Scheduling Problems [Greedy algorithms (Contd)]

Example I

A single machine, n jobs to be processed.

Input: $t_i$: processing time of job $i$, $i = 1, \ldots, n$.


Preemption not allowed. So when a job is done, another is loaded immediately after it.

Gantt charts

Suppose we process jobs in order: $1, 2, 3$.

Completion times:

\[
C_i(S) = 3 + 13 + 18 = 34.
\]

\[
\sum_{i=1}^{3} C_i(S) = 3 + 13 + 18 = 34.
\]

Suppose instead we process in order: $1, 3, 2$.

Completion times:

\[
C_i(S') = 3 + 8 + 18 = 29.
\]

\[
\sum_{i=1}^{3} C_i(S') = 3 + 8 + 18 = 29.
\]

$S'$ is better than $S$ in this problem.

(Minimizing $\sum C_i(S)$ increases throughput of the system)

And this is our goal.
**Greedy algorithm**

Let $P = (P_1, P_2, \ldots, P_n)$ be a permutation of $\{1, 2, \ldots, n\}$

$S_i = P_i, \quad i = 1, \ldots, n$

$$
\sum_{i=1}^{n} C_i = S_1 + (S_1 + S_2) + (S_1 + S_2 + S_3) + \ldots
$$

$$
= n S_1 + (n-1) S_2 + \ldots
$$

$$
= \sum_{k=1}^{n} (n-k+1) S_k \quad \leftarrow \text{Want } P \text{ that minimizes this.}
$$

**Algorithm**  
**Shortest Processing Time Rule (SPT)**

Arrange jobs so that processing times are non-decreasing order and process in this order. Complexity: $\Theta(n \log n)$

**Proof of Correctness**

Our method of proof for this is to show any non-SPT schedule is not optimal. Hence SPT is optimal.

Towards this end, let $S$ be a non-SPT schedule.

In $S$, there is a pair of successive jobs $i, j$ (in that order) such that $t_i > t_j$. $S$ looks like

Consider $S' = \ldots i \quad j \quad \ldots$

$s'$ found by interchanging $i, j$
It is "clear" that

c_k(s) = c_k(s') \quad k \neq i, j

c_i(s) = \alpha + t_i; \quad c_j(s) = \alpha + t_i + t_j

c_j(s') = \alpha + t_j; \quad c_i(s') = \alpha + t_j + t_i

\therefore \sum_{k} c_k(s) \geq \sum_{k} c_k(s')

\therefore c_i(s) + c_j(s) \geq c_i(s') + c_j(s')

\therefore \alpha + t_i + \alpha + t_j \geq \alpha + t_j + \alpha + t_i

\therefore t_i \leq t_j

\therefore \sum_{k} c_k(s) > \sum_{k} c_k(s')

\therefore S is not optimal. \therefore All Non-SPT Schedule are not optimal.

\therefore SPT is optimal. (Interchanging Equal one, does not change value)

An extension to multimachine case is in HW # 4.
Example II: This involves a "greedy" algorithm using another "greedy" subroutine inside it.

Problem: One m/c, non-preemptive scheduling, n jobs; each takes 1 unit of time.

Input: \(i^{th}\) job: \(d_i\), \(P_i\), \(d_i = 1 \ldots n\)

duedate  Profit
deadline

Desired output: Maximize total profit when, \(P_i\) in accrued only when it is completed on or before \(d_i\)

1. (e) \(C_{j}(S) \leq d_i\) if a job is late, 0 profit.

Greedy Algorithm:
Start with an empty set. Process jobs in decreasing order of \(P_i\); include the next job provided this together with already included jobs can all be done by their respective due dates. If not, reject this job and go to the next.

Now we illustrate this with a numerical example.
Numerical Example

Profit: \( P = [50, 10, 15, 30] \); \( d = [2, 1, 2, 1] \)

Recall all jobs take 1 unit of processing time
(Unit Execution Time System (UET))

\[
\begin{align*}
\text{Step 1: } & \{0\} \text{ is done before 2. its due date accept} \\
\text{Step 2: } & \text{Check if we can do both } \{1, 4\} \text{?} \\
& \text{Can process in any order. } \{4\} \text{ yes include 4.} \\
\text{Step 3: } & \{0, 4, 3\} \text{ No} \\
\text{Step 4: } & \{0, 4, 2\} \text{ No} \\
\text{Output: } & \{1, 4\} \\
\end{align*}
\]

It is the process of checking whether we can add the next job and meet all deadlines that requires a subroutine → done using Greedy Algorithm again.

This is addressed on the next page.
Sub-Problem (generalized):

One m/c, n jobs.

\( i^{th} \) job: \( d_i, t_i \)

deadline proc. time (recall: main problem has \( t_i = 1 \ \forall i \) but not here)

Q.1: Can all jobs be completed by their respective due dates?

Q.2: What if we change \( d'_i = d_i + \Delta \) (\( \Delta \) same for all jobs): What is Min \( \Delta \)?

Q.3: Maximum # that can be completed by deadline?

(There are more; done in Scheduling course)

Here we need answer to Q.1.

Earliest Due Date order Schedule (EDD) (\( \Theta(n\log n) \))

Arrange jobs in increasing order of due dates.

If any schedule meet all deadlines, EDD is one of them. Proof & correctness on next page.
To check if all deadlines are met in EDD
(Suppose jobs are renumbered so that
\[ d_1 \leq d_2 \leq \ldots \leq d_n. \]

We need to check:
\[ t_i \leq d_i, \]
\[ t_i + t_{i+1} \leq d_{i+1} \]
\[ \vdots \]
\[ t_i + \ldots + t_n \leq d_n \]

\[ \text{Complexity?} \]

\[ \text{If yes to all above questions, EDD meets all deadlines.} \]

\[ \text{Pf Suppose a job (or move) is late in EDD.} \]
\[ \text{Let } k \text{ be the first late job.} \]

\[ \text{EDD} \]
\[ \text{Suppose } S \text{ meets all deadlines.} \]
\[ \text{Some one move to right} \]
\[ \text{Suppose } S \text{ meets all deadlines.} \]
\[ \text{Some one move to right} \]
\[ \text{Hence if some job is late in EDD,} \]
\[ \text{some job will be late in } S \text{ vs.} \]
Implementation of this alg. (for the main problem) requires UNION-FIND data structure (will be covered later in Graph Algorithms).

New proof of correctness of Main algorithm.

Suppose GA selects a set \( I \subseteq \{1, \ldots, n\} \) of jobs.
Suppose Opt (max profit) set is \( J \subseteq \{1, \ldots, n\} \).

Note: \( I \cap J = \emptyset \), \( I - J \), \( J - I \) may all be non-empty.

Moreover, \( \exists S_I, S_J \) that show all of \( I \) or all of \( J \) can meet deadlines.

Suppose \( I = \{p, q, r, x, y\} \)
\( J = \{p, q, r, s, t, u, v, w\} \)
\( I \cap J = \{p, q, r\} \) Common jobs
\( I - J = \{x, y\} \)
\( J - I = \{s, t, u, v, w\} \)

We do not know which set has more jobs.
Only total profit in \( I \) or \( J \) is important.

For the sake of argument, I will depict \( S_I \) and \( S_J \) in the next page.
JOJ in Same time slot in both Schedules.

So without loss, we can assume Common jobs are done in the same time slot in both Schedules.

Now we compare jobs in the same time slots & their profits:

\[ a = b \]
\[ p_a = p_b \]

\[ a' \neq b' \]
\[ 3 \text{ Case, } p_a = p_b \]
\[ p_a > p_b, p_a < p_b \]

Show can't happen.
\& \text{If } P_a > P_b, \text{ replacing } b \text{ in } J \text{ by } a \quad (a \text{ cannot be in } J \text{ since all common jobs are in same time slot}), \text{ we get a higher profit than } J\text{ and new set meets all deadlines}. So this can not happen.

\& \text{If } P_b > P_a, \text{ greedy alg. consider } b \text{ before } a.

\text{Considering } a, \text{ it should have selected } b \text{ but } b \notin J. \text{ So greedy alg. was not done properly. So this case is not possible either.}

\therefore P_a = P_b \quad \text{Similar results for each time slot.}

\therefore \text{total profit } = \text{total profit of } J + \text{opt. profit by assumption}.

\therefore \text{Greedy alg. works correctly.}

\text{End of Greedy Algorithm.}