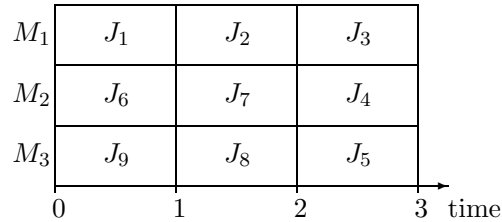


Figure 1: Solution of example 1

**Theorem 2** The algorithm A1 solves the problem  $MPMO/p_i = 1/C_{max}$  and in  $O(n \log n)$ .

**Scheduling on uniform machines**  $QMPMO/p_i = 1/C_{max}$

In this case, each machine  $M_j$  ( $1 \leq j \leq m$ ) has a speed  $S_j$  ( $S_j \geq 1$ ). The processing time of a job is, therefore, a standard processing time, i.e. that of a machine having a speed of processing equal to 1.

For this problem an approach of resolution can be the test of admissibility via the approach flow in a network. We build the network with two sets in the following way:

- The first set is that of the jobs of  $J$ .
- The second set is that of the machines of  $M$ .
- Each job  $J_i$  of  $J$  is connected to the source by an arc of capacity equal to 1.
- Each machine  $M_j$  of  $M$  is connected to the sink by an arc of capacity equal to  $b_j$
- If  $J_i$  can be processed on machine  $M_j$  then  $J_i$  is connected to  $M_j$  by an arc of capacity equal to 1.

It is seen that to find a maximum flow equal to  $n$  on the network with  $b_j = n$  for ( $1 \leq j \leq m$ ), corresponds to the construction of a realizable scheduling where all the jobs are assigned to the various machines.

Algorithm A2 ;

begin  $x := 0$ ;  $y := n \times \max_{1 \leq j \leq m} \{S_j\}$  ;

while  $x < y - 1$

do  $z := \lceil (x + y)/2 \rceil$  ;

- Apply the flow max. algorithm with  $b_j := \lfloor z/S_j \rfloor$  for  $j := 1, \dots, m$  ;

- If the maximum flow is equal to  $n$  then  $y := z$  else  $x := z$  endif

enddo ;  
 $C_{max} := \lceil (x + y)/2 \rceil$   
end.

$b_j := \lfloor z/S_j \rfloor$  corresponds to the maximum number of jobs which one can process on the machine  $M_j$  with makespan  $\leq z$ .

This algorithm is a dichotomic procedure of search on the interval  $[0, n \times \max_{1 \leq j \leq m} \{S_j\}]$ , it determines the smallest completion time  $C_{max}$  on this interval because  $0 < C_{max} \leq n \times \max_{1 \leq j \leq m} \{S_j\}$ .

**Theorem 3** *The algorithm A2 determines an optimal solution, in  $O(n^3 \log n \max_{1 \leq j \leq m} \{S_j\})$ , for problem QMPMO/ $p_i = 1/C_{max}$ .*

**Example 2** *Consider six jobs  $J_1, \dots, J_6$  to be processed on three machines  $M_1, M_2$  and  $M_3$  where the processed speeds of the machines are respectively 1, 2 et 3, under the constraints  $h_1 = h_2 = h_3 = 1, h_4 = h_5 = 2$  and  $h_6 = 3$ .*

The algorithm starts with the Construction of the network as figure 2 shows it below.

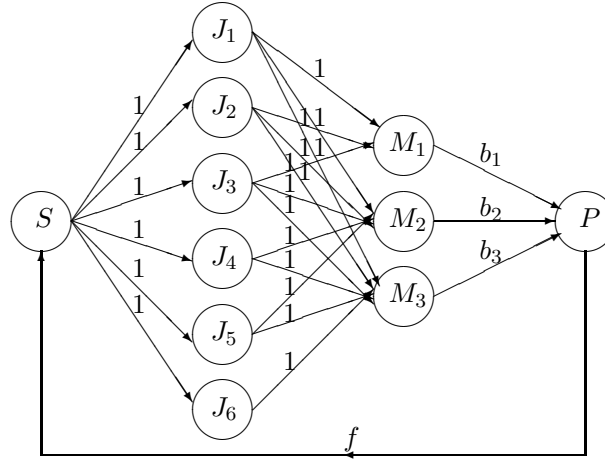


Figure 2: Network of example 2

Initialization :  $x := 0$  et  $y := n \times \max_{1 \leq j \leq 3} \{S_j\} = 18$

First iteration:  $z := 9$  ; application of the flow max. algorithm with  $b_1 = 9$  ,  $b_2 = 4$  and  $b_3 = 3$  ;  
 $f = 6 \implies y = 9$

Second iteration:  $z := 5$  ; application of the flow max. algorithm with  $b_1 = 5$  ,  $b_2 = 2$  and  $b_3 = 1$  ;  
 $f = 6 \implies y = 5$

Third iteration:  $z := 3$  ; application of the flow max. algorithm with  $b_1 = 3$  ,  $b_2 = 1$  and  $b_3 = 1$  ;  
 $f = 5 \implies x = 3$

Fourth iteration:  $z := 4$  ; application of the flow max. algorithm with  $b_1 = 4$  ,  $b_2 = 2$  and  $b_3 = 1$  ;  
 $f = 6 \implies y = 4$

Fifth iteration:  $y = x + 1$  ; The algorithm stops .

we stop the loop while with  $C_{max} = \lceil (x + y)/2 \rceil = 4$ . The optimal solution is given on figure 3.

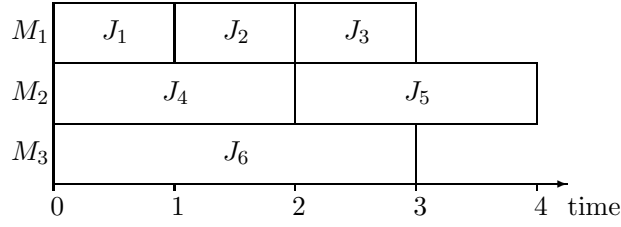


Figure 3: Solution of example 2

**Problem**  $PMPMO/pmtn/C_{max}$

In this part one illustrates a polynomial algorithm for the resolution of the problem  $PMPMO/pmtn/C_{max}$ . Let us note that the algorithm of Mc Naughton gives an optimal solution of the problem  $P/pmtn/C_{max}$  in  $O(n)$ .

Now we will propose an algorithm of resolution of the problem  $PMPMO/pmtn/C_{max}$ , by using the Mc Naughton algorithm as a procedure inserted in the algorithm, and as this procedure is of complexity  $O(n^2)$ , then its introduction into our algorithm will not negatively influence its complexity.

Let  $E = \{z/\exists J_i \in J \text{ with } h_i = z\}$ .

Algorithm A3 ;

begin -  $y := 0$  ;

- while  $E \neq \emptyset$

do - Choose the greatest index of  $E$  (let  $j$  be this index) ;

-  $E := E \setminus \{j\}$  ;  $F_j := \{J_i \in J/h_i = j\}$  ;  $J := J \setminus F_j$  ;

-  $y := \max\{y, \max_{i:h_i \geq j} \{p_i\}, (\sum_{i:h_i \geq j} p_i)/(m - j + 1)\}$  ;

- Apply the Mc Naughton algorithm to the jobs of  $F_j$  and the already scheduled jobs and the machines  $M_j, \dots, M_m$  with  $C_{max} = y$  ;

enddo ;

end.

**Theorem 4** *The algorithm A3 gives an optimal solution, in  $O(n^2)$ , for the problem  $PMPMO/pmtn/C_{max}$ .*

## 4 Exact Method

In this section we propose an exact algorithm of resolution for the problem of scheduling  $PMPMO/C_{max}$  which is certainly not polynomial, under the terms of the complexity of the problem.

**Lemma 5** *The value  $P_{max} = \max_{1 \leq i \leq n} \{p_i\}$  represents a lower bound for the optimal solution of the problem  $PMPMO/C_{max}$ .*

**Lemma 6** *The value  $\max_{j \in E} \left\{ \left( \sum_{i:h_i \geq j} p_i \right) / (m - j + 1) \right\}$  represents a lower bound for the optimal solution of the problem  $PMPMO/C_{max}$ .*

Therefore according to these two preceding lemmas, one can state the following theorem:

**Theorem 7** For the problem  $PMPMO//C_{max}$  we have:

$$C_{max}^* \geq \max\left\{\max_{1 \leq i \leq n} \{p_i\}, \max_{j \in E} \left\{ \left( \sum_{i: h_i \geq j} p_i \right) / (m - j + 1) \right\}\right\}$$

In what follows we will use a branch and bound method to solve the considered problem. For that, one defines the following procedures.

- a. Evaluation procedure: To determine the evaluation function  $g$  one uses the result of the preceding theorem:

$$g = \max\left\{\max_{1 \leq j \leq m} \{\bar{p}_j\}, \max_{1 \leq i \leq n} \{p_i\}, \max_{j \in E} \left\{ \left( \sum_{i: h_i \geq j} p_i \right) / (m - j + 1) \right\}\right\}$$

where  $\bar{p}_j$  is the sum of the processing times of the jobs assigned to the machine  $M_j$ . The evaluation procedure enables us to build a scheduling with length lower or equal to  $g$ . For that, one supposes that a list of priorities is given to the jobs not yet assigned as a LPT (Longest Processing Time) rule, and to each stage, the first available machine is selected to process the first free job of the list such as the total processing time, on the considered machine, be less or equal to  $g$ .

If a job  $J_i$  cannot be assigned to a machine  $M_j$  so that  $\bar{p}_j + p_i \leq g$ , then  $J_i$  is shared between the machines having an index  $\geq h_i$ , and the solution obtained is not feasible. The scheduling obtained is of length equal to  $g$ .

- b. Separation procedure: On each of the nodes of the tree structure, define  $y$  as the best exact evaluation (associated to a feasible solution) found and  $g$  is the calculated evaluation on the considered node.

If  $g$  is larger than  $y - 1$ , it is not necessary to explore more this node, since there cannot be an optimal solution of cost lower than  $y - 1$  starting from this node.

Else "  $g \leq y - 1$  " one looks whether the obtained solution associated to the evaluation  $g$  is feasible; if this is the case one assigns to  $y$  the value of  $g$ ; if not, one separates as follows:

Let  $J_i$  be the job which caused the non feasibility of the solution. Starting from the considered node one will create  $|m - h_i + 1|$  nodes such that:  $J_i$  is assigned to  $M_{h_i}$  for the first node,  $J_i$  is assigned to  $M_{h_i+1}$  for the second node,  $\dots$ ,  $J_i$  is affected to  $M_m$  for the  $|m - h_i + 1|$ th node.

Let us note that we use a backtracking exploration of the tree structure.

The algorithm A4 is composed of three procedures:

Procedure initialization ;

begin -  $M_1^*, \dots, M_m^* := \emptyset$  ; {no job is assigned to the machines}

-  $JC := J$  ;  $y := +\infty$  ;  $\bar{p}_1 := 0; \dots; \bar{p}_m := 0$

end ;

Procedure evaluation(input  $M_1^*, \dots, M_m^*, JC$  ; input/output  $g$  ; output  $S, bool$ ) ;

begin -  $g = \max\left\{\max_{1 \leq j \leq m} \{\bar{p}_j\}, \max_{1 \leq i \leq n} \{p_i\}, \max_{j \in E} \left\{ \left( \sum_{i: h_i \geq j} p_i \right) / (m - j + 1) \right\}\right\}$  ;

- Assign the jobs of  $JC$  to the machines of  $M$  in the non increasing order of  $h_i$  (and according to the non increasing order of the processing times in the case of equality) such that:  $\bar{p}_j \leq g \forall j = 1, \dots, m$  ;

- if a job  $J_i$  cannot be assigned to a machine  $M_j$  ( $\bar{p}_j > g$ )

then  $S := i$  ;  $bool := false$  else  $bool := true$

endif

end ;

Procedure SE(input  $M_1^*, \dots, M_m^*, JC$  ; input/output  $y$ );

begin - evaluation ( $M_1^*, \dots, M_m^*, JC, g, S, bool$ ) ;

- if  $g < y$

```

then if  $bool = true$ 
  then  $y := g$  ;  $\overline{M}_1 := M_1^*, \dots, \overline{M}_m := M_m^*$ 
  else {separation}
    -  $j := h_S$  ;  $JC := JC \setminus \{J_S\}$  ;
    - while  $(j \leq m)$  and  $(g < y)$ 
      do -  $M_j^* := M_j^* \cup \{J_S\}$  ;  $\overline{p}_j := \overline{p}_j + p_j$  ;
        -  $SE(M_1^*, \dots, M_m^*, JC, y)$  ;
        -  $M_j^* := M_j^* \setminus \{J_S\}$  ;  $\overline{p}_j := \overline{p}_j - p_j$  ;  $j := j + 1$ 
      enddo
    endif
  endif
end ;
  This algorithm is of course finite.

```

## 5 Heuristics

The exponential exact methods are non efficient for the NP-hard problems, for this reason polynomial methods have appeared, which not necessarily give an optimal solution but seek by various techniques the best possible approaching solution.

**Problem**  $PMPMO//C_{max}$

To solve the problem  $PMPMO//C_{max}$  we will use the algorithm of the problem  $PMPMO/pmtn/C_{max}$  to develop an algorithm which gives an 2-approximation, i.e. a feasible scheduling  $S'$  such as  $C_{max}^{S'} \leq 2 \cdot C_{max}^*$ .

Algorithm A5 ;

begin - Solve the problem  $PMPMO/pmtn/C_{max}$  by using the algorithm A3;

- Reschedule preempted jobs on all machines while respecting feasibility of the solution

end.

**Theorem 8** *The algorithm A5 gives an 2-approximation for the problem  $PMPMO//C_{max}$ .*

In the continuation of this part, we will propose an heuristic of resolution, which gives a result better than that of the algorithm A5, and in  $O(n \log n + nm)$ . To this end, we use an LPT arrangement by giving a certain order of processing of the jobs on the machines, according to the feasibility of the imposed constraints.

Algorithm A6 ;

begin while  $E \neq \emptyset$

do - Choose the greatest index of  $E$  (let  $j$  be this index) ;

-  $E := E \setminus \{j\}$ ;  $F_j := \{J_i \in J / h_i = j\}$  ;  $J := J \setminus \{F_j\}$  ;

- Arrange the jobs of  $F_j$  according to the rule LPT;

- for  $k = 1$  to  $|F_j|$

do Schedule the first job, not yet processed, of the set  $F_j$ ,

on the first free machine among the machines  $M_j, \dots, M_m$ .

enddo

enddo

end.

**Theorem 9** *The heuristic A6 runs in  $O(n \log n + nm)$ .*

## 6 Numerical experimentations

Algorithms proposed in this paper (heuristics and exact method) for the problem  $PMPM/ /C_{max}$  were tested and compared on instances generated randomly according to an uniform law. Several numbers of machines ( $m \in \{4, 6, 12, 20\}$ ) and jobs ( $n \in \{7, 10, 15, 20, 25, 30, 35, 40, 50, 60, 80, 100\}$ ) were used for the various tests. Some numerical results obtained are given and summarized.

The various tests realized (on a pentium IV) have shown that for instances of small size, the optimal solution is found quickly in a reasonable time. The approached solution is to be very near the optimum with a ratio of performance lower than 1.2 in the majority of the cases and lower than 1.1 in more than 90 % of the cases. For the instances of big size, the exact method did not enable us to determine the optimal solution but the results obtained confirm the efficiency of the heuristics A6.

## 7 Conclusion

In this paper, we have shown that the problem of minimization of the makespan on ordered parallel machines is NP-hard. We have presented exact polynomial algorithms to solve some easy subproblems. Two heuristics and an exact method with numerical experiments are also presented. Let us note that the heuristic ones are generally used to tackle with this type of NP-hard problems.

## References

- [1] J. Blazewicz (1987), Selected topics in scheduling theory, *Annals of discrete mathematics* **31**, 1 – 60.
- [2] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt and J. Weglarz (1996), *Scheduling computer and manufacturing processes*, Springer-Verlag, Berlin.
- [3] J. Blazewicz, J. Lazewicz, K.H. Ecker, G. Schmidt and J. Weglarz (1994), *Scheduling in computer and manufacturing systems*, Springer-Lehrbuch, Berlin.
- [4] M. Boudhar (1996), *Sur quelques problèmes d'ordonnancement d'atelier*, thèse de magister, USTHB-Algiers.
- [5] P. Brucker (1995), *Scheduling algorithms*, Springer-Verlag, Berlin.
- [6] K.R. Baker (1974), *Introduction to sequencing and scheduling*, John Wiley & Sons, New York.
- [7] P. Brucker and S. Knust (2006), *Complexity results of scheduling problems*, page web: <http://www.mathematik.uni-osnabrueck.de/research/OR/class/>.
- [8] J. Carlier and P. Chretienne (1988), *Problèmes d'ordonnancement : modélisation, complexité et algorithmes*, Masson, Paris.
- [9] M.R. Garey and D.S. Johnson (1979), *Computers and intractability: a guide to the theory of NP-Completeness*, W.H. Freeman and Company, San Francisco.
- [10] V. Zsuzsanna (2005), On scheduling problems with parallel multi-purpose machines, *EGRES Technical Report TR-2005-02*, EtvS University, Budapest.