Assignment 3 solutions (Approx 2013)

(1) Scheduling on a single machine.

Given: set of jobs J.

For $j \in J$: p_j = processing time

w_j = weight

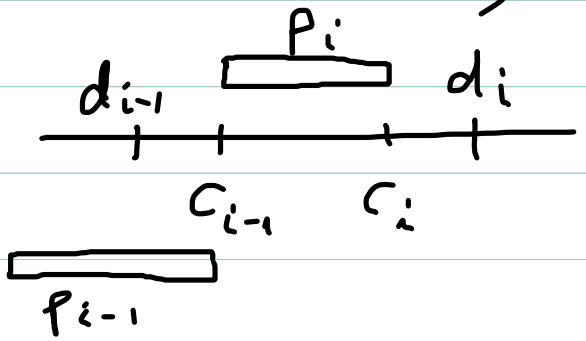
d_j = deadline

Output: $S \subseteq J$, $C_j \leq d_j$ $\forall j \in S$ where C_j: completion time

Measure: max $\sum_{j \in S} w_j$

a) Let $1 \leq j \leq n$ be indices of jobs in the order of the optimal schedule: $C_1 < C_2 < \dots < C_n$. Suppose that not all jobs that terminate before due date are scheduled first.

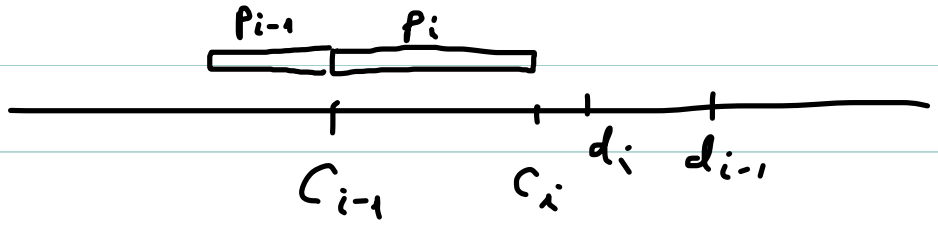
$\Rightarrow \exists\ 2 \leq i \leq n$, $C_{i-1} > d_{i-1}$ and $C_i \leq d_i$.



Swapping i with $i-1$ on the schedule will not decrease the cost of the solution since job i will complete earlier, so it continues to be completed on time.

$\Rightarrow$ There exists an optimal schedule where there is no pair of jobs $i-1$ & i as above (all jobs that finish on time are scheduled first).

Consider now that both jobs i and $i-1$ complete on time on the optimal solution, but $d_{i-1} > d_i$.



By swapping i & $i-1$ on the schedule, we obtain for the new completion times C'_i and C'_{i-1} the following:

$C'_i < C'_{i-1} = C_i \leq d_i \leq d_{i-1}$ and both i and $i-1$ continue to finish before their deadline, and the cost of the solution does not decrease.

b) Let $w_j = 1 \ \forall j \in J$. We give a dynamic programming algorithm for the problem.

Consider the jobs ordered by their deadline, i.e.

$$d_j \leq d_{j+1} \quad \forall \ 1 \leq j \leq n-1.$$

We need to identify a subset $S \subseteq J$ of maximum cardinality that can be scheduled before deadlines.

Let $T(i, w)$ represent the smallest makespan of a feasible schedule of w jobs from the set $\{1, 2, \dots, i\}$.
A feasible schedule denotes a schedule where all jobs complete before their deadlines.

Let $T(i, w) = M > \sum_{j \in J} p_j$ if there is no feasible schedule with w jobs.

Notice that $T(i,w) = M$ for $w > i$.

Base case: $T(1,1) = \begin{cases} p_1, & \text{if } p_1 \le d_1 \\ M, & \text{if } p_1 > d_1 \end{cases}$ and $T(i,0) = 0 \quad \forall 1 \le i \le n$

Recursive computation:

$$T(i,w) = \min\{T(i-1, w-1) + p_i, T(i-1, w)\}$$

The cost of the optimal solution is $\max\{w : T(n,w) < M\}$.
The actual set of jobs scheduled by the optimal solution can be obtained in the same way as for the knapsack D.P.

(2) The case of arbitrary weights: The idea is the same, except that the subproblem is a little different:

$T(i,w)$: shortest makespan of a feasible schedule from the set of jobs $\{1, \dots, i\}$ that has a total weight <u>at least</u> w.

$T(i,0) = 0$, $1 \le i \le n$, as before.

$$T(1,w) = \begin{cases} p_1, & w \le w_1 \text{ and } p_1 \le d_1 \\ M, & \text{otherwise} \end{cases}$$

$$T(i,w) = \begin{cases} \min\{T(i-1, w-w_i) + p_i, T(i-1,w)\}, & \\ \qquad \text{if } T(i-1, w-w_i) \le d_i - p_i & \\ T(i-1,w), & \text{otherwise} \end{cases}$$

As before, the optimal cost is obtained from

$$\max\{w : T(n,w) \le M\}.$$

(2.b) FPTAS:

Let w_{max} be the maximum weight among all jobs, and let μ be an appropriately chosen value so that $\frac{w_{max}}{\mu} \le f(\varepsilon, n)$, where $f(\varepsilon,n)$ is some function polynomial in n and $\frac{1}{\varepsilon}$.

Let $w'_i = \left\lfloor \frac{w_i}{\mu} \right\rfloor \in \mathbb{Z}$.

If the D.P. algorithm is run on the problem instance with jobs $j \in J$ and triple (p_j, d_j, w'_j), it finds an optimal solution in time $O(n^2 f(\varepsilon,n))$ which is polynomial in n and $\frac{1}{\varepsilon}$.

The solution found is optimal to the rounded instance and since every weight may be off by $\mu = \frac{w_{max}}{f(\varepsilon,n)}$, it follows that the optimal solution to the original instance cannot be farther away than $n \cdot \mu = n \cdot \frac{w_{max}}{f(\varepsilon,n)}$. Denote by z' the cost of the solution found, and by z_{opt} the cost of the optimal solution.

$$\Rightarrow z' \ge z_{opt} - n\frac{w_{max}}{f(\varepsilon,n)} \ge z_{opt} - n\frac{z_{opt}}{f(\varepsilon,n)} \quad (\text{since } z_{opt} \ge w_{max})$$

For $f(\varepsilon,n) = \frac{n}{\varepsilon}$, we get $z' \ge z_{opt}(1-\varepsilon)$. The algorithm runs on $O(n^3 \frac{1}{\varepsilon})$ time, and therefore it is an FPTAS.

(5) Single machine jobs scheduling. Given:

Set of jobs J; w_j: weight, p_j: processing time for $j \in J$.

Output: a permutation (ordering) of the jobs.

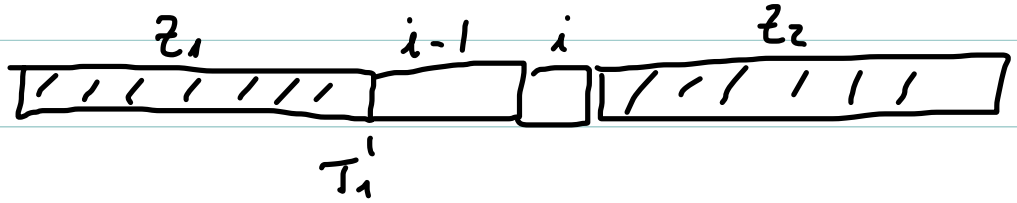
Measure: min $\sum_{j \in J} w_j C_j$, C_j: completion time.

Proof of optimality for Smith's rule.

Suppose $J = \{1, \dots, n\}$ and jobs are indexed in the order of the optimal schedule. Assume there exists $i \in \{2, \dots, n\}$ so that

$$\frac{w_{i-1}}{p_{i-1}} < \frac{w_i}{p_i}.$$

Consider the cost of the schedule with jobs i and $i-1$ swapped



Let z_{opt} be the cost of the optimal solution and z the cost of the schedule with i & $i-1$ swapped. Let z_1 & z_2 be the costs of the jobs scheduled before $i-1$ and after i respectively.

$$\Rightarrow z = z_1 + z_2 + (T_1 + p_i) w_i + (T_1 + p_i + p_{i-1}) w_{i-1}$$

$$z_{opt} = z_1 + z_2 + (T_1 + p_{i-1}) w_{i-1} + (T_1 + p_i + p_{i-1}) w_i$$

$$\Rightarrow z_{opt} - z = T_1 \cancel{(w_{i-1} - w_i)} + p_{i-1} w_{i-1} - p_i w_i + T_1 \cancel{(w_i - w_{i-1})} +$$

$$+ (p_i + p_{i-1})(w_i - w_{i-1}) =$$

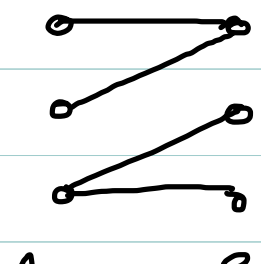
$$= p_{i-1} (w_i - \cancel{w_{i-1}} + \cancel{w_{i-1}}) + p_i (\cancel{w_i} - w_{i-1} - \cancel{w_i}) =$$

$= p_{i-1} \cdot w_i - p_i w_{i-1}$.

$\Rightarrow \frac{z_{opt} - z}{w_i \cdot w_{i-1}} = \frac{p_{i-1}}{w_{i-1}} - \frac{p_i}{w_i} > 0$ because $0 < \frac{w_{i-1}}{p_{i-1}} < \frac{w_i}{p_i}$

$\Rightarrow z_{opt} > z$, a contradiction.

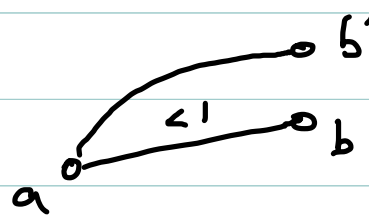
(4.1) Let $G = (A, B, E)$ be a bipartite graph with $|A| \leq |B|$. Let c_{ij} denote the cost of edge (i, j)



This graph does not admit a complete matching, but it admits a matching of size $2 < |A|$.

(4.2) Suppose that graph G admits a complete matching and that the LP relaxation is feasible.

Assume there exists a variable x_{ab} in the optimal solution of the LP relaxation with $0 < x_{ab} < 1$.



Since

$$\sum_{j \in B} x_{aj} = 1$$

$\Rightarrow \exists\, b' \in B, \quad 0 < x_{ab'} < 1$

Without loss of generality,

assume $c_{ab} \leq c_{ab'}$. Then the solution x'_{ij} which is equal to x_{ij} except for

$$\begin{cases} x'_{ab'} = 0 & \text{and} \\ x'_{ab} = x_{ab} + x_{ab'} \end{cases}$$

is also feasible, its cost is no larger than the cost of x_{ij}, and the number of fractional variables of x'_{ij} is smaller by one. Proceeding this way, all fractional variables can be eliminated.

(5) The proof of integrality of the extreme points of the LP relaxation of complete bipartite matching is immediate from Problem 4 above.

Suppose x is an extreme point and it is fractional.

$\Rightarrow \exists (a,b) \in E \quad 0 < x_{ab} < 1.$

As before $\Rightarrow \exists (a,b') \in E \quad 0 < x_{ab'} < 1.$

We will construct two feasible solutions x^1 and x^2 (vectors, the superscript is not power notation) so that $x = \lambda x^1 + (1-\lambda) x^2$, which is a contradiction.

Let $x^1_{ij} = x^2_{ij} = x_{ij}$ for $i \neq a$ or $j \notin \{b, b'\}$;

Denote $\alpha = x_{ab} + x_{ab'} \leq 1$ and let $\lambda = \frac{x_{ab}}{\alpha}$.

Let $x^1_{ab} = \alpha$, $x^1_{ab'} = 0$, $x^2_{ab} = 0$, $x^2_{ab'} = \alpha$.

Observe that $x_{ab} + x_{ab'} = x^1_{ab} + x^1_{ab'} = x^2_{ab} + x^2_{ab'} = \alpha$,
so x^1 and x^2 are feasible solutions and

$$x = \lambda x^1 + (1-\lambda) x^2.$$