Single machine models: Number of Tardy Jobs

Problem 1||∑ Uj:

- Structure of an optimal schedule:
  - set S₁ of jobs meeting their due dates
  - set S₂ of jobs being late
  - jobs of S₁ are scheduled before jobs from S₂
  - jobs from S₁ are scheduled in EDD order
  - jobs from S₂ are scheduled in an arbitrary order

- Result: a partition of the set of jobs into sets S₁ and S₂ is sufficient to describe a solution
Single machine models: Number of Tardy Jobs

Algorithm 1 $||\sum U_j$:

1. enumerate jobs such that $d_1 \leq \ldots \leq d_n$;
2. $S_1 := \emptyset; \ t := 0$;
3. FOR j:=1 TO n DO
4. $\quad S_1 := S_1 \cup \{j\}; \ t := t + p_j$;
5. $\quad$ IF $t > d_j$ THEN
6. $\quad$ Find job $k$ with largest $p_k$ value in $S_1$;
7. $\quad S_1 := S_1 \backslash \{k\}; \ t := t - p_k$;
8. $\quad$ END
9. END
**Single machine models:** Number of Tardy Jobs

**Remarks Algorithm 1||∑U_j**

- Principle: schedule jobs in order of increasing due dates and always when a job gets late, remove the job with largest processing time; all removed jobs are late

- complexity: \(O(n \log(n))\)

- Example: \(n = 5;\) \(p = (7, 8, 4, 6, 6);\) \(d = (9, 17, 18, 19, 21)\)
**Single machine models**: Number of Tardy Jobs

Remarks Algorithm 1 || $\sum U_j$

- Principle: schedule jobs in order of increasing due dates and always when a job gets late, remove the job with largest processing time; all removed jobs are late
- Complexity: $O(n \log(n))$
- Example: $n = 5; p = (7, 8, 4, 6, 6); d = (9, 17, 18, 19, 21)$

![Diagram of job scheduling](attachment:job_scheduling_diagram.png)
Single machine models: Number of Tardy Jobs

Remarks Algorithm 1 || $\sum U_j$

- Principle: schedule jobs in order of increasing due dates and always when a job gets late, remove the job with largest processing time; all removed jobs are late

- Complexity: $O(n \log(n))$

- Example: $n = 5$; $p = (7, 8, 4, 6, 6)$; $d = (9, 17, 18, 19, 21)$

\[
\begin{array}{cccccc}
0 & 1 & 3 & 4 & 5 & d_5 \\
5 & 10 & 15 & 20 & &
\end{array}
\]
**Single machine models**: Number of Tardy Jobs

**Remarks Algorithm 1||\(\sum U_j\)**

- Principle: schedule jobs in order of increasing due dates and always when a job gets late, remove the job with largest processing time; all removed jobs are late

- complexity: \(O(n \log(n))\)

- Example: \(n = 5\); \(p = (7, 8, 4, 6, 6)\); \(d = (9, 17, 18, 19, 21)\)

- Algorithm 1||\(\sum U_j\); computes an optimal solution

Proof on the board
Single machine models: Weighted Number of Tardy Jobs

Problem 1|| $\sum w_j U_j$

- problem 1|| $\sum w_j U_j$ is NP-hard even if all due dates are the same; i.e. 1|d_j = d| $\sum w_j U_j$ is NP-hard
  Proof on the board (reduction from PARTITION)

- priority based heuristic (WSPT-rule):
  schedule jobs in decreasing $w_j/p_j$ order
Single machine models: Weighted Number of Tardy Jobs

Problem 1\|\sum w_j U_j

- problem 1\|\sum w_j U_j is NP-hard even if all due dates are the same; i.e. 1|d_j = d|\sum w_j U_j is NP-hard
  
  Proof on the board (reduction from PARTITION)

- priority based heuristic (WSPT-rule):
  schedule jobs in decreasing \( w_j/p_j \) order

- WSPT may perform arbitrary bad for 1\|\sum w_j U_j:
**Single machine models:** Weighted Number of Tardy Jobs

**Problem 1||Σ w_jU_j**

- problem 1||Σ w_jU_j is NP-hard even if all due dates are the same; i.e. 1|d_j = d|Σ w_jU_j is NP-hard
  Proof on the board (reduction from PARTITION)

- priority based heuristic (WSPT-rule):
  schedule jobs in decreasing w_j/p_j order

- WSPT may perform arbitrary bad for 1||Σ w_jU_j:
  \[ n = 3; p = (1, 1, M); w = (1+\epsilon, 1, M-\epsilon); d = (1+M, 1+M, 1+M) \]
  \[ \sum w_jU_j(WSPT) / \sum w_jU_j(opt) = (M - \epsilon)/(1 + \epsilon) \]
Single machine models: Weighted Number of Tardy Jobs

Dynamic Programming for $1|| \sum w_j U_j$

- assume $d_1 \leq \ldots \leq d_n$
- as for $1|| \sum U_j$ a solution is given by a partition of the set of jobs into
  sets $S_1$ and $S_2$ and jobs in $S_1$ are in EDD order
- Definition:
  $$-F_j(t) := \text{minimum criterion value for scheduling the first } j \text{ jobs such that the processing time of the on-time jobs is at most } t$$
- $F_n(T)$ with $T = \sum_{j=1}^{n} p_j$ is optimal value for problem $1|| \sum w_j U_j$
- Initial conditions:
  $$F_j(t) = \begin{cases} 
\infty & \text{for } t < 0; \ j = 1, \ldots, n \\
0 & \text{for } t \geq 0; \ j = 0
\end{cases} \quad (1)$$
Single machine models: Weighted Number of Tardy Jobs

Dynamic Programming for $1|| \sum w_j U_j$ (cont.)

- if $0 \leq t \leq d_j$ and $j$ is late in the schedule corresponding to $F_j(t)$, we have $F_j(t) = F_{j-1}(t) + w_j$

- if $0 \leq t \leq d_j$ and $j$ is on time in the schedule corresponding to $F_j(t)$, we have $F_j(t) = F_{j-1}(t - p_j)$
Single machine models: Weighted Number of Tardy Jobs

Dynamic Programming for $1||\sum w_j U_j$ (cont.)

- if $0 \leq t \leq d_j$ and $j$ is late in the schedule corresponding to $F_j(t)$, we have $F_j(t) = F_{j-1}(t) + w_j$

- if $0 \leq t \leq d_j$ and $j$ is on time in the schedule corresponding to $F_j(t)$, we have $F_j(t) = F_{j-1}(t - p_j)$

- summarizing, we get for $j = 1, \ldots, n$:

$$F_j(t) = \begin{cases} 
\min\{F_{j-1}(t - p_j), F_{j-1}(t) + w_j\} & \text{for } 0 \leq t \leq d_j \\
F_j(d_j) & \text{for } d_j < t \leq T \end{cases}$$
**Single machine models**: Weighted Number of Tardy Jobs

DP-algorithm for $1||\sum w_j U_j$

1. initialize $F_j(t)$ according to (1)
2. FOR $j := 1$ TO $n$ DO
3. FOR $t := 0$ TO $T$ DO
4. update $F_j(t)$ according to (2)
5. $\sum w_j U_j(OPT) = F_n(d_n)$
**Single machine models:** Weighted Number of Tardy Jobs

DP-algorithm for $1||\sum w_jU_j$

1. initialize $F_j(t)$ according to (1)
2. FOR $j := 1$ TO $n$ DO
3. FOR $t := 0$ TO $T$ DO
4. update $F_j(t)$ according to (2)
5. $\sum w_jU_j(OPT) = F_n(d_n)$

- complexity is $O(n \sum_{j=1}^{n} p_j)$
- thus, algorithm is pseudopolynomial
Single machine models: Total Tardiness

Basic results:

• 1||\(\sum T_j\) is NP-hard

• preemption does not improve the criterion value

  → 1|\(pmtn|\sum T_j\) is NP-hard

• idle times do not improve the criterion value

• Lemma 1: If \(p_j \leq p_k\) and \(d_j \leq d_k\), then an optimal schedule exist in which job \(j\) is scheduled before job \(k\).

  Proof: exercise

• this lemma gives a dominance rule
Single machine models: Total Tardiness

Structural property for $1||\sum T_j$

- let $k$ be a fixed job and $\hat{C}_k$ be latest possible completion time of job $k$ in an optimal schedule

- define

$$\hat{d}_j = \begin{cases} 
  d_j & \text{for } j \neq k \\
  \max\{d_k, \hat{C}_k\} & \text{for } j = k
\end{cases}$$

- Lemma 2: Any optimal sequence w.r.t. $\hat{d}_1, \ldots, \hat{d}_n$ is also optimal w.r.t. $d_1, \ldots, d_n$.

Proof on the board
Single machine models: Total Tardiness

Structural property for $1||\sum T_j$ (cont.)

- let $d_1 \leq \ldots \leq d_n$
- let $k$ be the job with $p_k = \max\{p_1, \ldots, p_n\}$
- Lemma 1 implies that an optimal schedule exists where
  \[ \{1, \ldots, k - 1\} \rightarrow k \]
- **Lemma 3**: There exists an integer $\delta$, $0 \leq \delta \leq n - k$ for which an optimal schedule exist in which
  \[ \{1, \ldots, k - 1, k + 1, \ldots, k + \delta\} \rightarrow k \text{ and } k \rightarrow \{k + \delta + 1, \ldots, n\} \]

Proof on the board
Single machine models: Total Tardiness

DP-algorithm for $1||\sum T_j$

- Definition:
  
  $F_j(t) := \text{minimum criterion value for scheduling the first } j \text{ jobs}$
  
  starting their processing at time $t$

- by Lemma 3 we get:
  
  there exists some $\delta \in \{1, \ldots, j\}$ such that $F_j(t)$ is achieved by
  
  scheduling

  1. first jobs $1, \ldots, k - 1, k + 1, \ldots, k + \delta$ in some order
  2. followed by job $k$ starting at $t + \sum_{l=1}^{k+\delta} p_l - p_k$
  3. followed by jobs $k + \delta + 1, \ldots, j$ in some order

  where $p_k = \max_{l=1}^{j} p_l$
Single machine models: Total Tardiness

DP-algorithm for $1|| \sum T_j$ (cont.)

- Definition:

$$J(j, l, k) := \{i | i \in \{j, j+1, \ldots, l\}; p_i \leq p_k; i \neq k\}$$
Single machine models: Total Tardiness

DP-algorithm for $1||\sum T_j$ (cont.)

- Definition:
  \[
  J(j, l, k) := \{i | i \in \{j, j+1, \ldots, l\}; p_i \leq p_k; i \neq k\}
  \]
**Single machine models**: Total Tardiness

DP-algorithm for $1||\sum T_j$ (cont.)

- **Definition:**
  - $J(j, l, k) := \{i | i \in \{j, j + 1, \ldots, l\}; p_i \leq p_k; i \neq k\}$
  - $V(J(j, l, k), t) :=$ minimum criterion value for scheduling the jobs from $J(j, l, k)$ starting their processing at time $t$
**Single machine models:** Total Tardiness

DP-algorithm for $1||\sum T_j$ (cont.)

- we get:
  $$V(J(j, l, k), t) = \min_\delta \left\{ V(J(j, k' + \delta, k'), t) + \max\{0, C_{k'}(\delta) - d_{k'}\} + V(J(k' + \delta + 1, l, k'), C_{k'}(\delta)) \right\}$$

  where $p_{k'} = \max\{p_{j'}|j' \in J(j, l, k)\}$ and
  $$C_{k'}(\delta) = t + p_{k'} + \sum_{j' \in V(J(j, k' + \delta, k'))} p_{j'}$$

- $V(\emptyset, t) = 0$, $V(\{j\}, t) = \max\{0, t + p_j - d_j\}$
Single machine models: Total Tardiness

DP-algorithm for $1||\sum T_j$ (cont.)

• optimal value of $1||\sum T_j$ is given by $V(\{1, \ldots, n\}, 0)$

• complexity:
  - at most $O(n^3)$ subsets $J(j, l, k)$
  - at most $\sum p_j$ values for $t$
  - each recursion (evaluation $V(J(j, l, k), t)$) costs $O(n)$ (at most $n$ values for $\delta$)

  total complexity: $O(n^4 \sum p_j)$ (pseudopolynomial)