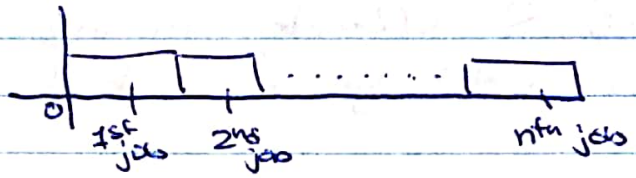
(2) Basic Single Machine Setup

Basic Single Machine (BSM) Setup

The BSM setup consists of:

- 1 M/C
- All jobs are released at time 0.
- No precedence constraints between jobs forcing that some jobs must complete before others

Objective: Minimize total completion time $\sum C_j$.



$$\sum C_j = (P_{1st\,job}) \cdot n + (P_{2nd\,job}) \cdot (n-1) + \ldots + (P_{nth\,job}) \cdot 1 \quad (1)$$

Defn 2.1 (Shortest Processing time - SPT)
The SPT rule is to schedule jobs in increasing order of processing time (breaking ties arbitrarily)

Theorem 2.1:
In the BSM setup, a schedule minimizes $\sum C_j$ iff it is produced via the SPT rule. (called an "SPT schedule")

[Proof]
We will prove this by an interchange argument.

($\Rightarrow$) Suppose we have a schedule S that minimizes $\sum C_j$, but is NOT an SPT schedule

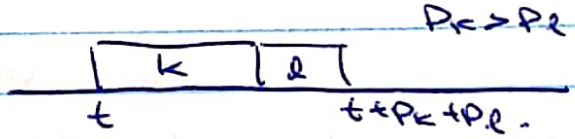
Cont'd

Proof (cont'd)

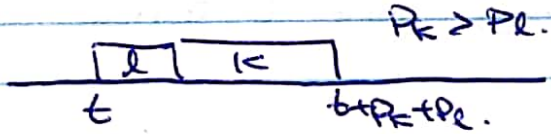
Then, there must be jobs j, j' such that $p_j > p_{j'}$, but j is scheduled before j'. (We call this an inversion). In fact there must be 2 consecutive jobs k, l s.t. $p_k > p_l$, but k is scheduled immediately before l in S. (Consider moving inwards from j, j' until you find k, l).

Let S' be the schedule obtained from S by interchanging k, l

S:



S':



Consider the change in the objective value:

$$\sum_j C_j^{S'} - \sum_j C_j^S$$

$C_j^{S'}$ denotes completion time of j in S'

$$= (C_k^{S'} + C_l^{S'}) - (C_k^S + C_l^S)$$

Since $C_j^{S'} = C_j^S \quad \forall j \neq k, l$

$$= t + p_k + p_l + t + p_l - (t + p_k + t + p_k + p_l)$$

$$= p_l - p_k < 0, \quad \text{since } p_k > p_l.$$

which contradicts that S is an optimal schedule.

Cont'd

[Proof] Cont'd

So, we have shown that:

- Every optimal schedule is an SPT schedule
- All SPT schedules have the same objective value — We see this because we can always move from SPT schedule to another by interchanging pairs of jobs with equal processing times. The objective value is unchanged

These 2 statements imply that that all SPT schedules are optimal (which is exactly the reverse direction) □

Objective: Minimize $\sum w_j C_j$

(Note: If $w_j = p_j$, then this problem reduces to the previous one)

Def'n 2.2 (Weighted Shortest Processing time - WSPT)

The weighted shortest processing time rule is to schedule jobs in decreasing order of w_j/p_j. (Known as the density of job j)

Theorem 2.2:

In the BSM setup, a schedule minimizes $\sum w_j C_j$ iff it is a WSPT schedule

[Proof]

We will use an interchange argument.

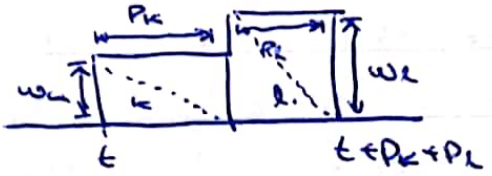
($\Rightarrow$) Let S be an optimal schedule that minimizes $\sum w_j C_j$, but is NOT a WSPT schedule.

(Cont'd)

[Proof] (cont'd)

There must be consecutive jobs k, l s.t. k is scheduled immediately before l in S, but $w_k/p_k < w_l/p_l$.

S:



S':

Let S' be obtained from S by interchanging k, l. Then we get:

$$\sum w_j C_j^{S'} - \sum w_j C_j^{S}$$

$$= w_k C_k^{S'} + w_l C_l^{S'} - (w_k C_k^{S} + w_l C_l^{S})$$

$$= w_k (t + p_k + p_l) + w_l (t + p_l) - (w_k (t + p_k) + w_l (t + p_k + p_l))$$

$$= w_k p_l - w_l p_k$$

$$= p_l p_k \left(\frac{w_k}{p_k} - \frac{w_l}{p_l} \right) < 0, \text{ since } \frac{w_k}{p_k} < \frac{w_l}{p_l}.$$

which contradicts that S is an optimal schedule.

This also shows that interchanging equal density jobs preserves the objective value. Hence, all WSPT schedules have the same objective value, and so are optimal schedules; completing the proof.

□

Objective: Minimize maximum lateness

Recall that lateness is:

$$L_j := C_j - d_j$$

So, we minimize:

$$L_{max} := \max_j L_j$$

Defn 2.3 (Earliest Due Date Rule - EDD)

The EDD rule is to schedule jobs in increasing order of d_j.

Theorem 2.3

The EDD rule minimizes L_{max} in the BSH setup

Note: There are optimal schedules that minimize L_{max}, but NOT EDD schedules.

[Proof]

Let S be an optimal schedule, but is NOT an EDD schedule. Then, there are consecutive jobs k, l s.t.:

1) k is scheduled immediately before l in S.
2) $d_k > d_l$

Let S' be constructed from S by interchanging k, l. (We want to show $L_{max}^{S'} \leq L_{max}^{S}$)

We know that:

- $\forall j \neq k, l$, $L_j^{S'} = L_j^S$ (Since $C_j^{S'} = C_j^S$)
- $L_k^S = C_k^{S'} - d_k = C_j^S - d_k < C_l^S - d_l = L_l^S$
  $\Rightarrow L_k^{S'} < L_l^S$
- $L_l^{S'} = C_l^{S'} - d_l \leq C_l^S - d_l = L_l^S$

So,

$$\max(L_k^{S'}, L_l^{S'}) \leq L_l^S \leq \max(L_k^S, L_l^S)$$

$$\Rightarrow \max_j L_j^{S'} \leq \max_j L_j^S$$

□

Objective: Minimize f_{max}

We have a non-decreasing function:

$f_j : \mathbb{R}_{\geq 0} \to \mathbb{R}$ for each job j.

Where:

$f_j(C_j)$, $C_j :=$ Completion time for j

is the cost incurred for job j.

We want a schedule that minimizes:

$$f_{max} := \max_j f_j(C_j)$$

(Note: With $f_j(x) = x - d_j$, we retrieve our previous problem of minimizing max. lateness)

Motivation for LCh Rule:

We define:

$f^*_{max}(J) :=$ objective value of an optimal schedule for a set of jobs J

(

And also "relax" our definition of schedule:

A schedule specifies when each job in J starts (and hence finishes) and should be s.t. at most one job is processed at any point of time.

(Note: This allows us to have gaps between jobs)

We notice that:

An optimal schedule for J completes at time: $\sum_{j \in J} p_j$.

→ continued

(continued)
So, if r is the last job scheduled in an optimal schedule for J, then:

$$f^*_{max}(J) \geq f_r(C_r)$$
$$= f_r\left(\sum_{j \in J} p_j\right)$$
$$\geq \min_{k \in J} f_k\left(\sum_{j \in J} p_j\right) \quad (1)$$

And, for any $k \in J$,

$$f^*_{max}(J) \geq f^*_{max}(J \setminus \{k\}) \quad (2)$$

Since dropping jobs cannot increase the objective value.
So, combining (1) and (2), we get:
Let $l \in J$ be s.t.:

$$f_l\left(\sum_{j \in J} p_j\right) = \min_{k \in J} f_k\left(\sum_{j \in J} p_j\right)$$

Then,

$$f^*_{max}(J) \geq \max\left(f_l\left(\sum_{j \in J} p_j\right), f^*_{max}(J \setminus \{l\})\right)$$

Notice that this suggests that l should be the last job since $C_l = \sum_{j \in J} p_j$, and this also gives us that:

$$f^*_{max}(J) = \max\left(f_l\left(\sum_{j \in J} p_j\right), f^*_{max}(J \setminus \{l\})\right)$$

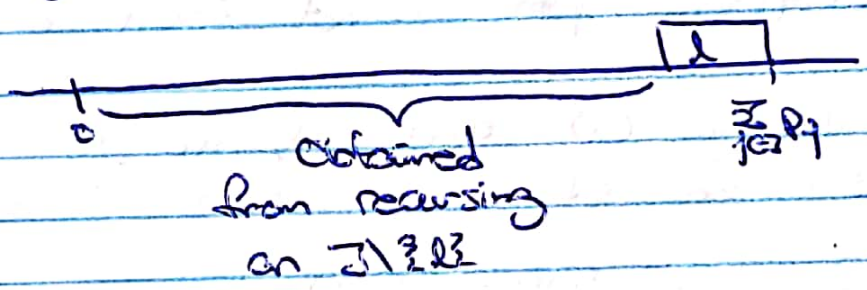
↳ def'n

Defn 2.4 (Least Cost last Rule - LCL)

- $J \leftarrow \{1, \ldots, n\}$
- While $J \neq \emptyset$:
  - Choose $l \in J$ s.t
    $$f_l\left(\sum_{j \in J} p_j\right) = \min_{k \in J} f_k\left(\sum_{k \in J} p_j\right)$$
  - Schedule l last among J (so that l completes at time $\sum_{j \in J} p_j$)
  - $J \leftarrow J \setminus \{l\}$

Note:

(i) Visually, we get the following:



(ii) There are 2 ways of constructing schedules: from time 0 to $\sum_{j \in J} p_j$ (i.e. front to back) or from time $\sum_{j \in J} p_j$ to time 0 (i.e from back to front). In this case, scheduling from the back is more convenient.

Co Theorem

Theorem 2.4

In the BSM setup, every LCL schedule for a job-set J has objective value $\leq f^*_{max}(J)$, hence every LCL schedule minimizes f_{max}.

[Proof]

We prove by induction on $|J|$.

(BC) If $|J|=1$, then the statement is trivially true

(IH) Now, suppose this statement holds when $|J|=n$.

(IC) Consider job-set J with $|J|=n+1$.

Let l be as in LCL s.t.:

$$f_l\left(\sum_{j\in J} p_j\right) = \min_{k\in J} f_k\left(\sum_{j\in J} p_j\right)$$

And the objective value of the LCL schedule will be:

$$\max\left(f_l\left(\sum_{j\in J} p_j\right),\ f_{max} \text{ in schedule constructed by LCL for } J\setminus\{l\}\right)$$

$$\leq \max\left(f_l\left(\sum_{j\in J} p_j\right),\ f^*_{max}(J\setminus\{l\})\right) \quad (\text{by (IH)})$$

$$\leq f^*_{max}(J)$$

which completes the induction step

□