We now show how to prove NP-hardness by illustrating it on 3 NP-hard scheduling problems.

1 \((P2||C_{max})\)

We show this by reducing Partition to \((P2||C_{max})\). Recall in Partition we have \(n\) numbers \(\{a_1, \ldots, a_n\}\) such that \(\sum a_i = 2B\). The goal is to decide if there is a subset \(S\) such that \(\sum_{x \in S} x = B\)?

Given an instance of partition, construct an instance of \((P2||C_{max})\) with \(n\) jobs, and job \(j\) having processing time \(p_j = a_j\).

**Claim 1.1.** The makespan of the above instance is \(B\) iff there exists a partition of the numbers.

**Proof.** If there is a partition \(S\) which adds up to \(B\), then the remaining numbers also add up to \(B\). Thus, scheduling jobs corresponding to the numbers in \(S\) on machine 1 and the remaining jobs in machine 2 gives a makespan of \(B\).

If the makespan is indeed \(B\), then since the sum of processing times is \(2B\), both the machines must finish at time \(B\). Thus, the set of jobs on any machine forms a partition. \(\square\)

2 \((1|\text{prec}|\sum U_j)\)

We show that \((1|\text{prec}|\sum U_j)\) is NP-hard by reducing Clique to it. Recall that Clique problem was given a graph \(G\) and a value \(c\), if there was a clique of size exactly \(c\) in \(G\). A clique in a graph is a set of vertices such that there is an edge present between each pair of vertices.

Let \(G = (V, E)\) be a graph with \(k\) vertices and \(m\) edges and let \(c\) be the size of clique we are looking for. A clique has \(\ell = \frac{1}{2}c(c - 1)\) edges.

We build an instance of \((1|\text{prec}|\sum U_j)\) with \(n = k + m\) jobs. There are two kinds of jobs, a vertex job corresponding to each vertex and an edge job corresponding to each edge. The processing time of each job \(j\) is \(p_j = 1\). The due dates of the jobs \(j\) corresponding to vertices are \(d_j = k + m\) and the due dates of jobs \(i\) corresponding to edges are \(d_i = c + \ell\). For every triple of jobs \(u, v\) vertex jobs and \((u, v)\) edge job, there is a precedence constraint \(u \rightarrow (u, v)\) and \(v \rightarrow (u, v)\).

**Lemma 2.1.** There exists a schedule with exactly \(m - \ell\) late jobs iff if \(G\) has a clique of size \(c\).

**Proof.** First note that in any idle-time free schedule, the jobs corresponding to vertices will always be on time.
Suppose $G$ has a clique of size $c$. Let $S$ be the set of $c$ vertices in the clique. Consider the schedule where the corresponding $c$ jobs are scheduled first (in any order) followed by the $\ell$ jobs corresponding to the edges in this clique (in any order). The time for these jobs to complete is $c + \ell$, so all the jobs in this set meet their deadline. The only jobs that cannot meet their deadline are the jobs corresponding to edges not in the clique. There are $m - \ell$ such jobs.

To prove the reverse, we consider the contrapositive. (This is slightly different from what we did in class). Suppose $G$ has no clique of size $c$. Consider any set of $\ell$ edge-jobs. Let $T$ be the set of node-jobs corresponding to the endpoints of the edge-jobs. Then $|T| > c$, because these edges cannot form a clique. In order to schedule these $\ell$ jobs, the jobs in $T$ must be scheduled first. Then at least one of the $\ell$ jobs will be late. The number of late jobs is therefore at least $m - \ell + 1$.

3 \((F3||C_{\text{max}})\)

In class, we showed that \((F3||C_{\text{max}})\) was weakly NP-hard by reducing it from \textsc{Partition}. We show that \(F3||C_{\text{max}}\) is strongly NP-hard by reducing 3-partition to \((F3||C_{\text{max}})\).

Recall the 3-partition problem: Given positive integers $a_1, \ldots, a_{3t}$, such that $\sum_{j=1}^{3t} a_j = tB$ and $\frac{B}{4} < a_j < \frac{B}{2}$, and is there a $t$-partition of $\{a_1, \ldots, a_{3t}\}$ into three-element subsets $S_1, S_2, \ldots, S_t$ such that $\sum_{x \in S_i} x = B$ for all $i$?

Note that if we show that there is a set $S$ which contains elements summing to $B$, then it must contain 3 elements. This follows from the assumption on the $a_j$'s. Therefore it is enough to just partition without bothering about the number of elements in the set.

We build an instance of \(F3||C_{\text{max}}\). We have two kinds of jobs. There are $3t$ jobs $\alpha_1, \alpha_2, \ldots, \alpha_{3t}$ corresponding to the $3t$ numbers. There are $t+1$ auxiliary jobs $\beta_0, \beta_1, \ldots, \beta_{t-1}, \beta_t$. The processing times are as follows.

<table>
<thead>
<tr>
<th>Jobs</th>
<th>$\beta_0$</th>
<th>$\beta_1$</th>
<th>...</th>
<th>$\beta_{t-1}$</th>
<th>$\beta_t$</th>
<th>$\alpha_1$</th>
<th>...</th>
<th>$\alpha_{3t}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine 3</td>
<td>2B</td>
<td>2B</td>
<td>...</td>
<td>2B</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
<tr>
<td>Machine 2</td>
<td>B</td>
<td>B</td>
<td>...</td>
<td>B</td>
<td>B</td>
<td>$a_1$</td>
<td>...</td>
<td>$a_{3t}$</td>
</tr>
<tr>
<td>Machine 1</td>
<td>0</td>
<td>2B</td>
<td>...</td>
<td>2B</td>
<td>2B</td>
<td>0</td>
<td>...</td>
<td>0</td>
</tr>
</tbody>
</table>

**Lemma 3.1.** There is a schedule of makespan $(2t + 1)B$ iff there is a 3-partition.

**Proof.** Suppose we only want to schedule jobs $\beta_0, \beta_1, \ldots, \beta_t$. If we schedule them in the order $\beta_0, \beta_1, \ldots, \beta_t$, there is no idle time on machine 1, there are gaps of size $B$ after every job on machine 2 and there is no idle time on machine 3 after time $B$. The makespan of this schedule is given by the completion time of machine 3: $B + 2Bt = (2t + 1)B$.

Note that a makespan of value $< (2t + 1)B$ cannot be achieved for these $t + 1$ jobs: Each of these jobs will incur a delay time of $B$ at the start on machine 3 and the sum of processing times on machine 3 is $2tB$.

The question is: can we schedule the remaining $3t$ jobs without increasing the makespan? Since the processing time of these jobs on machine 1 is 0, we can schedule these jobs right at the start of the schedule without affecting the makespan. Likewise, we can schedule these $3t$ jobs at the end of the schedule on the third machine without affecting the makespan. As a result, we can schedule these jobs whenever we want on machine 2.
On machine 2, there are $t$ gaps of idle time of length $B$ each. So, if there is a way to split these jobs into these gaps, then there is a $t$-partition such that each set of jobs has processing time $\leq b$. Since the sum of these $3t$ processing times is $tB$, each set must have total processing time $B$. Therefore, there is a solution to the 3-partition problem.

If there is a solution to the 3-partition problem, then there is a $t$-partition of the numbers into 3-element subsets such that each set has sum equal $B$. These 3-element subsets will fit into the gaps in our schedule. $\square$