# CS 467/567, Assignment 1

Wentao Lu, Yi Ren, Baichuan Lian

## 1 Objective

To prove NP-completeness for the two-machine scheduling problem by formulating a reduction from the knapsack problem.

## 2 Definition

- TWO-MACHINE SCHEDULING: Given the execution time $a_1, a_2 \ldots, a_n$ of $n$ independent tasks and a deadline $D$, is it possible to schedule the $n$ tasks on two identical machines in a non-preemptive fashion (meaning that once a task starts executing it must run to completion) such that all the task complete by the deadline $D$.
- KNAPSACK: Given $n$ objects with volumes $v_i$ and prices $s_i$, $1 \leq i \leq n$, a knapsack of volume $V$, and a constant $S > 0$, is there a subset $N' \subseteq \{1, \ldots, n\}$ such that $\Sigma_{i \in N'} v_i < V$ (the objects in $N'$ can be placed in the knapsack) and $\Sigma_{i \in N'} s_i \geq S$ (their price exceeds the given limit $S$)?

## 3 Proof of NP-completeness

Our complete proof consists of two steps.

### 3.1 TWO-MACHINE SCHEDULING IS NP

Given any candidate solution (purported certificate) of this problem which divides the $n$ independent tasks into two bins, we can easily compute the total execution time of tasks in each bin, and then check if both of them $\leq D$. Since the solution can be verified in linear time, by definition[1, p. 1049] it is an NP problem.

### 3.2 TWO-MACHINE SCHEDULING IS NP-hard

After Stephen Cook proved 3-SAT to be the first known NP-complete problem in 1971[2], the next year Richard Karp further proved that another 21 common computational problems are all NP-complete[3], one of which is the knapsack problem. To prove NP-hardness, we will show that the knapsack problem is reducible to the two-machine scheduling problem.[1]

[1]We have found a lot of information about the makespan approximation algorithms, but nothing about the proof of this exact problem. As a workaround, we have referenced the reduction to the partition problem in Karp's paper[3] and adapted it here since these two problems are very similar with each other.

For every instance of the knapsack problem, construct $n+2$ independent tasks with execution time $\{a_1, \ldots, a_{n+2}\}$ and a deadline $D = \Sigma_{i=1}^{n} a_i + 2$. Let

$$\begin{aligned} a_i &= v_i + s_i\,,\ 1 \le i \le n \\ a_{n+1} &= \Sigma_{i\in P} v_i + \Sigma_{i\in P} s_i + 2 \\ a_{n+2} &= \Sigma_{i=1}^{n} a_i - \Sigma_{i\in P} v_i - \Sigma_{i\in P} s_i + 2 \end{aligned}$$

where $P$ denotes the set of items placed in the knapsack. It is evident that this mapping takes polynomial time in the size of input. Now let's prove that this translation is indeed a reduction.

$\Rightarrow$ First, suppose that the knapsack problem has a true instance, which means there is indeed a subset of items $P \subseteq \{1, \ldots, n\}$ such that

$$\begin{aligned} \Sigma_{i\in P} v_i &< V \\ \Sigma_{i\in P} s_i &\ge S \end{aligned}$$

Let

$$\begin{aligned} M_1 &= \{a_i \mid i \in P\} \cup \{a_{n+2}\} \\ M_2 &= \{a_i \mid i \notin P\} \cup \{a_{n+1}\} \end{aligned}$$

Then we have

$$\text{the sum of } M_1 = \Sigma_{i\in P} v_i + \Sigma_{i\in P} s_i + \Sigma_{i=1}^{n} a_i - \Sigma_{i\in P} v_i - \Sigma_{i\in P} s_i + 2 = \Sigma_{i=1}^{n} a_i + 2 \tag{1}$$

$$\text{the sum of } M_2 = \Sigma_{i=1}^{n} a_i - \Sigma_{i\in P} v_i - \Sigma_{i\in P} s_i + \Sigma_{i\in P} v_i + \Sigma_{i\in P} s_i + 2 = \Sigma_{i=1}^{n} a_i + 2 \tag{2}$$

From (1) and (2), it's clear that we have $M_1 = M_2 = D$, also notice that $M_1 \cup M_2$ is the set of all tasks, therefore we have found a feasible schedule for two identical machines.

$\Leftarrow$ Conversely, suppose that the translated two-machine scheduling problem has a true instance, so the set of $n+2$ independent tasks can be divided into two subsets, which can be scheduled on two machines $M_1$ and $M_2$, respectively, by the deadline $D = \Sigma_{i=1}^{n} a_i + 2$.

Since execution time must be nonnegative, and notice that $a_{n+1} + a_{n+2} = \Sigma_{i=1}^{n} a_i + 4 > D$, this means $a_{n+1}$ and $a_{n+2}$ must not be executed on the same machine. Without loss of generality, assume that $a_{n+2}$ is scheduled on $M_2$, then the execution time of the rest of the tasks scheduled on $M_2$ must sum up to

$$D - a_{n+2} = \Sigma_{i=1}^{n} a_i + 2 - a_{n+2} = \Sigma_{i\in P} v_i + \Sigma_{i\in P} s_i$$

As per our reduction assumption, it imples that this subset $P \subseteq \{1, \ldots, n\}$ has the property

$$\begin{aligned} \Sigma_{i\in P} v_i &< V \\ \Sigma_{i\in P} s_i &\ge S \end{aligned}$$

Thus, it makes a true instance of the knapsack problem as well.

Now we have KNAPSACK $\le_p$ TWO-MACHINE SCHEDULING, which implies that the two-machine scheduling problem is at least as hard as the knapsack. As a result, according to *Lemma 34.8* from the textbook[1], this completes our proof that two-machine scheduling is also NP-complete.

## 4 Optimization Variant

The optimization variant of our two-machine scheduling problem is the so-called *makespan* problem, in our case with two machines denoted by $P2||C_{max}$[4], which can be described as:

Given the execution time $a_1, a_2 \ldots, a_n$ of $n$ independent tasks, find a way to schedule the $n$ tasks on two identical parallel machines in a non-preemptive fashion, so as to minimize the maximum task completion time. In other words, we would like to do our best to equally assign the $n$ tasks to both machines, such that the total execution time on the machine that finishes last is minimized. This optimization problem $P2||C_{max}$ is NP-hard.[3]

## References


[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009, ISBN: 978-0-262-03384-8. [Online]. Available: http://mitpress.mit.edu/books/introduction-algorithms.

[2] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, 1971, pp. 151–158. DOI: 10.1145/800157.805047. [Online]. Available: https://doi.org/10.1145/800157.805047.

[3] R. M. Karp, "Reducibility among combinatorial problems," in *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, 1972, pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9. [Online]. Available: https://doi.org/10.1007/978-1-4684-2001-2_9.

[4] P. Schuurman and G. J. Woeginger, "Approximation schemes - a tutorial," in *Lectures on Scheduling*, 2000.