Advanced Core in Algorithm Design #3
算法設計要論 第3回

Yasushi Kawase
河瀬 康志

Oct. 19th, 2021

last update: 1:41pm, October 19, 2021

Schedule

Lec. #	Date	Topics
1	10/5	Introduction, Stable matching
2	10/12	Basics of Algorithm Analysis, Graphs
3	10/19	Greedy Algorithms (1/2)
4	10/26	Greedy Algorithms (2/2)
5	11/2	Divide and Conquer (1/2)
6	11/9	Divide and Conquer (2/2)
7	11/16	Dynamic Programming (1/2)
8	11/30	Dynamic Programming (2/2)
9	12/7	Network Flow (1/2)
10	12/14	Network Flow (2/2)
11	12/21	NP and Computational Intractability
12	1/4	Approximation Algorithms (1/2)
13	1/11	Approximation Algorithms (2/2)
14	1/18	Final Examination

Outline

1. Interval Scheduling
2. Interval Partitioning
3. Scheduling to Minimize Lateness

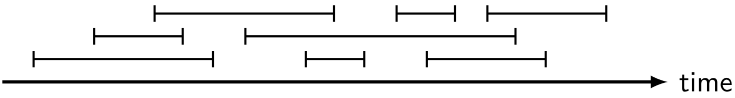
Interval Scheduling

Problem

- Input: jobs $J = \{1, 2, \ldots, n\}$, job j starts at $s(j)$ and finishes at $f(j)$
- Goal: find maximum subset of mutually compatible jobs

two jobs that don't overlap

Example



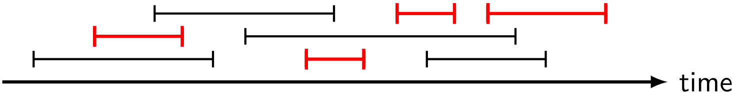
Interval Scheduling

Problem

- Input: jobs $J = \{1, 2, \ldots, n\}$, job j starts at $s(j)$ and finishes at $f(j)$
- Goal: find maximum subset of mutually compatible jobs

two jobs that don't overlap

Example

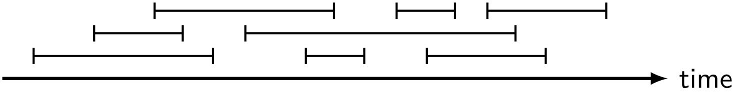


Algorithm

Greedy Algorithm

$R \leftarrow J,\ A \leftarrow \emptyset$;
while $R \neq \emptyset$ **do**
  Let $i \in \arg\min\{f(i) \mid i \in R\}$;
  $A \leftarrow A \cup \{i\}$;
  $R \leftarrow \{j \in R \mid s(j) > f(i)\}$;
Return A;

Example



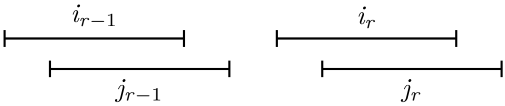
Optimality

Theorem

The greedy algorithm outputs an optimal solution

Proof

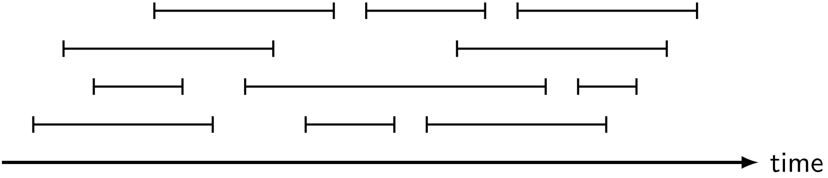
- $A = \{i_1, i_2, \ldots, i_k\}$: algorithm's output $(f(i_1) \leq \cdots \leq f(i_k))$
- $A^* = \{j_1, j_2, \ldots, j_m\}$: optimal solution $(f(j_1) \leq \cdots \leq f(j_m))$
- Claim: $f(i_r) \leq f(j_r)$ for all $r = 1, 2, \ldots, k$
    - Base case: $f(i_1) \leq f(j_1)$ by the definition
    - Induction step: $f(i_{r-1}) \leq f(j_{r-1}) \Rightarrow f(i_r) \leq f(j_r)$



- If $m > k$, the algorithm can choose j_{k+1} after i_k $\longrightarrow$ Contradiction

Quiz

What is the optimal value of the following interval scheduling?



Outline

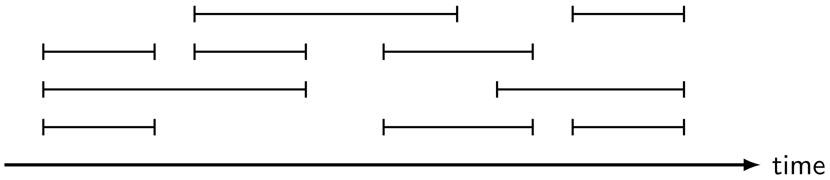
Interval Partitioning (Interval Coloring)

Problem

- Input: jobs $J = \{1, 2, \ldots, n\}$, job j starts at $s(j)$ and finishes at $f(j)$
- Goal: minimum number of people who can do all jobs

each person can do at most one job simultaneously

Example



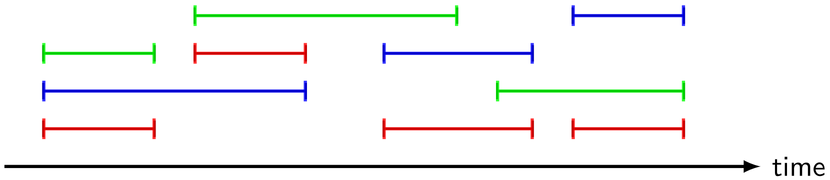
Interval Partitioning (Interval Coloring)

Problem

- Input: jobs $J = \{1, 2, \ldots, n\}$, job j starts at $s(j)$ and finishes at $f(j)$
- Goal: minimum number of people who can do all jobs

each person can do at most one job simultaneously

Example $(\text{optimal} = 3)$



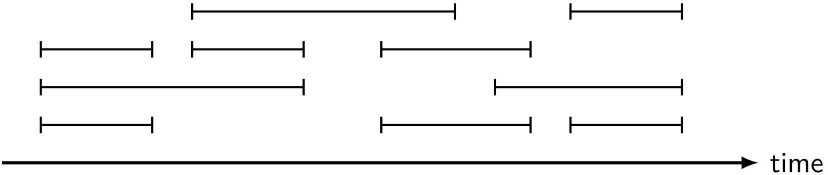
Basic observation

Observation

maximum number of pairwise overlapping intervals

the optimal value $\geq$ depth

Example (depth $= 3$)



Basic observation

Observation

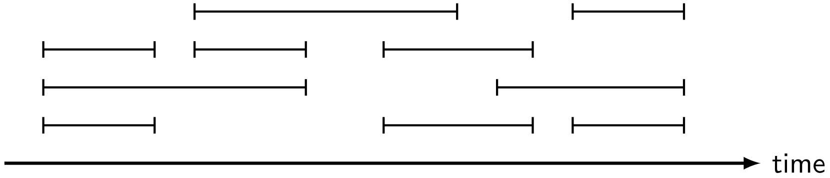
the optimal value $\geq$ depth

maximum number of pairwise overlapping intervals

Theorem

the optimal value $=$ depth

Example (depth $= 3$)

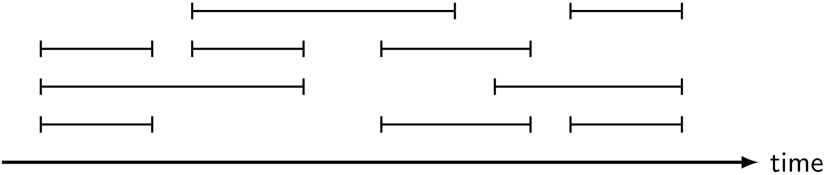


Algorithm

Greedy Algorithm

sort and relabel the jobs by their start times $(s(1) \leq \cdots \leq s(n))$;
let d be the depth and prepare d people;
for $j \leftarrow 1, 2, \ldots, n$ **do**
  assign j to any person who is free within time $(s(j), f(j))$;

Example



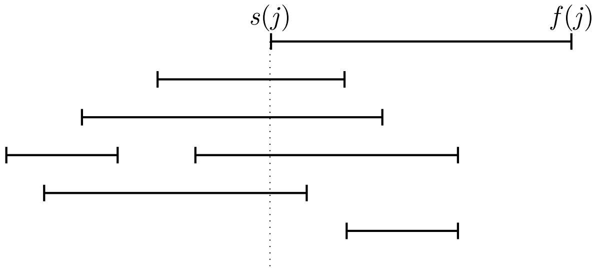
Correctness

Theorem

The greedy algorithm correctly assigns the jobs to d people

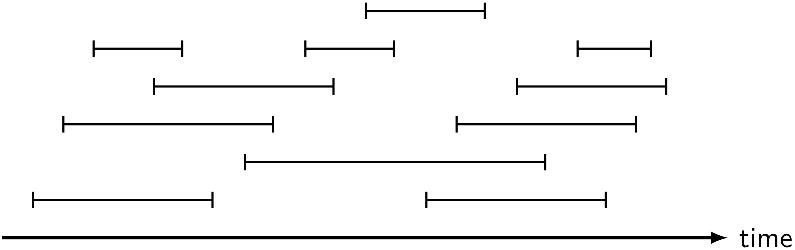
Proof: when the algorithm assigns job j, at least one person is free

→ the greedy algorithm is correct



Quiz

What is the optimal value of the following interval partitioning?



Outline

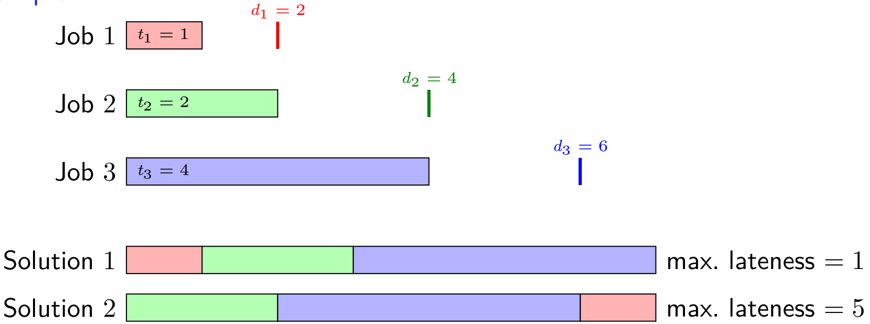
Scheduling to Minimize Lateness

Problem

- Input: n jobs $J = \{1, 2, \ldots, n\}$
  job j has deadline d_j and processing time t_j
- Goal: minimize the maximum lateness

$\max\{0, f_j - d_j\}$ where f_j is the finish time of j

Example

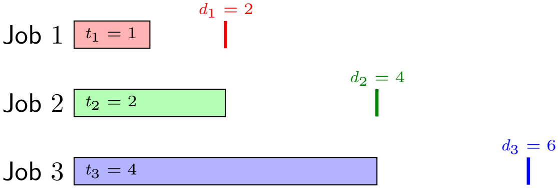


Algorithm

Greedy Algorithm (Earliest Deadline First)

sort and relabel the jobs by their deadlines ($d(1) \leq \cdots \leq d(n)$);
$f \leftarrow 0$;
for $j \leftarrow 1, 2, \ldots, n$ **do**
  assign j to the time interval $[f,\, f + t_j]$;
  $f \leftarrow f + t_j$;

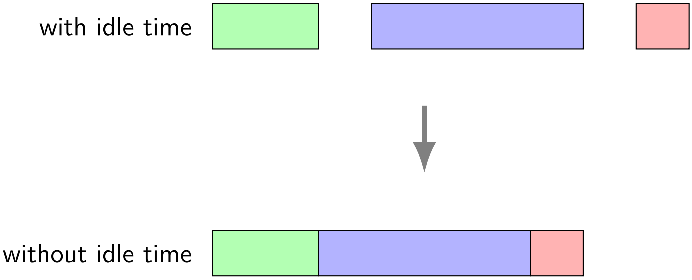
Example



Basic observation

Observation

There is an optimal schedule with no idle time



Inversion

Definition

$(i, j) \in J^2$ is inversion if (1) i is scheduled before j and (2) $d_i > d_j$

Observation

$\exists$ inversion $\Longrightarrow$ $\exists$ adjacent (consecutively scheduled) inversion

Suppose that (i, j) is inversion ($d_i > d_j$)

- if $d_k \leq d_j \Rightarrow (i, k)$ is inversion
- if $d_k > d_j \Rightarrow (k, j)$ is inversion

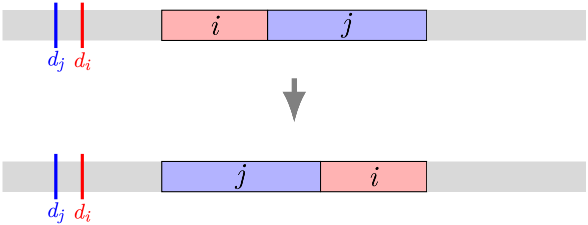
→ repeating this, we can find an adjacent inversion

Optimality

Proposition

Swapping an adjacent inversion does not increase maximum lateness

→ any schedule with no inversions and no idle time is optimal

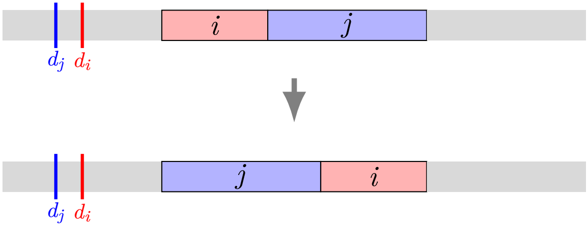


Optimality

Proposition

Swapping an adjacent inversion does not increase maximum lateness

→ any schedule with no inversions and no idle time is optimal



Theorem

The greedy algorithm outputs an optimal schedule

What is the minimum of the maximum lateness?