CMSC 451 9/16

Discrete Optimization

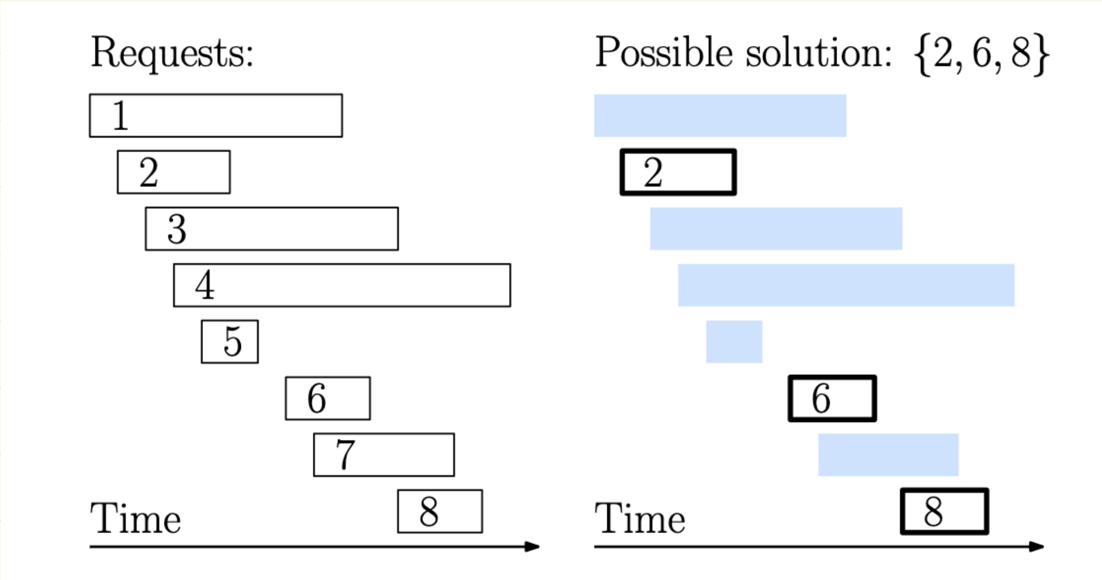
Compute a discrete structure of a given class to maximize/minimize a given objective function subject to a given set of constraints.

Greedy: Build a solution step by step.
each based on current best choice
never revoke

Interval Scheduling

Given a set $R = \{r_1, \dots, r_n\}$ of requests, each being an interval $r_i = [s_i, f_i]$.

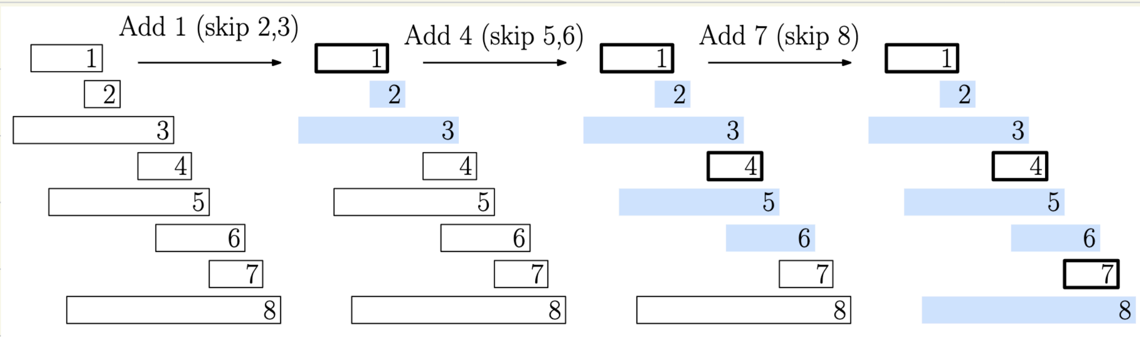
Compute a subset of non-overlapping requests of maximum cardinality



Algorithm: select a request that does not conflict with prior selection with some measure minimized

eg: Earliest Start
Earliest Finish
Shortest Duration
....

```
interval-scheduling (s, f)
    sort requests by f[i]
    S ← ∅
    prev_finish ← -∞
    for i = 1 to n
        if s[i] > prev_finish
            S ← S ∪ {i}
            prev_finish ← f[i]
```



Running time: $O(n \log n)$

Correctness:

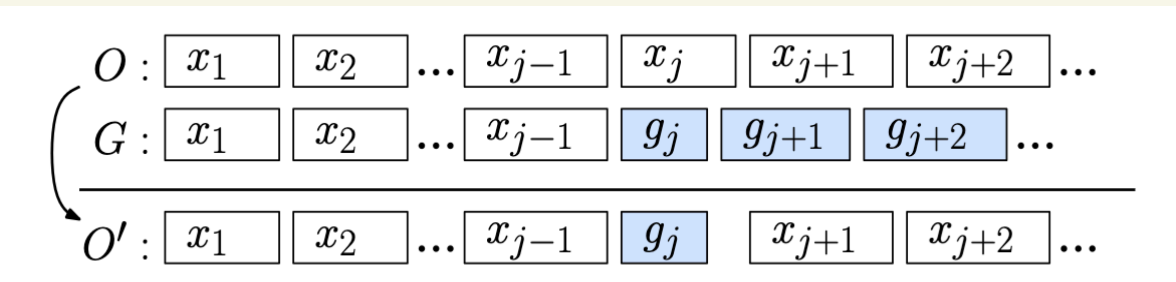
Feasibility: always give a non-conflicting schedule.
Easy to prove

Optimality: No. of requests is maximized.

Proof: Let $O = \langle x_1, \ldots, x_K \rangle$ be a optimal schedule.
$G = \langle g_1, \ldots, g_{K'} \rangle$ produced by the algorithm.

$G = O \Rightarrow$ good

$G \neq O \Rightarrow$ let j be the smallest index s.t. $x_j \neq g_j$.



O' is also optimal.

Repeat: let j' be the smallest index s.t. $x_{j'} \neq g_{j'}$

$j < j' < j'' < \cdots < j'''' \Rightarrow O'''' = G$

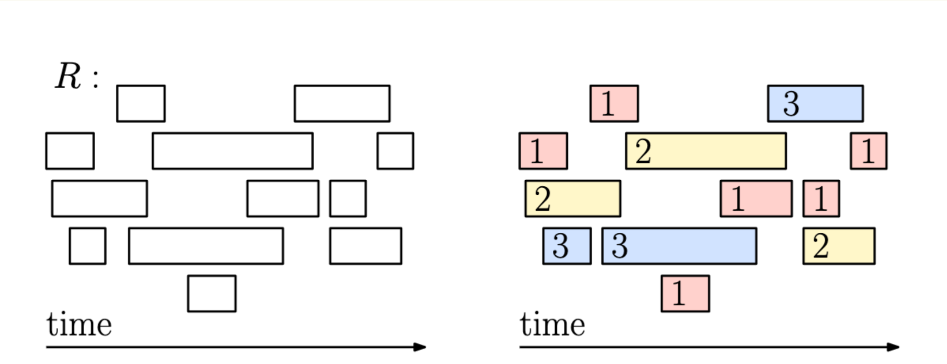
Interval Partitioning:

Given set $R = \{r_1, \dots, r_n\}$ of requests,

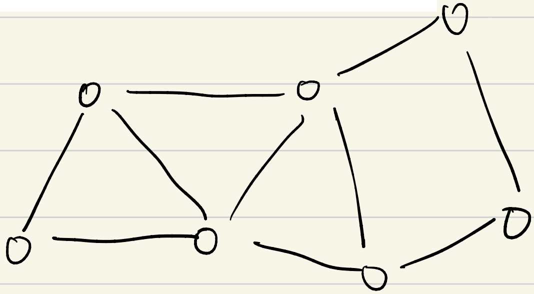
start / finish time $[s_i, f_i]$

partition R into subsets $R_1, \dots, R_d$,

s.t. each is non-conflicting. minimize d.



Coloring problem:



NP-Complete for general graphs.

Greedy Algorithm:

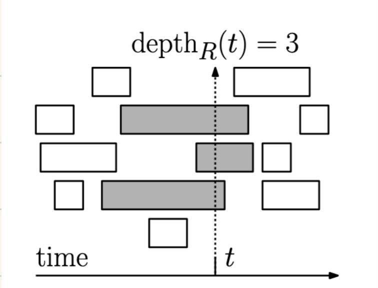
Try place each request (increasing order in S_i) onto the smallest available color.

```
Interval-partition (s, f)
    sort requests by s_i
    for i = 1 to n
        X ← ∅
        for j = 1 to i-1
            if [s_j, f_j] overlaps w/ [s_i, f_i]
                X ← X ∪ { color[j] }
        color[i] ← smallest color not in X.
```

Running time?

Correctness :

Feasibility : easy.



Optimality ?.

Depth : at some time t, # requests that occupy t.

Proof : 1. any solution has $d \geq$ depth.

2. greedy gives $d =$ depth.

Let i be the request that first violates $d =$ depth.

Then $\forall$ $c \in [1, \dots, \text{depth}]$,

$\exists$ j before i, s.t. color[j] $= c$,

j conflict with i.

{all j's & i} are depth + 1 requests that occupies time s_i

$\boxed{\text{Q : Why sort by } s_i \text{ ?}}$

Scheduling to Minimize Lateness.

Given set of tasks T, execution time t_i,
deadline d_i,

Goal: compute a schedule of tasks s.t.
not overlapping & minimize lateness.

Lateness: max deadline excess.

s_i: starting time. f_i = finish time, $f_i = s_i + t_i$

$$l_i = \max(0, f_i - d_i)$$

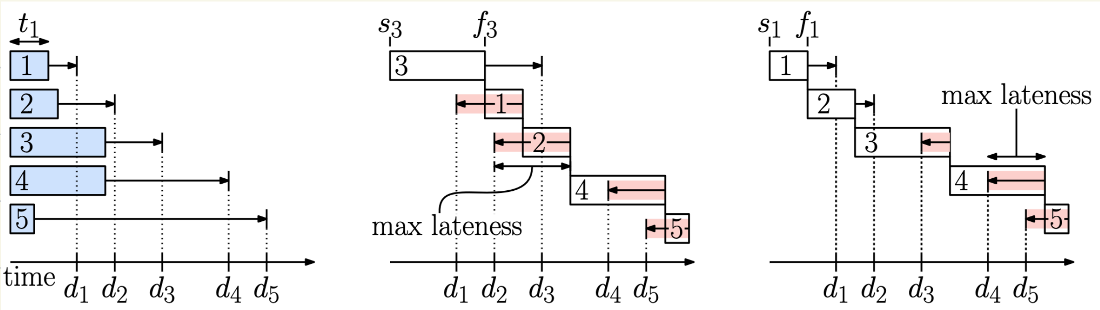
$$L(S) = \max_{i=1,\dots,n} l_i$$

Greedy Algorithm:

Smallest duration first?

Earliest deadline first.

Algorithm: sorting + a scan $O(n \log n)$



Correctness :

O be any optimal solution.

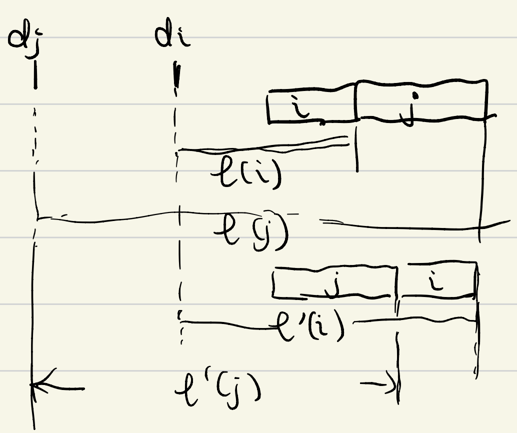
$\langle x_1, \dots, x_n \rangle$.

if $\forall i<j$, $d_i < d_j$, $\Rightarrow O = G$. good!

otherwise, let i, j, be smallest s.t.

$j = i+1$ but $d_i > d_j$

Claim: swap x_i & x_j gives better lateness



$\ell'(i) < \ell(j)$

$\ell'(j) < \ell(j)$

$\Rightarrow L(O') \leq L(O)$