Exercises

Exercise 1 Consider the following instance of the scheduling problem $1||\sum_j w_jC_j$. Give an optimal schedule and its value.

jobs	1	2	3	4
w_j	6	11	9	5
p_j	3	5	7	4

Solution: $\frac{p_2}{w_2} = \frac{5}{11} \leq \frac{p_1}{w_1} = \frac{3}{6} \leq \frac{p_3}{w_3} = \frac{7}{9} \leq \frac{p_4}{w_4} = \frac{4}{5}$. The optimal order is is $2, 1, 3, 4$ which gives completion times $C_2 = 3, C_1 = 8, C_3 = 15$ and $C_4 = 19$. The value is $w_1C_1 + w_2C_2 + w_3C_3 + w_4C_4 = 6 \cdot 8 + 11 \cdot 5 + 9 \cdot 15 + 5 \cdot 19 = 333$.

Exercise 2 Consider the following instance of the scheduling problem $1||L_{\max}$. Give an optimal schedule and its value.

jobs	1	2	3	4
p_j	5	4	3	6
d_j	3	5	11	12

Solution: Place the jobs in Earliest Due Date (EDD) order. Since $d_1 < d_2 < d_3 < d_4$, the optimal order is $1, 2, 3, 4$. $L_{\max} = max\{C_1 - d_1, C_2 - d_2, C_3 - d_3, C_4 - d_4\} = \max\{5 - 3, 9 - 5, 12 - 11, 18 - 12\} = 6$.
(EDD is always optimal but other optimal schedules may be possible. Here, $2, 1, 3, 4$ is optimal as well.)

Exercise 3 De decision problems PARTITION and 3-PARTITION are both NP-complete and are defined as follows:

PARTITION: An instance is given by positive numbers A and $a_1, a_2, \ldots, a_n$ with $\sum_i a_i = 2A$. Question: Is there an $S \subset \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} a_i = A$.?

3-PARTITION: An instance is given by positive numbers B and $b_1, b_2, \ldots, b_{3m}$ with $\sum_i b_i = mB$. Question: Is there a partition of $\{1, 2, \ldots, 3m\}$ into $S_1, S_2, \ldots, S_m$ such that $\sum_{j \in S_i} b_j = B$ for all $i = 1, \ldots, m$?

(a) Show that the PARTITION problem can be reduced to the scheduling problem $P2||C_{\max}$.
(b) Show that the 3-PARTITION problem can be reduced to the scheduling problem $P||C_{\max}$.

Solution: (a) The two problems are almost identical. Given an instance of PARTITION (with the notation as above) define an instance of $P2||C_{\max}$ as follows. Take n jobs with processing time $p_j = a_j, j = 1 \dots n$ and let the number of machines be $m = 2$.

There is an S with $\sum_{i \in S} a_i = A$. $\Leftrightarrow$ There is a schedule of length $\leq A$.

Hence, if we can solve the scheduling problem efficiently, then we can solve the Partition problem efficiently.

(b) Give an instance of 3-PARTITION (with the notation as above) define an instance of $P||C_{\max}$ as follows. Take $3m$ jobs with processing time $p_j = b_j, j = 1 \dots 3m$. Let m be the number of machines.

There exists a 3-Partition. $\Leftrightarrow$ There is a schedule of length $\leq B$.

Hence, if we can solve the scheduling problem efficiently, then we can solve the 3-Partition problem efficiently.

Exercise 4 Consider the scheduling problem $1|r_j|\sum_j C_j$ and the following algorithm (SPT):
When the machine is not processing any job, then start the job that has the smallest processing time p_j among the available jobs. (We say that a job is available if it has been released but not started yet).
Show by an example that this algorithm does not always lead to an optimal schedule.

Solution: Many answers are possible. Take for example a long job that is followed directly by a small job: $p_1 = 10, r_1 = 0$ and $p_2 = 1, r_2 = 1$. The algorithm does job 1 first. This gives $C_1 = 10$ and $C_2 = 11$. The value is $10+11 = 21$. However, it is optimal to do job 2 first. This gives $C_2 = 2, C_1 = 12$ with value $2+12 = 14$.

Exercise 5 Consider the scheduling problem $P|r_j, pmtn|C_{\max}$. Give a polynomial time algorithm which solves the problem by formulating it as a linear program (LP). Assume for simplicity that $0 = r_1 \leq r_2 \leq \cdots \leq r_n$ where n is the number of jobs.

Hint: Use a variable Z for the length of the schedule. The objective then becomes: minimize Z. Take as variables x_{tj} ($t = 1, 2 \ldots, n$) which denote the amount of time spent on job j between time r_t and r_{t+1} ($t \leq n-1$) and between r_n and Z ($t = n$). Explain how an optimal LP-solution can be translated into a feasible schedule.

Solution: Each job needs to be processed completely. This gives the following constraint:

$$\sum_{t=1}^{n} x_{tj} = p_j \quad \text{for all jobs } j.$$

The amount of time spent in interval $[r_t, r_{t+1}]$ is no more than the length of the interval:

$$\begin{aligned} x_{tj} &\leq r_{t+1} - r_t, && \text{for all jobs } j \text{ and intervals } t \leqslant n-1 \\ x_{nj} &\leq Z - r_n, && \text{for all jobs } j. \end{aligned}$$

Another constraint is that the total processing time in the interval $[r_t, r_{t+1}]$ is no more than the number of machines time the length of the interval:

$$\begin{aligned} \sum_{j=1}^{n} x_{tj} &\leq m(r_{t+1} - r_t) && \text{for all intervals } t \leqslant n-1 \\ \sum_{j=1}^{n} x_{nj} &\leq m(Z - r_n) \end{aligned}$$

Further, no job j can start before its release time r_j:

$$x_{tj} = 0, \text{ for all } t < j.$$

The complete LP becomes:

$$\min \ Z \tag{1}$$

$$s.t. \qquad \sum_{t=1}^{n} x_{tj} = p_j, \qquad \text{for all jobs } j \tag{2}$$

$$x_{tj} \leq r_{t+1} - r_t \qquad \text{for all } t \leqslant n-1, \text{ and all } j \tag{3}$$

$$x_{nj} \leq Z - r_n \qquad \text{for all } j \tag{4}$$

$$\sum_{j=1}^{n} x_{tj} \leq m(r_{t+1} - r_t) \qquad \text{voor all } t \leqslant n-1 \tag{5}$$

$$\sum_{j=1}^{n} x_{nj} \leq m(Z - r_n) \tag{6}$$

$$x_{tj} = 0 \qquad \text{for all } t < j \tag{7}$$

$$x_{tj} \geqslant 0 \qquad \text{for all } t, j \tag{8}$$

Note that an optimal solution x^*_{tj}, Z^* is not yet a feasible schedule since jobs are not assigned to machines. A schedule can be obtained using McNaughton's wrap-around-rule. That rule was used for the problem $P|pmtn|C_{max}$. Note that we have such a scheduling problem for each interval. For interval t, the processing times are x^*_{tj} for $j = 1, \ldots, n$. McNaughton's wrap-around-rule gives a schedule of length

$$\max\{\max_j\{x^*_{tj}\} \ , \ \frac{1}{m}\sum_{j=1}^{n} x^*_{tj}\}.$$

Constraints (3)+(5) ((4)+(6) for the last interval) ensure that the length of the wrap-around-schedule for the interval is no more than the length of the interval, $r_{t+1} - r_t$.

In short: The algorithm first solves the LP and then a feasible schedule is found using the wrap-around-rule for each of the n intervals.

Exercises from the slides.

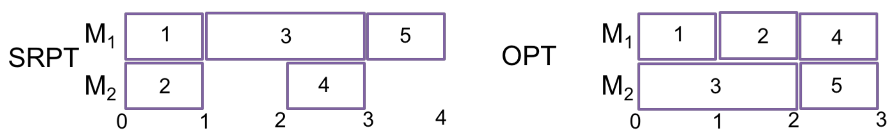
Exercise 1 (Slides) Show (by an example) that SRPT is not optimal on parallel machines.

SPRT on m parallel machine:
At any moment in time, process the m jobs with smallest remaining processing time (or all jobs if there are less than m jobs available at that time.

Solution: The following instance works:

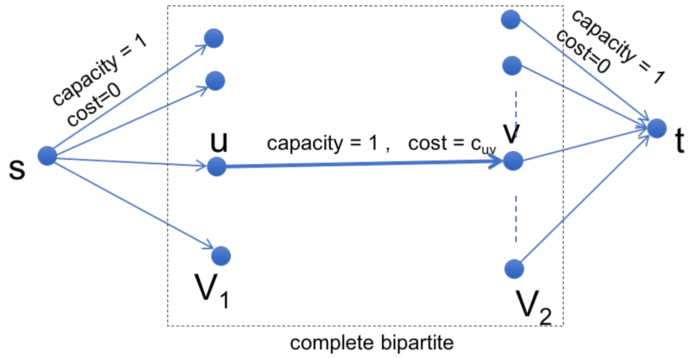
jobs	1	2	3	4	5
r_j	0	0	0	2	2
p_j	1	1	2	1	1

The total completion time is 12 for SRPT and 11 for the optimal schedule.



Exercise 2 (Slides) This exercise refers to problem $R||\sum C_j$ on the slides. Form this exercise, it follows that this scheduling problem can be solved efficiently. Let $G = (V_1 \cup V_2, E)$ be a complete bipartite graph with $|V_1| \leq |V_2|$. For any pair $u \in V_1$ and $v \in V_2$ let c_{uv} be the cost of edge (u, v). Say that a matching M is perfect if all vertices in V_1 are matched. Since the graph is complete and $|V_1| \leq |V_2|$, a perfect matching exists. In the MINCOST PERFECT MATCHING problem we need to find a perfect matching for which the total cost of the edges in the matching is minimized.
Show how the Mincost perfect matching problem can be reduced to a mincost flow problem.
Hint: Remember from week 1 how the maximum matching problem can be reduced to the maximum flow problem.

Solution: See the figure. To solve the mincost perfect matching problem we find a minimum cost s-t-flow of value $|V_1|$. This problem can be solved in polynomial time. Since all capacities are 1, the optimum flow has only flow values 0 or 1 on the edges. This corresponds to a matching.



Exercise 3 (Slides) (Difficult)
We have seen a 2-approximation for the problem $1|prec|\sum C_j$. Consider the following generalizations:

- $1|prec|\sum w_j C_j$
- $1|r_j, prec|\sum C_j$

(a) Does the same algorithm and proof apply for the weighted version $1|prec|\sum w_j C_j$?

(b) Try to apply the same technique to the problem $1|r_j, prec|\sum C_j$. What is the approximation ratio that you get?

Answer:(a) The algorithm and proof are exactly the same, except that we add the weights. The complete proof is given here. The changes are in red.

$1|prec|\sum w_j C_j$

For any set of jobs $S \subseteq \{1, 2, \ldots, n\}$ denote $p(S) = \sum_{j \in S} p_j$.

Lemma 1. *For any feasible schedule and for any set of jobs $S \subseteq \{1, 2, \ldots, n\}$:*

$$\sum_{j \in S} p_j C_j \geqslant \frac{1}{2} p(S)^2.$$

Proof. See slides. □

With the lemma above we see that the following LP is a relaxation of our scheduling problem. Here, there is a variable C_j for each jobs j.

$$\begin{array}{lll} \text{(LP)}\ \min & Z = \sum_{j=1}^{n} w_j C_j & \\ s.t. & C_j \geqslant 0 & \text{for all jobs } j \\ & C_k \geqslant C_j + p_k & \text{for all pairs } j \to k \\ & \sum_{j \in S} p_j C_j \geqslant \frac{1}{2} p(S)^2 & \text{for all sets } S \subseteq \{1, \dots, n\} \end{array}$$

Algorithm

1. Solve the LP. Let Z^*_{LP} be the optimal value and let C_j be the LP-values and relabel s.t. $C_1 \leqslant C_2 \leqslant \dots C_n$.
2. Place the jobs in the order $1, 2, \dots, n$. Let C'_j be the completion time of job j in this schedule.

Theorem 1. *The algorithm above is a 2-approximation algorithm for* $1|prec|\sum w_j C_j$

Proof. Consider an arbitrary job j. From the last constraint in the LP we see that

$$C_j \sum_{k \leqslant j} p_k = \sum_{k \leqslant j} C_j p_k \geqslant \sum_{k \leqslant j} C_k p_k \geq \frac{1}{2}(\sum_{k \leqslant j} p_k)^2 \quad \Rightarrow \quad C_j \geq \frac{1}{2} \sum_{k \leqslant j} p_k. \qquad (9)$$

Further, we have that in the final schedule $C'_j = \sum_{k \leqslant j} p_k$. Combining these, we get

$$C'_j = \sum_{k \leqslant j} p_k \leq 2C_j.$$

Now take the sum over all jobs:

$$\sum_j w_j C'_j \leqslant 2 \sum_j w_j C_j = 2Z^*_{LP} \leqslant 2\text{OPT}.$$

□

(b) $1|r_j, prec|\sum C_j$

Lemma 1 still holds. The LP is almost the same. We add one constraint. Note that non-negativity, $C_j \geqslant 0$, is now implied by the first constraint.

$$\begin{aligned}
\text{(LP)}\ \min \quad & Z = \sum_{j=1}^{n} C_j \\
s.t. \quad & C_j \geqslant r_j + p_j && \text{for all jobs } j \\
& C_k \geqslant C_j + p_k && \text{for all pairs } j \rightarrow k \\
& \sum_{j \in S} p_j C_j \geqslant \tfrac{1}{2} p(S)^2 && \text{for all sets } S \subseteq \{1, \ldots, n\}
\end{aligned}$$

Algorithm

1. Solve the LP. Let Z^*_{LP} be the optimal value and let C_j be the LP-values and relabel s.t. $C_1 \leqslant C_2 \leqslant \ldots C_n$.
2. Schedule the jobs non-preemptively and as early as possible in the order $1, 2, \ldots, n$. Denote the obtained schedule by σ and let C'_j be the completion time of job j in this schedule.

Theorem 2. *The algorithm above is a 3-approximation algorithm for* $1|r_j, prec|\sum C_j$

Proof. The first part of the proof is exactly the same as that of $1|r_j|\sum C_j$. Consider an arbitrary job j. Since jobs are scheduled in σ in the order $1, 2, ...$ we have that

(i) only jobs $k \leqslant j$ are scheduled before time C'_j in σ.

Further, since at time C_j all jobs $k \leqslant j$ have been released and jobs are scheduled as early as possible, we have that

(ii) there is no idle time in σ between time C_j and C'_j.

From (i) and (ii) we see that

$$C'_j \leqslant C_j + \sum_{k \leq j} p_k. \tag{10}$$

Next we use agian (9) and combining this with (10): $C'_j \leqslant C_j + 2C_j = 3C_j$. Now take the sum over all jobs:

$$\sum_j C'_j \leqslant 3 \sum_j C_j = 3Z^*_{LP} \leqslant 3\text{OPT}.$$

□