Greedy (2)

Nov 9th, 2017

Algorithm Design and Analysis

YUN-NUNG (VIVIAN) CHEN HTTP://ADA17.CSIE.ORG

國立臺灣大學
National Taiwan University

Slides credited from Hsu-Chun Hsiao

Outline

- Greedy Algorithms
- Greedy #1: Activity-Selection / Interval Scheduling
- Greedy #2: Coin Changing
- Greedy #3: Fractional Knapsack Problem
- Greedy #4: Breakpoint Selection
- Greedy #5: Huffman Codes
- Greedy #6: Scheduling to Minimize Lateness
- Greedy #7: Task-Scheduling

Greedy Algorithms

To yield an optimal solution, the problem should exhibit

1. **Greedy-Choice Property** : making locally optimal (greedy) choices leads to a globally optimal solution
2. **Optimal Substructure** : an optimal solution to the problem contains within it optimal solutions to subproblems

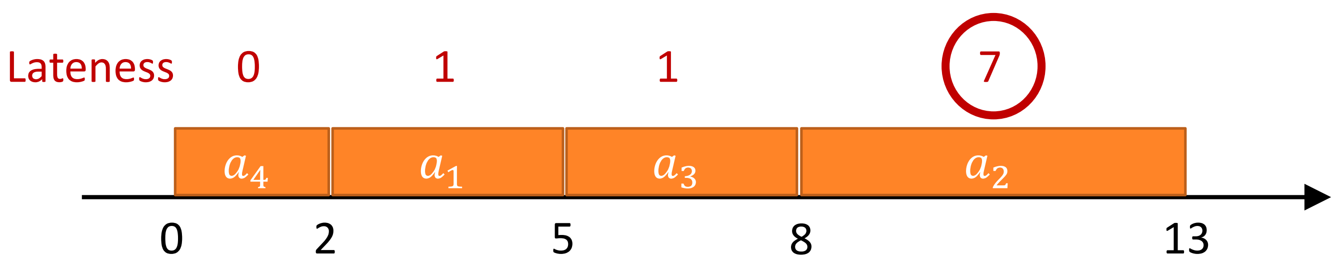
4 Scheduling to Minimize Lateness

Scheduling to Minimize Lateness

- Input: a finite set $S = \{a_1, a_2, \dots, a_n\}$ of n tasks, their processing time $t_1, t_2, \dots, t_n$, and integer deadlines $d_1, d_2, \dots, d_n$

Job	**1**	**2**	**3**	**4**
Processing Time (t_i)	3	5	3	2
Deadline (d_i)	4	6	7	8

- Output: a schedule that minimizes the maximum lateness



Scheduling to Minimize Lateness

Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \dots, t_n$, and deadlines $d_1, d_2, \dots, d_n$
Output: the schedule that minimizes the maximum lateness

- Let a schedule H contains $s(H, j)$ and $f(H, j)$ as the start time and finish time of job j
  - $f(H, j) - s(H, j) = t_j$
  - Lateness of job j in H is $L(H, j) = \max\{0, f(H, j) - d_j\}$
- The goal is to minimize $\max_j L(H, j) = \max_j \{0, f(H, j) - d_j\}$

Possible Greedy Choices

Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \dots, t_n$, and deadlines $d_1, d_2, \dots, d_n$
Output: the schedule that minimizes the maximum lateness

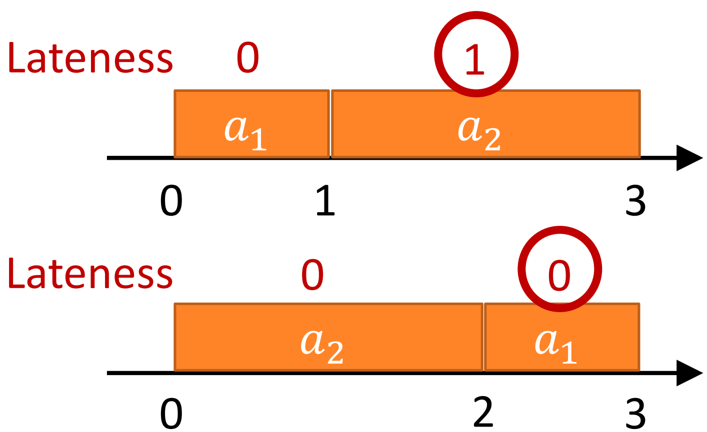
- Idea
  - Shortest-processing-time-first w/o idle time?
  - Earliest-deadline-first w/o idle time?

Practice: prove that any schedule w/ idle is not optimal

Possible Greedy Choices

Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \ldots, t_n$, and deadlines $d_1, d_2, \ldots, d_n$
Output: the schedule that minimizes the maximum lateness

- Idea
  - **Shortest-processing-time-first w/o idle time?**



Job	1	2
Processing Time (t_i)	1	2
Deadline (d_i)	10	2

Possible Greedy Choices

Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \dots, t_n$, and deadlines $d_1, d_2, \dots, d_n$
Output: the schedule that minimizes the maximum lateness

- Idea
  - **Earliest-deadline-first w/o idle time?**
- Greedy algorithm

```
Min-Lateness(n, t[], d[])
  sort tasks by deadlines s.t. d[1]≤d[2]≤ ...≤d[n]
  ct = 0 // current time
  for j = 1 to n
    assign job j to interval (ct, ct + t[j])
    s[j] = ct
    f[j] = s[j] + t[j]
    ct = ct + t[j]
  return s[], f[]
```

$$T(n) = \Theta(n \log n)$$

Prove Correctness
– Greedy-Choice Property

Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \ldots, t_n$, and deadlines $d_1, d_2, \ldots, d_n$
Output: the schedule that minimizes the maximum lateness

- Greedy choice: first select the task with the earliest deadline
- Proof via contradiction
  - Assume that there is no OPT including this greedy choice
    - If OPT processes a_1 as the i-th task (a_k), we can switch a_k and a_1 into OPT'
  - The maximum lateness must be equal or lower, because $L(\mathrm{OPT}') \leq L(\mathrm{OPT})$

Prove Correctness
– Greedy-Choice Property

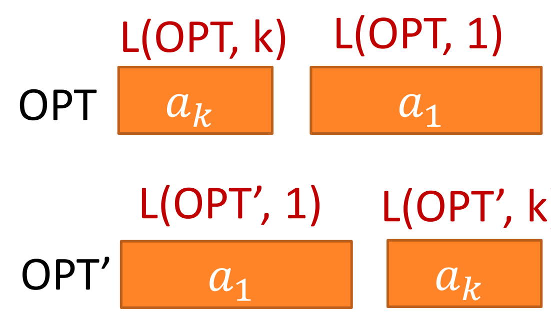
Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \dots, t_n$, and deadlines $d_1, d_2, \dots, d_n$
Output: the schedule that minimizes the maximum lateness

- $L(\text{OPT'}) \leq L(\text{OPT})$

$$\iff \max(L(\text{OPT'}, 1), L(\text{OPT'}, k)) \leq \max(L(\text{OPT}, k), L(\text{OPT}, 1))$$

$$\iff \max(L(\text{OPT'}, 1), L(\text{OPT'}, k)) \leq L(\text{OPT}, 1)$$

$$\iff L(\text{OPT'}, k) \leq L(\text{OPT}, 1) \because L(\text{OPT'}, 1) \leq L(\text{OPT}, 1)$$

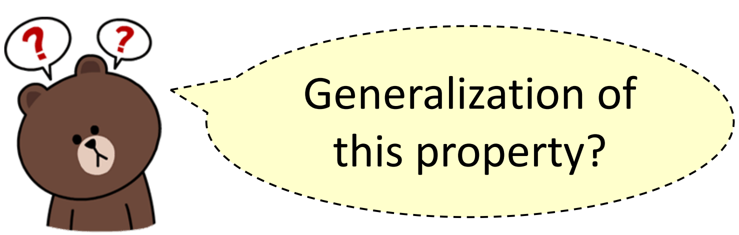


<u>If a_k is not late in OPT’:</u>

$$L(\text{OPT'}, k) = 0$$

<u>If a_k is late in OPT’:</u>

$$\begin{aligned} L(\text{OPT'}, k) &= f(\text{OPT'}, k) - d_k \\ &= f(\text{OPT}, 1) - d_k \\ &\leq f(\text{OPT}, 1) - d_1 \\ &= L(\text{OPT}, 1) \end{aligned}$$



Prove Correctness
– No Inversions

Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \dots, t_n$, and deadlines $d_1, d_2, \dots, d_n$
Output: the schedule that minimizes the maximum lateness

- There is an optimal scheduling w/o *inversions* given $d_1 \leq d_2 \leq \cdots \leq d_n$
  - a_i and a_j are *inverted* if $d_i < d_j$ but a_j is scheduled before a_i
- Proof via contradiction
  - Assume that OPT has a_i and a_j that are inverted
  - Let OPT' = OPT but a_i and a_j are swapped
  - OPT' is equal or better than OPT, because $L(\text{OPT}') \leq L(\text{OPT})$

Prove Correctness
– No Inversions

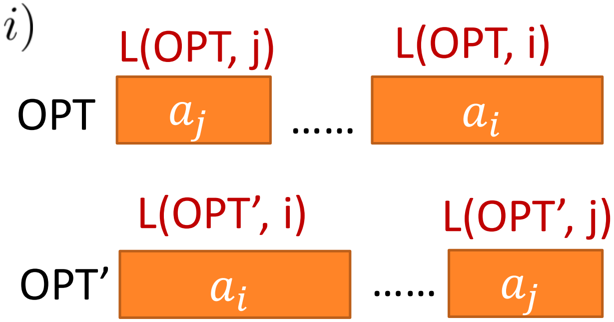
Scheduling to Minimize Lateness Problem
Input: n tasks with their processing time $t_1, t_2, \ldots, t_n$, and deadlines $d_1, d_2, \ldots, d_n$
Output: the schedule that minimizes the maximum lateness

- $L(\text{OPT'}) \leq L(\text{OPT})$

$$\iff \max(L(\text{OPT'}, i), L(\text{OPT'}, j)) \leq \max(L(\text{OPT}, j), L(\text{OPT}, i))$$

$$\iff \max(L(\text{OPT'}, i), L(\text{OPT'}, j)) \leq L(\text{OPT}, i) \because d_i < d_j$$

$$\iff L(\text{OPT'}, j) \leq L(\text{OPT}, i) \because L(\text{OPT'}, i) \leq L(\text{OPT}, i)$$



<u>If a_j is not late in OPT':</u>

$$L(\text{OPT'}, j) = 0$$

<u>If a_j is late in OPT':</u>

$$\begin{aligned} L(\text{OPT'}, j) &= f(\text{OPT'}, j) - d_j \\ &= f(\text{OPT}, i) - d_j \\ &\leq f(\text{OPT}, i) - d_i \\ &= L(\text{OPT}, i) \end{aligned}$$

Optimal Solution = Greedy Choice + Subproblem Solution

The earliest-deadline-first greedy algorithm is optimal

14 Task-Scheduling

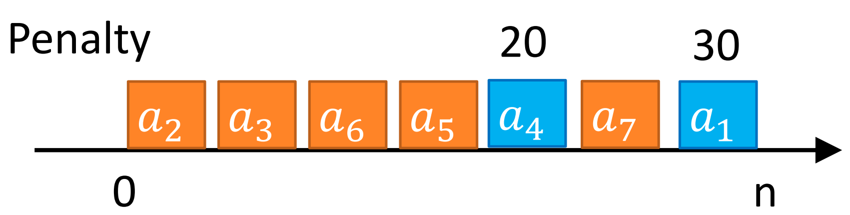
Textbook Chapter 16.5 – A task-scheduling problem as a matroid

Task-Scheduling Problem

- Input: a finite set $S = \{a_1, a_2, \dots, a_n\}$ of n ***unit-time tasks***, their corresponding integer deadlines $d_1, d_2, \dots, d_n$ $(1 \le d_i \le n)$, and nonnegative penalties $w_1, w_2, \dots, w_n$ if a_i is not finished by time d_i

Job	**1**	**2**	**3**	**4**	**5**	**6**
Deadline (d_i)	1	2	3	4	4	6
Penalty (w_i)	30	60	50	20	70	10

- Output: a schedule that minimizes the total penalty

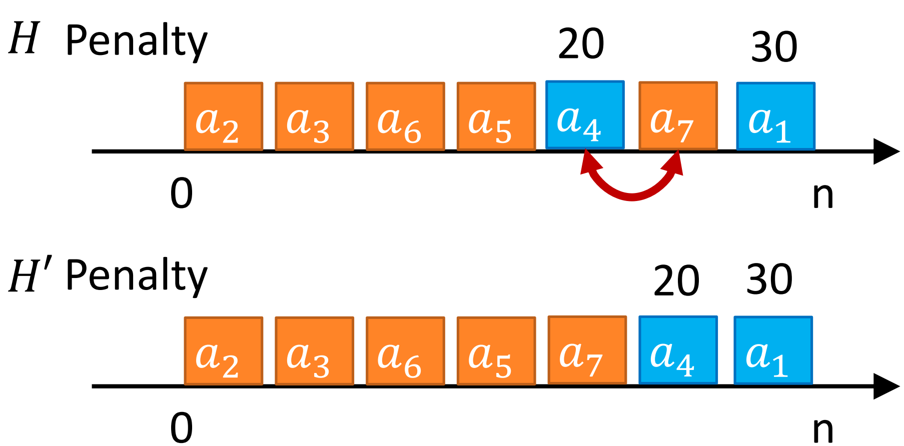


Task-Scheduling Problem

Task-Scheduling Problem
Input: n tasks with their deadlines $d_1, d_2, \dots, d_n$ and penalties $w_1, w_2, \dots, w_n$
Output: the schedule that minimizes the total penalty

Task	1	2	3	4	5	6	7
d_i	1	2	3	4	4	4	6
w_i	30	60	40	20	50	70	10

- Let a schedule H is the OPT
  - A task a_i is late in H if $f(H, i) > d_j$
  - A task a_i is early in H if $f(H, i) \le d_j$
  - We can have an **early-first** schedule H' with the same total penalty (OPT)



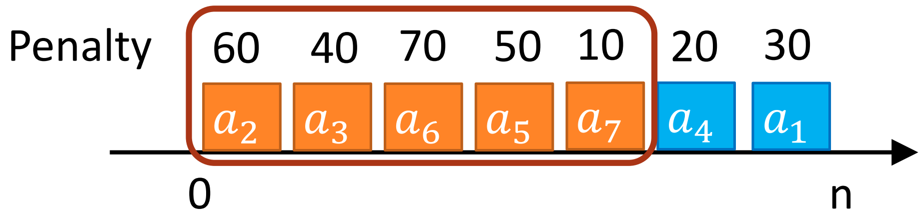
If the late task proceeds the early task, switching them makes the early one earlier and late one still late

Possible Greedy Choices

Task-Scheduling Problem
Input: n tasks with their deadlines $d_1, d_2, \dots, d_n$ and penalties $w_1, w_2, \dots, w_n$
Output: the schedule that minimizes the total penalty

- Rethink the problem: "maximize the total penalty for the set of early tasks"

Task	1	2	3	4	5	6	7
d_i	1	2	3	4	4	4	6
w_i	30	60	40	20	50	70	10

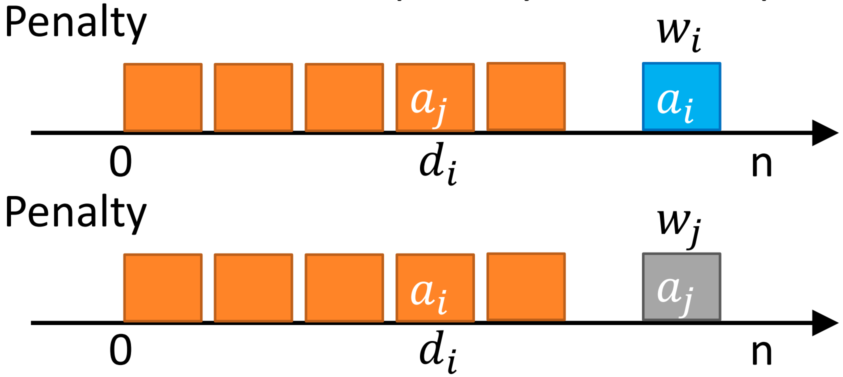


- Idea
  - Largest-penalty-first w/o idle time?
  - Earliest-deadline-first w/o idle time?

Prove Correctness

Task-Scheduling Problem
Input: n tasks with their deadlines $d_1, d_2, \dots, d_n$ and penalties $w_1, w_2, \dots, w_n$
Output: the schedule that minimizes the total penalty

- Greedy choice: select the **largest-penalty** task into the early set if feasible
- Proof via contradiction
  - Assume that there is no OPT including this greedy choice
    - If OPT processes a_i after d_i, we can switch a_j and a_i into OPT'
  - The maximum penalty must be equal or lower, because $w_i \geq w_j$



Prove Correctness

Task-Scheduling Problem
Input: n tasks with their deadlines $d_1, d_2, \dots, d_n$ and penalties $w_1, w_2, \dots, w_n$
Output: the schedule that minimizes the total penalty

- Greedy algorithm

```
Task-Scheduling(n, d[], w[])
  sort tasks by penalties s.t. w[1] ≥ w[2] ≥ … ≥ w[n]
  for i = 1 to n
    find the latest available index j <= d[i]
    if j > 0
      A = A ∪ {i}
      mark index j unavailable
return A // the set of early tasks
```

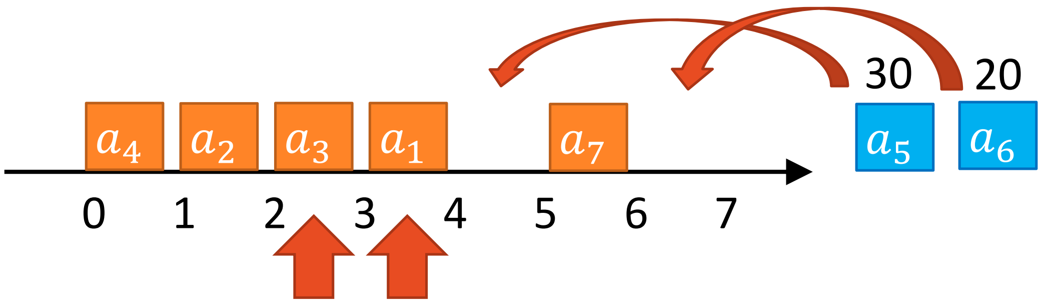
$$T(n) = O(n^2)$$

Can it be better?

Practice: reduce the time for finding the latest available index

Example Illustration

Job	1	2	3	4	5	6	7
Deadline (d_i)	4	2	4	3	1	4	6
Penalty (w_i)	70	60	50	40	30	20	10



Total penalty = 30 + 20 = 50

Concluding Remarks

- "Greedy": always makes the choice that looks best at the moment in the hope that this choice will lead to a globally optimal solution
- When to use greedy
  - Whether the problem has optimal substructure
  - Whether we can make a greedy choice and remain only one subproblem
  - Common for optimization problem

Optimal Solution = Greedy Choice + Subproblem Solution

- Prove for correctness
  - Optimal substructure
  - Greedy choice property

Question?

Important announcement will be sent to @ntu.edu.tw mailbox
& post to the course website

Course Website: http://ada17.csie.org

Email: ada-ta@csie.ntu.edu.tw