RAIRO Operations Research
RAIRO Oper. Res. **36** (2002) 149-156
DOI: 10.1051/ro:2002008

SCHEDULING JOBS IN OPEN SHOPS WITH LIMITED MACHINE AVAILABILITY *, **

Jacek Błażewicz[1] and Piotr Formanowicz[1]

Communicated by Bernard Lemaire

Abstract. In this paper, open shop scheduling problems with limited machine availability are studied. Such a limited availability of machines may appear in many real-life situations, *e.g.* as preventive maintenance activities. Three types of jobs are distinguished: non-preemptable, resumable and preemptable. An operation of a resumable job if not completed before a non-availability period of a machine may be suspended and continued without additional cost when the machine becomes available. In the paper, results are given for the scheduling problems associated with the three types of jobs. For preemptable jobs polynomial-time algorithms based on the two-phase method are proposed.

Keywords: Limited machine availability, computational complexity, two-phase method, mathematical programming.

1. Introduction

In the deterministic scheduling theory it is usually assumed that machines are continuously available for processing jobs. Most of the scheduling models and

Received November, 2000.

* *This paper stems from the summer school ODIP'2000 (Ordonnancement Déterministe pour l'Informatique et la Production), orgnanized at the CNRS Centre of Aussois, in September 2000, by Alix Munier, Stéphane Dauzère Pérès and Philippe Chrétienne.*

** *This research has been partially supported by KBN grant.*

[1] Institute of Computing Science, Poznań University of Technology, Piotrowo 3A, 60-965 Poznań, Poland and Institute of Bioorganic Chemistry, Polish Academy of Sciences, Noskowskiego 12/14, 61-704 Poznań, Poland; e-mail: Piotr.Formanowicz@cs.put.poznan.pl

algorithms include this assumption, which is difficult to justify in practice. In many situations, it may happen that machines are not available for processing in certain periods of time. Such a limited availability may result from preschedules, preventive maintenance or from an application of the rolling time horizon planning algorithms [3]. The rolling horizons are used mainly because most of the real world production planning problems are dynamic. This means that the input data are being frequently updated. A natural approach to handling this dynamic is to trigger a new planning horizon when the changes in the data justify it. However, due to many constraints, such as process preparation for instance, it is necessary to take earlier plans as fixed which obviously limits availability of machines for any subsequent schedules. The processors of a computer environment also may be not available. Consider for example a system with two sets of jobs to be processed. Jobs from one of these sets have high priority and have to be scheduled in certain time intervals. So, the scheduling procedure should assign processors to these jobs first. When planning an execution of remaining jobs, the procedure has to consider periods in which processing of high priority jobs will take place as periods of processor non-availability.

Problems of scheduling jobs on parallel machines with limited machine availability attracted relatively more researchers' attention than other problems of this type and have been studied for example in [2, 4, 5, 7, 9, 14, 16, 17].

In the present paper we consider problems of scheduling jobs in open shop systems. To our best knowledge, no papers concern problems of this type, only a few ones deal with flow shop systems. In paper [11] a concept of *resumable* tasks has been introduced. Tasks of this type may be preempted only at the starting point of non-availability period and then its processing is continued without additional cost when the machine becomes available. In [12] it has been shown that scheduling resumable jobs in a two-machine flow shop system with one period of non-availability is an NP-hard problem. In [6, 10] it has been proved that the more general problem with an arbitrary number of non-availability periods on one of the machines is strongly NP-hard. Also in [6, 10] a branch and bound method for that problem has been proposed, while in [1] some heuristic approaches have been presented.

Good overviews of scheduling jobs on machines with limited availability are given in [6, 15, 18].

In what follows three types of jobs are distinguished: non-preemptable, resumable and preemptable. In the paper, results are given for the scheduling problems associated with the three types of jobs. For preemptable jobs, for which the job constraint is that at any time no two pieces of the same job can be processed simultaneously, polynomial-time algorithms based on the two-phase method are proposed.

The organization of the paper is as follows. In Section 2 we provide problem formulation and notation. In Sections 3, 4 and 5 respectively, the special case of non-preemptable, resumable and preemptable jobs is considered. The paper ends with conclusions in Section 6.

2. Problem formulation and notation

We have a set $\mathcal{M}=\{M_1, M_2, \ldots, M_m\}$ of m machines and a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n jobs. Each job J_j consists of m operations $O_{1j}, O_{2j}, \ldots, O_{mj}$ and operation O_{ij} has to be executed on machine M_i. For each operation O_{ij} there is given its processing time p_{ij} and possibly its ready time r_{ij} and due date d_{ij}. Moreover, let t_{ij} and c_{ij} denote, respectively, starting time and completion time of operation O_{ij}. Each job may be processed by only one machine at a time and each machine may process only one job at a time. Each machine is not available for processing jobs in certain time intervals. We will call such intervals *holes* for convenience. Let us denote by s_{ik} and h_{ik} the starting time and the length of k-th hole on machine M_i, respectively. Moreover, let K_i be the number of holes on machine M_i. Note, that in open shops the machine indices can be permuted, since there is no fixed order of the jobs' flow through the shop and there are no precedence constraints among operations of the jobs.

To denote problems with limited machine availability we will use a modification of the standard three-field notation proposed in [2, 6, 10]. Here, in the first field symbol h_{ik} is used to denote that there may be an arbitrary number of holes on each machine. If k is replaced by an integer it denotes a given number of holes on every machine. Similarly, if i is replaced by an integer, it denotes a number of machine on which the holes occur. In the second field of the standard notation we will use symbol rs to denote resumable jobs. For example $O2, h_{1k}|rs|C_{\max}$ denotes the problem of scheduling resumable jobs in a two-machine open shop system with arbitrary number of holes on the first machine in order to minimize schedule length. $O, h_{ik}|pmtn|L_{\max}$ denotes the problem of scheduling preemptable jobs in an open shop system with an arbitrary number of machines and an arbitrary number of holes on each machine, in order to minimize maximum lateness.

At the end of this section we will provide formulation of a Partition problem [8] which will be used in proofs of Section 4.

Partition

Parameters: set $A = \{a_1, a_2, \ldots, a_n\}$ n of positive integers and positive integer B such that $\sum_{i=1}^{n} a_i = 2B$.

Question: "can set A be partitioned into disjoint sets A_1 and A_2 such that $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i = B$?"

3. Non-preemptable jobs

According to the following theorem problems of scheduling non-preemptable jobs in the two-machine open shop system with one hole only are intractable.

Theorem 1. *Problems* $O2, h_{11}||C_{\max}$ *and* $O2, h_{21}||C_{\max}$ *are NP-hard.*

Proof. Since problem $1, h_1||C_{\max}$ is NP-hard [11] problems $O2, h_{11}||C_{\max}$ and $O2, h_{21}||C_{\max}$ are also NP-hard. To see this let us consider any instance of the single machine problem. We can create a corresponding instance of the open shop problem where operations processed by a machine which is not-continuously available have processing times equal to those in single machine problem. Operations performed by the other machine (without non-availability period) have equal processing times less than the shortest processing time in the single machine problem. It is clear that in this case if we have a positive answer for the single machine problem, we also have such an answer for the open shop and *vice versa*. □

If there are an arbitrary number of holes these problems become strongly NP-hard.

Theorem 2. *Problems* $O2, h_{1k}||C_{\max}$ *and* $O2, h_{2k}||C_{\max}$ *are NP-hard in the strong sense.*

Proof. Since problem $1, h_k||C_{\max}$ is NP-hard in the strong sense [11], problems $O2$, $h_{1k}||C_{\max}$ and $O2, h_{2k}||C_{\max}$ are also strongly NP-hard. The proof is analogous to the previous one. □

4. Resumable jobs

Similarly like in the case of non-preemptable jobs, scheduling resumable jobs in the two-machine open shop system with one hole is an intractable problem.

Theorem 3. *Problem* $O2, h_{11}|rs|C_{\max}$ *is NP-hard.*

Proof. The proof will be done by transforming Partition into the decision version of problem $O2, h_{11}|rs|C_{\max}$.

Given an instance of Partition, let us create an instance of the scheduling problem in the following way. There are n jobs with processing requirements $p_{1j} = a_j$ and $p_{2j} = n \max_{1\le j\le n}\{a_j\}$ for $j = 1, \ldots, n$. Moreover, there is a hole of length $h_{11} = n^2 \max_{1\le j\le n}\{a_j\}$ on the first machine which begins at $s_{11} = B$.

The question is: "is there any feasible schedule of length not greater than $2B + n^2 \max_{1\le j\le n}\{a_j\}$?".

$\Rightarrow$ If the instance of Partition has a solution, then there exists a feasible schedule of length $C_{\max} = 2B + n^2 \max_{1\le j\le n}\{a_j\}$. It can be created by scheduling first-machine operations corresponding to set A_1 from Partition before a hole and remaining ones after it. Second-machine operations are scheduled "under" the hole as in Figure 1.

$\Leftarrow$ Let S be an optimal schedule of the jobs. The makespan $M(S)$ of S is at least $2B + n^2 \max_j\{a_j\}$. So we have $M(S) = 2B + n^2 \max_j\{a_j\}$. Assume the job J_1 is resumed on the first machine, then job J_1 cannot be processed on the second machine since there is not enough time on that machine before the starting

time of J_1 or after the completion time of J_1. Since S has no idle time on the first machine, the sum of the processing times of the tasks performed before the hole on the first machine is equal to the sum of the processing times of the tasks performed after the hole on the first machine. So the instance of PARTITION has a solution. □

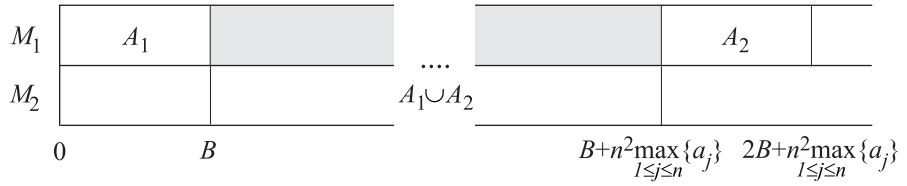

FIGURE 1. Schedule with one hole on the first machine which corresponds to a partition.

5. PREEMPTABLE JOBS

It turns out that preemptions result in polynomial-time solvability of the problems considered in the previous sections. Moreover, it is possible to solve in polynomial time problems of scheduling jobs on arbitrary number of machines. In this section two algorithms will be proposed. The first algorithm can be used to solve problem $O, h_{ik}|pmtn|C_{\max}$. This algorithm is a variant of the two-phase method [3]. In the first phase linear programming problem is solved. Let $0 < \omega_1 < \omega_2 < \ldots < \omega_q$ be a list of moments when availability of machines changes. In the linear programming formulation some additional denotations will be used:

$e_0 < e_1 < \ldots < e_\kappa$ – a list of different moments in $\{0, C_{\max}\} \cup \{\omega_l : l = 1, \ldots, q\}$;
$x_{ij}^{(l)}$ – part of operation O_{ij} processed on machine M_i in interval $[e_{l-1}, e_l]$;
$x_{ij}^{(l)} \in [0,1]$ for $i = 1, \ldots, m$ and $l = 1, \ldots, \kappa$;
$x_{ij}^{(l)} = 0$ for all l for which machine M_i is not available in $[e_{l-1}, e_l]$.

Now, we can formulate the linear programming problem:

$$\text{minimize } C_{\max} \tag{1}$$

subject to:

$$\sum_{i=1}^{m} p_{ij} x_{ij}^{l} \le e_l - e_{l-1}, \quad j = 1, \ldots, n, \quad l = 1, \ldots, \kappa \tag{2}$$

$$\sum_{j=1}^{n} p_{ij} x_{ij}^{l} \leq e_l - e_{l-1}, \quad i = 1, \ldots, m, \quad l = 1, \ldots, \kappa \tag{3}$$

$$\sum_{l=1}^{\kappa} x_{ij}^{l} = 1, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n. \tag{4}$$

Inequalities (2) guarantee that the sum of parts of any job processed in a particular interval on all machines does not exceed the length of this interval. Similarly, constraints (3) guarantee that the total amount of work done in any interval by particular machine does not exceed the length of the interval. Equations (4) ensures that every operation will be fully executed by a proper machine.

As a result of solving the above linear programming problem an assignment of operations to intervals $[e_{l-1}, e_l]$, $l = 1, \ldots, \kappa$ is obtained. Note, that this assignment need not to constitute a feasible schedule because of a possible parallel execution of operations belonging to one job. In order to obtain an optimal schedule the second phase of the method has to be used [3, 6].

In the second phase an optimal assignment of the job parts to machines in time is determined. This is done by using a procedure based on a network flow, which is applied to every interval considered in the first phase (see [3, 6]).

Now, let us consider problem $O, h_{ij}|pmtn, r_j|L_{\max}$. It can be solved in polynomial time by a combination of the two-phase method and a binary search procedure. The algorithm searches for an interval $[L', L'']$ in which an optimal value of $L_{\max}$ is located. Points in time when holes begin or end and ready time of jobs and their due dates increased by $L_{\max}$ create intervals in which jobs are processed. To each of the intervals $[L', L'']$ of $L_{\max}$ values there corresponds a fixed sequence of the above mentioned events, from which a formulation of linear programming problem is possible. As a result of solving this problem an answer to the question whether an optimal value of $L_{\max}$ is located in a given interval is obtained. If the answer is negative the binary search procedure is continued.

Let $e_0 < e_1 < \ldots < e_\kappa$ be a list of different moments in $\{r_j, d_j + L_{\max} : j = 1, \ldots, n\} \cup \{\omega_l : l = 1, \ldots, q\} \cup \{0\}$, $A_j = \{l : e_{l-1} \geq r_j \text{ i } e_l \leq d_j + L_{\max}\}$, $B_l = \{j : e_{l-1} \geq r_j \text{ i } e_l \leq d_j + L_{\max}\}$.

The corresponding linear program is:

$$\text{minimize } L_{\max} \tag{5}$$

subject to:

$$\sum_{i=1}^{m} p_{ij} x_{ij}^{l} \leq e_l - e_{l-1}, \quad j = 1, \ldots, n, \quad l \in A_j \setminus \{0\} \tag{6}$$

$$\sum_{j\in B_l} p_{ij}x_{ij}^l \le e_l - e_{l-1}, \quad i = 1,\dots,m, \quad l = 1,\dots,\kappa \tag{7}$$

$$L' \le L_{\max} \le L'' \tag{8}$$

$$\sum_{l\in A_j\setminus\{0\}} x_{ij}^l = 1, \quad i = 1,\dots,m, \quad j = 1,\dots,n. \tag{9}$$

Inequalities (6) and (7) ensure that an amount of work done in any interval does not exceed its capacity. Inequalities (8) guarantee that an optimal value of $L_{\max}$ is searched for only in a given interval. Thanks to (9) every operation is done by a proper machine.

Next, a binary search procedure is used to find an interval $[L', L'']$ containing an optimal value of $L_{\max}$.

The second phase of the two-phase method is used in the last stage of binary search procedure and gives a schedule with a minimal value of $L_{\max}$.

Using binary search procedure the linear programming problem (5–9) must be solved $O(\log n + \log q)$ times.

6. Conclusions

In this paper open shop systems with limited machine availability have been considered. Scheduling problems where it is not assumed that machines are continuously available are of great practical importance. Three types of jobs have been considered, *i.e.* non-preemptable, resumable and preemptable. Resumable jobs are specific for problems with limited machine availability. They can be preempted only at a starting point of machine non-availability period and then continued on the same machine, when it becomes available, without any additional cost. We showed that if there is only one non-availability period, problems of scheduling non-preemptable or resumable jobs in two-machine open shop are NP-hard. If there is an arbitrary number of such periods on one of the machines, scheduling non-preemptable jobs becomes a strongly NP-hard problem. On the other hand, if jobs can be preempted at any time problems become easy. We proposed two polynomial-time algorithms based on the two-phase method for scheduling preemptable jobs in open shops with an arbitrary number of machines and an arbitrary number of non-availability periods on each of them. An open question remains if scheduling resumable jobs in two-machine open shop with an arbitrary number of non-availability periods on one of the machines is a strongly NP-hard problem.

References

[1] J. Błażewicz, J. Breit, P. Formanowicz, W. Kubiak and G. Schmidt, Heuristic algorithms for the two-machine flowshop with limited machine availability. *Omega - International J. Management Sci.* **29** (2001) 599-608.

[2] J. Błażewicz, M. Drozdowski, P. Formanowicz, W. Kubiak and G. Schmidt, Scheduling preemtable tasks on parallel processors with limited availability. *Parallel Comput.* **26** (2000) 1195-1211.
[3] J. Błażewicz, K. Ecker, E. Pesch, G. Schmidt and J. Węglarz, *Scheduling Computer and Manufacturing Processes*, 2nd Edition. Springer-Verlag, Berlin (2001).
[4] D. Dolev and M.K. Warmuth, Profile scheduling of opposing forests and level orders. *SIAM J. Algebraic Discrete Meth.* **6** (1985) 665-687.
[5] D. Dolev and M. Warmuth, Scheduling flat graphs. *SIAM J. Comput.* **14** (1985) 638-657.
[6] P. Formanowicz, Scheduling jobs in systems with limited availability of processors (in Polish), Ph.D. Thesis. Institute of Computing Science, Poznań University of Technology, Poznań (2000).
[7] P. Formanowicz, Selected deterministic scheduling problems with limited machine availability. *Pro Dialog* **13** (2001) 91-105.
[8] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, San Francisco (1979).
[9] M.R. Garey, D.S. Johnson, R.E. Tarjan and M. Yannakakis, Scheduling opposing forests. *SIAM J. Alg. Disc. Meth.* **4** (1983) 72-93.
[10] W. Kubiak, J. Błażewicz, P. Formanowicz, J. Breit and G. Schmidt, Two-machine flow shops with limited machine availability. *Eur. J. Oper. Res.* **136** (2002) 528-540.
[11] C.-Y. Lee, Machine scheduling with an availability constraint. *J. Global Optim.* **9** (1996) 395-416.
[12] C.-Y. Lee, Minimizing the makespan in the two-machine flow shop scheduling problem with an availability constraint. *Oper. Res. Lett.* **20** (1997) 129-139.
[13] C.-Y. Lee, Two-machine flowshop scheduling with availability constraints. *Eur. J. Oper. Res.* **114** (1999) 420-429.
[14] Z. Liu and E. Sanlaville, Preemptive scheduling with variable profile, precedence constraints and due dates. *Discrete Appl. Math.* **58** (1995) 253-280.
[15] E. Sanlaville and G. Schmidt, Machine scheduling with availability constraints. *Acta Informatica* **35** (1998) 795-811.
[16] G. Schmidt, Scheduling on semi-identical processors. *Z. Oper. Res. A* **28** (1984) 153-162.
[17] G. Schmidt, Scheduling independent tasks with deadlines on semi-identical processors. *J. Oper. Res. Soc.* **39** (1988) 271-277.
[18] G. Schmidt, Scheduling with limited machine availability. *Eur. J. Oper. Res.* **121** (2000) 1-15.