Operations Research Letters 51 (2023) 187–189

Contents lists available at ScienceDirect

Operations Research Letters

journal homepage: www.elsevier.com/locate/orl

On the complexity of scheduling unrelated parallel machines with limited preemptions

Jan Karel Lenstra [a,*], Nodari Vakhania [b]

[a] Centrum Wiskunde & Informatica, Amsterdam, the Netherlands
[b] Universidad Autónoma del Estado de Morelos, Cuernavaca, Mexico

ARTICLE INFO

Article history:
Available online 10 February 2023

Keywords:
Scheduling
Unrelated parallel machines
Preemption
NP-hardness

ABSTRACT

We consider the problem of finding a minimum-length preemptive schedule for n jobs on m parallel machines. The problem is solvable in polynomial time, whether the machines are identical, uniform or unrelated. For identical or uniform machines, it is easy to obtain an optimal schedule in which the portion of a job that is assigned to a single machine is processed without interruption. We show that imposing this condition in the case of unrelated machines makes the problem NP-hard.

A set of n jobs has to be processed on a set of m parallel machines so as to minimize the length of the schedule or the makespan. The jobs and the machines are available from time 0 onwards. A machine can process at most one job at a time and a job can be processed by at most one machine at a time. Job preemption is allowed: the processing of a job may be interrupted and resumed on the same or on a different machine. There are three models, depending on the definition of processing times. On *identical* machines, job j requires time p_j on any machine ($j = 1, \ldots, n$); on *uniform* machines, job j has a processing requirement p_j, machine i has a speed s_i, and processing job j on machine i takes time p_j/s_i; on *unrelated* machines, job j requires time p_{ij} on machine i ($i = 1, \ldots, m$, $j = 1, \ldots, n$). Using the notation introduced by Graham et al. [7], we denote these problems by $P|pmtn|C_{\max}$, $Q|pmtn|C_{\max}$, and $R|pmtn|C_{\max}$.

Minimizing the makespan on unrelated machines appears to be harder than on identical or uniform machines. In the non-preemptive case, the three problems are NP-hard in the strong sense. $P||C_{\max}$ and $Q||C_{\max}$ have polynomial-time approximation schemes [8] [9]; for $R||C_{\max}$ we can find a schedule of length less than twice the optimum in polynomial time, while finding a schedule of length less than 3/2 times the optimum is NP-hard [13] [16]. The preemptive cases are easy. $P|pmtn|C_{\max}$, $Q|pmtn|C_{\max}$ and the two-machine problem $R2|pmtn|C_{\max}$ can be solved by combinatorial algorithms in time $O(n)$ [15], $O(n + m\log m)$ [6] and $O(n)$ [4], respectively, but for the general problem $R|pmtn|C_{\max}$ we have to resort to a combination of linear programming and open shop scheduling [12]. $P|pmtn|C_{\max}$ and $Q|pmtn|C_{\max}$ can even be solved in polylogarithmic space, or in polylogarithmic time on a polynomial number of processors [1] [14], but linear programming is P-complete under logspace transformations [2] [18] and hence unlikely to admit a polylogarithmic solution; see [11] for an introduction into these concepts. An open question is whether $R|pmtn|C_{\max}$ is P-complete as well; this would imply that, also in this sense, the problem is in a different complexity class than $P|pmtn|C_{\max}$ and $Q|pmtn|C_{\max}$.

In the present paper, we introduce a model with limited preemptions: the portion of a job that is allocated to one machine must be processed on that machine without interruption. This is a natural assumption in situations where such interruptions cause unreasonable reset times or costs. We will call the interruption of a job on one machine a *single-machine preemption* or a *split*; the latter term was introduced in [17]. Also in this model, $P|pmtn|C_{\max}$ and $Q|pmtn|C_{\max}$ appear to be easier than $R|pmtn|C_{\max}$. The combinatorial algorithms for identical and uniform machines and for two unrelated machines cited above do not introduce splits, but the LP-based approach for $R|pmtn|C_{\max}$ may very well do so. We will show that imposing the no-split condition makes the three-machine problem $R3|pmtn|C_{\max}$ NP-hard in the ordinary sense and the general problem $R|pmnt|C_{\max}$ NP-hard in the strong sense. These results even hold in the *restricted assignment* model, where for each job j a processing requirement p_j and a set of machines is given such that the time that job j takes on any machine in that set is equal to p_j and the time that job j takes on any other machine is prohibitively large.

We dedicate this paper to the memory of Gerhard Woeginger. The strong NP-hardness proof that we will present below is inspired by the reduction of 3-partition to the open shop problem $O||C_{\max}$ that he presented in [20].

* Corresponding author.
E-mail addresses: jkl@cwi.nl (J.K. Lenstra), nodari@uaem.mx (N. Vakhania).

https://doi.org/10.1016/j.orl.2023.02.004

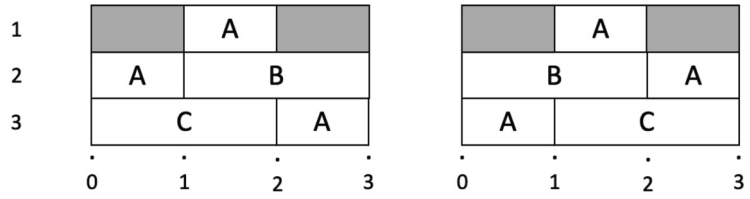

Fig. 1. Two schedules of length 3.

1. NP-hardness for three machines

We will show that minimizing the makespan for n jobs on three unrelated parallel machines without single-machine preemptions is NP-hard in the ordinary sense, even in the case of the restricted assignment model. In an earlier paper [19], NP-hardness was established for the more general case that the jobs become available at individual release dates.

We present a reduction from the partition problem, which is NP-hard [10]: given k items j of size a_j such that $\sum_{j=1}^{k} a_j = 2b$, does there exist a set $S \subset \{1, ..., k\}$ such that $\sum_{j\in S} a_j = b$?

Given an instance of partition, we define an instance of the 3-machine problem as follows. There are k partition jobs j, which require time a_j/b on machine 1 ($j = 1, \ldots, k$). Note that these jobs together occupy two time units on machine 1. In addition, there are three auxiliary jobs: job A requires time 3 on each machine, job B requires time 2 on machine 2, and job C requires time 2 on machine 3. All unspecified processing times are very large. We claim that there exists a schedule of length 3 if and only if the instance of partition is a yes-instance.

Suppose there exists a set S such that $\sum_{j\in S} a_j = b$. We schedule job A for one time unit on each of the machines 2, 1, 3, in that order, and assign job B to the interval $[1, 3)$ on machine 2 and job C to the interval $[0, 2)$ on machine 3. Finally, we schedule the partition jobs $j \in S$ and $j \notin S$ in the intervals $[0, 1)$ and $[2, 3)$, respectively, on machine 1. This yields a schedule of length 3 without splits. Note that we could also have reversed the machine ordering of job A. We refer to Fig. 1.

Conversely, suppose there exists a schedule of length 3. In such a schedule, each machine is fully occupied. Hence, job A is processed for one time unit on each machine. On machine 2, it either precedes of follows job B. In the former case, it must follow job C on machine 3 and is therefore processed in the interval $[1, 2)$ on machine 1. Hence, the partition jobs are processed in the intervals $[0, 1)$ and $[2, 3)$ on machine 1 without splits. In case job A follows job B on machine 2, it must precede job C on machine 3 and is still processed in the interval $[1, 2)$ on machine 1. In either case, the schedule on machine 1 certifies that we have a yes-instance of partition.

2. Strong NP-hardness for m machines

We will now show that minimizing the makespan for n jobs on m unrelated parallel machines without single-machine preemptions, where m is specified as part of the input to the problem, is NP-hard in the strong sense, even in the case of the restricted assignment model.

We present a reduction from the 3-partition problem, which is known to be NP-hard in the strong sense [3]: given $3k$ items j of size a_j such that $\sum_{j=1}^{3k} a_j = kb$ and $b/4 < a_j < b/2$ for $j = 1, \ldots, 3b$, does there exist a partition of the index set $\{1, ..., 3k\}$ into k 3-item subsets such that the sizes of the items in each subset sum up to exactly b?

Given an instance of 3-partition, we define an instance of the scheduling problem with $2k - 1$ machines. For convenience, we denote the machines by $M_0, M_1, \ldots, M_{k-1}, N_1, \ldots, N_{k-1}$. When specifying jobs and processing times, we assume, as before, that all unspecified processing times are prohibitively large.

There are $3k$ partition jobs j, which require time a_j/b on machine M_0 ($j = 1, \ldots, 3k$). Note that these jobs together occupy k time units on M_0. Our purpose is to construct a schedule of length $2k - 1$ in which the partition jobs are forced into the k unit-time intervals $[0, 1), [2, 3), \ldots, [2k - 2, 2k - 1)$ on M_0. Hence, we need to define auxiliary jobs that, in any schedule of length $2k - 1$, must occupy the remaining $k - 1$ unit-time intervals $[1, 2), [3, 4), \ldots, [2k - 3, 2k - 2)$ on M_0.

Our construction consists of a number of phases. For the first phase, focus on the first and last of the intervals that need to be blocked, $[1, 2)$ and $[2k - 3, 2k - 2)$. We define six jobs A_1, A_{k-1}, B_1, B_{k-1}, C_1, C_{k-1}, and four machines M_1, M_{k-1}, N_1, N_{k-1}.

- Job A_1 requires time $2k - 1$ on M_0, M_1 and N_1.
- Job B_1 requires time $2k - 2$ on M_1.
- Job C_1 requires time 2 on N_1.
- Job A_{k-1} requires time $2k - 1$ on M_0, M_{k-1} and N_{k-1}.
- Job B_{k-1} requires time $2k - 2$ on M_{k-1}.
- Job C_{k-1} requires time 2 on N_{k-1}.

We will not define other jobs that can be processed on M_1, M_{k-1}, N_1, N_{k-1}.

Consider a schedule of length $2k - 1$ and assume that all machines are fully occupied. It follows that job A_1 is processed for one time unit on M_0 and M_1 and for $2k - 3$ time units on N_1. Similarly, job A_{k-1} is processed for one time unit on M_0 and M_{k-1} and for $2k - 3$ time units on N_{k-1}. On M_1, job A_1 either precedes or follows job B_1.

Consider the case that, on M_1, job A_1 precedes job B_1 and hence occupies the interval $[0, 1)$. On N_1, it must follow job C_1 and hence occupy the interval $[2, 2k - 1)$. It therefore occupies the interval $[1, 2)$ on M_0. Now suppose that, on M_{k-1}, job A_{k-1} precedes job B_{k-1}, occupying the interval $[0, 1)$. On N_{k-1}, it must then follow job C_{k-1}, occupying the interval $[2, 2k - 1)$. It must therefore also occupy the interval $[1, 2)$ on M_0, which is infeasible. We conclude that job A_{k-1} follows job B_{k-1} on M_{k-1}, that it precedes job C_{k-1} on N_{k-1}, and that it occupies the interval $[2k - 3, 2k - 2)$ on M_0. We refer to Fig. 2.

In the reverse case, job A_1 follows job B_1 on M_1, job A_{k-1} precedes job B_{k-1} on M_{k-1}, and these jobs still occupy the intervals $[1, 2)$ and $[2k - 3, 2k - 2)$ on M_0.

To block the other intervals on M_0, we define more phases. In each phase, the processing times of the B-jobs decrease by 2 and those of the C-jobs increase by 2. In phase i, we need to block the intervals $[2i - 1, 2i)$ and $[2k - 2i - 1, 2k - 2i)$. We define six jobs A_i, A_{k-i}, B_i, B_{k-i}, C_i, C_{k-i} and four machines M_i, M_{k-i}, N_i, N_{k-i}.

- Job A_i requires time $2k - 1$ on M_0, M_i and N_i.
- Job B_i requires time $2k - 2i$ on M_i.
- Job C_i requires time $2i$ on N_i.
- Job A_{k-i} requires time $2k - 1$ on M_0, M_{k-i} and N_{k-i}.
- Job B_{k-i} requires time $2k - 2i$ on M_{k-i}.
- Job C_{k-i} requires time $2i$ on N_{k-i}.

If k is odd, the construction consists of $(k - 1)/2$ such phases. If k is even, the construction consist of $(k - 2)/2$ such phases and a final phase $k/2$ in which we need to block the middle unit-time interval $[k - 1, k)$ on M_0. For this, we need three jobs $A_{k/2}$, $B_{k/2}$,

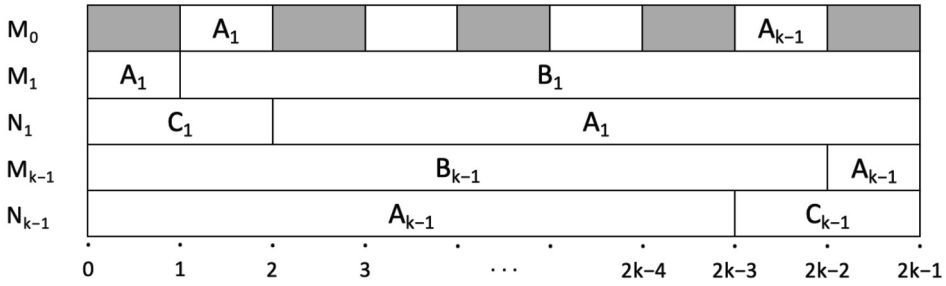

Fig. 2. A schedule of length $2k-1$.

$C_{k/2}$ and two machines $M_{k/2}$ and $N_{k/2}$, where
- job $A_{k/2}$ requires time $2k-1$ on M_0, $M_{k/2}$ and $N_{k/2}$,
- job $B_{k/2}$ requires time k on $M_{k/2}$, and
- job $C_{k/2}$ requires time k on $N_{k/2}$,

which will remind the reader of the NP-hardness proof for three machines.

This completes the construction. It should be clear that, if the instance of 3-partition has a yes-answer, we can construct a schedule of length $2k-1$ without splits. Conversely, if a schedule of length $2k-1$ exists, it is indeed the case that all machines are fully occupied, and the schedule on M_0 certifies that we have a yes-instance of 3-partition.

3. Open problems

It remains an open question if the problem on a *fixed* number of unrelated parallel machines without single-machine preemptions is solvable in pseudopolynomial time or NP-hard in the strong sense. There is a striking parallel to the complexity status of non-preemptive open shop scheduling: $O2||C_{\max}$ is solvable in $O(n)$ time, $O3||C_{\max}$ is NP-hard in the ordinary sense, $O||C_{\max}$ is NP-hard in the strong sense, and the precise complexity of $Om||C_{\max}$ remains open; see [20] and in particular Open problem 11.1.

Finding an optimal no-split schedule is an interesting algorithmic challenge. The two-phase approach for the case that splits are allowed falls short in several ways. We briefly recall the algorithm [12]. One first solves a linear program to determine the schedule length C^*:

$$C^* = \min\{C \mid \begin{array}{ll} \sum_{i=1}^{m} x_{ij}/p_{ij} = 1 & (j=1,\ldots,n),\\ \sum_{i=1}^{m} x_{ij} \le C & (j=1,\ldots,n),\\ \sum_{j=1}^{n} x_{ij} \le C & (i=1,\ldots,m),\\ x_{ij} \ge 0 & (i=1,\ldots,m,\ j=1,\ldots,n)\};\end{array}$$

here, x_{ij} represents the total time that is to be spent by job j on machine i. Next, given the distribution of the jobs over the machines, one solves a preemptive open shop problem to construct a feasible schedule of length C^* [5]; this will generally introduce splits.

When splits are not allowed, the open shop in phase 2 is non-preemptive. The reductions given in this paper show that it is NP-hard to determine if there exists a schedule without splits of length C^*. Moreover, the linear program in phase 1 may give a suboptimal distribution of the jobs over the machines. Consider, for example, an instance of the partition problem with $k=1$ and $a_1=2$, and apply our first reduction to obtain an instance of $R3|pmtn|C_{\max}$. The linear program has an optimal solution $C^*=3$ with $x_{11}=x_{2B}=x_{3C}=2$, $x_{1A}=x_{2A}=x_{3A}=1$, the shortest non-preemptive schedule respecting that solution has a length 4, but there exist schedules without splits of length 3.5.

References

[1] E. Dekel, S. Sahni, Parallel scheduling algorithms, Oper. Res. 31 (1983) 24–49.
[2] D. Dobkin, R.J. Lipton, S. Reiss, Linear programming is log-space hard for P, Inf. Process. Lett. 8 (1979) 96–97.
[3] M.R. Garey, D.S. Johnson, Strong NP-completeness results: motivation, examples, and implications, J. ACM 25 (1978) 499–508.
[4] T. Gonzalez, E.L. Lawler, S. Sahni, Optimal preemptive scheduling of two unrelated processors, ORSA J. Comput. 2 (1990) 219–224.
[5] T. Gonzalez, S. Sahni, Open shop scheduling to minimize finish time, J. ACM 23 (1976) 665–679.
[6] T. Gonzalez, S. Sahni, Preemptive scheduling of uniform processor systems, J. ACM 25 (1978) 92–101.
[7] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, Ann. Discrete Math. 5 (1979) 287–326.
[8] D.S. Hochbaum, D.B. Shmoys, Using dual approximation algorithms for scheduling problems: theoretical and practical results, J. ACM 34 (1987) 144–162.
[9] D.S. Hochbaum, D.B. Shmoys, A polynomial approximation scheme for machine scheduling on uniform processors: using the dual approximation approach, SIAM J. Comput. 17 (1988) 539–551.
[10] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum Press, New York, 1972, pp. 85–103.
[11] G.A.P. Kindervater, J.K. Lenstra, Parallel computing in combinatorial optimization, Ann. Oper. Res. 14 (1988) 245–289.
[12] E.L. Lawler, J. Labetoulle, On preemptive scheduling of unrelated parallel processors by linear programming, J. ACM 25 (1978) 612–619.
[13] J.K. Lenstra, D.B. Shmoys, E. Tardos, Approximation algorithms for scheduling unrelated parallel machines, Math. Program. 46 (1990) 259–271.
[14] C.U. Martel, A parallel algorithm for preemptive scheduling of uniform machines, J. Parallel Distrib. Comput. 5 (1988) 700–715.
[15] R. McNaughton, Scheduling with deadlines and loss functions, Manag. Sci. 6 (1959) 1–12.
[16] E. Shchepin, N. Vakhania, An optimal rounding gives a better approximation for scheduling unrelated machines, Oper. Res. Lett. 33 (2005) 127–133.
[17] E. Shchepin, N. Vakhania, On the geometry, preemptions and complexity of multiprocessor and open shop scheduling, Ann. Oper. Res. 159 (2008) 183–213.
[18] L.G. Valiant, Reducibility by algebraic projections, in: Logic and Algorithmic; Monographie No. 30 de l'Enseignement de Mathématique, Geneva, 1982, pp. 365–380.
[19] N. Vakhania, On preemptive scheduling on unrelated machines using linear programming, AIMS Math. 8 (3) (2023) 7061–7082.
[20] G.J. Woeginger, Multi-Operation Models; chapter 11: Open Shops. elementsofscheduling.nl, 2021.