Greedy Interval Scheduling

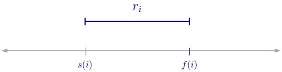
- Interval Scheduling - Generic Greedy Algorithm
- Sub-Optimal Greedy Algorithms
  - Earliest Starting Request First
  - Latest Finishing Request First
  - Shortest Duration Request First
  - Least Conflicting Request First
- Earliest Finish Time First Algorithm
  - Correctness and Optimality
  - Implementation and Runtime

Imdad ullah Khan

Interval Scheduling: Problem

- $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ (set of requests)
- Starting and finishing time of r_i: $s(i)$ and $f(i)$

  for $1 \leq i \leq n \quad s(i) < f(i)$



- Duration of request r_i is d_i is $f(i)$ - $s(i)$

r_i and r_j are **compatible** if they do not overlap in time

Otherwise r_i and r_j are **conflicting**

$$\underbrace{s(i) < f(i) \quad < \quad s(j) < f(j)}_{r_i \text{ is to the left of } r_j} \qquad \text{OR} \qquad \underbrace{s(j) < f(j) \quad < \quad s(i) < f(i)}_{r_i \text{ is to the right of } r_j}$$

A set is compatible if all pairs in it are compatible

Interval Scheduling: Generic Greedy Algorithm

Input: A set $\mathcal{R}$ of requests

Output: A largest compatible subset $S \subset \mathcal{R}$

Generic Greedy algorithm

Process requests in a fixed order and select compatible requests greedily

Algorithm Geedy Interval Scheduling Algorithm ($\mathcal{R}$)

$A \leftarrow \emptyset$
while $\mathcal{R} \neq \emptyset$ **do**
  select a request r_x from $\mathcal{R}$
  remove from $\mathcal{R}$ all those requests conflicting with r_x
  $A \leftarrow A \cup \{r_x\}$
return A

By construction the algorithm is correct ▷ (A is a compatible subset)

Interval Scheduling: Greedy Algorithm

- Earliest Starting Request First
- Latest Finishing Request First
- Shortest Duration Request First
- Least Conflicting Request First
  - Found a counter example to optimality of each of the above
- Earliest finishing request first

  ▷ Idea is to make resource free as soon as possible
- Optimal?
  - Produced optimal solution on all above examples
  - Need a proof of optimality

Earliest Finishing Time First: Algorithm

Algorithm : Interval Scheduling Algorithm ($\mathcal{R}$)

$A \leftarrow \emptyset$
while $\mathcal{R} \neq \emptyset$ **do**
  Select request r_x with smallest finishing time from $\mathcal{R}$
  Remove from $\mathcal{R}$ all requests conflicting with r_x
  $A \leftarrow A \cup \{r_x\}$
return A

Correctness: A is a feasible solution

- A is a valid output
- $A \subset \mathcal{R}$ and A is compatible
- At every step the algorithm only added r_i that was compatible with all of A so far

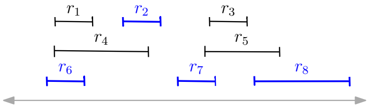
Earliest Finishing Time First: Optimality

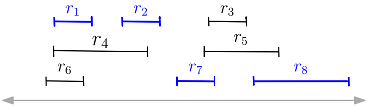
Optimality: A is an optimal solution

Let $\mathcal{O}$ be an optimal solution

We need to prove that $|A| = |O|$

Note that $A = O$ is not necessary





Both $\{r_1, r_2, r_7, r_8\}$ and $\{r_6, r_2, r_7, r_8\}$ are optimal solutions

Earliest Finishing Time First: Optimality

Optimality: A is an optimal solution

Let $\mathcal{O}$ be an optimal solution

We need to prove that $|A| = |O|$

Note that $A = O$ is not necessary

Let $A = a_1, a_2, \ldots, a_k$

Let $O = p_1, p_2, \ldots, p_m$

Need to prove that $|A| = k = m = |O|$

Earliest Finishing Time First: Optimality

Let $A = a_1, a_2, \ldots, a_k$ $\qquad O = p_1, p_2, \ldots, p_m$

Let A and O both be sorted by finishing time

Lemma: For $1 \leq i \leq k \quad f(a_i) \leq f(p_i)$

$\triangleright$ Note the range $1 \leq i \leq k$ cannot be $1 \leq i \leq m$

Proof uses the intuition (earliest finishing time first)

Our algorithm stays ahead by releasing the resource as early as possible

Earliest Finishing Time First: Optimality

Let $A = a_1, a_2, \ldots, a_k$ $\qquad O = p_1, p_2, \ldots, p_m$

Let A and O both be sorted by finishing time

Lemma: For $1 \leq i \leq k \quad f(a_i) \leq f(p_i)$

Proof by Induction on i

Basis Step: $f(a_1) \leq f(p_1)$ because $f(a_1) \leq f(*)$

Earliest Finishing Time First: Optimality

Let $A = a_1, a_2, \ldots, a_k$ $\qquad O = p_1, p_2, \ldots, p_m$

Let A and O both be sorted by finishing time

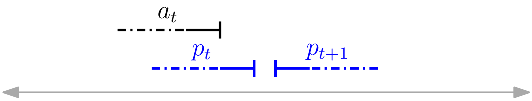
Lemma: For $1 \leq i \leq k$ $\quad f(a_i) \leq f(p_i)$

Inductive Step: If $f(a_t) \leq f(p_t)$, then $f(a_{t+1}) \leq f(p_{t+1})$

$f(a_t) \leq f(p_t) \leq s(p_{t+1})$ $\quad \because p_{t+1}$ is compatible with p_t

p_{t+1} is compatible with a_t, so p_{t+1} is available for A

Algorithm must choose a_{t+1} with $f(a_{t+1}) \leq f(p_{t+1})$



Earliest Finishing Time First: Optimality

Let $A = a_1, a_2, \ldots, a_k$ $\qquad O = p_1, p_2, \ldots, p_m$

Let A and O both be sorted by finishing time

Lemma: For $1 \leq i \leq k \quad f(a_i) \leq f(p_i)$

Theorem: $k = m$

Suppose $m > k, \quad \exists\, p_{k+1} \in O, \quad p_{k+1} \notin A$

$f(a_k) < f(p_k) \leq s(p_{k+1})$

p_{k+1} is compatible with all requests in A

p_{k+1} is available, $p_{k+1} \in \mathcal{R}$ after iteration k

Earliest Finishing Time First: Implementation

Algorithm : Interval Scheduling Algorithm ($\mathcal{R}$)

$A \leftarrow \emptyset$
while $\mathcal{R} \neq \emptyset$ **do**
  Select request r_x with smallest finishing time from $\mathcal{R}$
  Remove from $\mathcal{R}$ all requests conflicting with r_x
  $A \leftarrow A \cup \{r_x\}$
return A

Naive implementation: Uses list or array

- $O(n)$ for finding earliest finishing time request r_x ▷ FINDMIN by $f(\cdot)$
- $O(n)$ for deleting requests conflicting with r_x ▷ check all remaining requests
- r_i is conflicting with r_x if $s(r_i) \leq f(r_x)$ (in this setting)
- Total runtime : $O(n^2)$

Earliest Finishing Time First: Implementation

Algorithm : Interval Scheduling Algorithm ($\mathcal{R}$)

$A \leftarrow \emptyset$
while $\mathcal{R} \neq \emptyset$ **do**
  Select request r_x with smallest finishing time from $\mathcal{R}$
  Remove from $\mathcal{R}$ all requests conflicting with r_x
  $A \leftarrow A \cup \{r_x\}$
return A

Efficient implementation: Sort the $2d$-array $\mathcal{R}$ by $f(\cdot)$

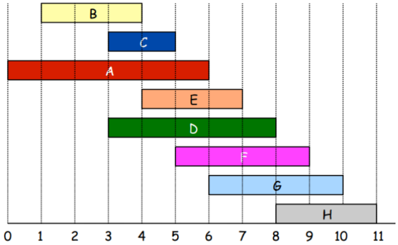
- Maintain index of first "not-deleted" request in $\mathcal{R}$
- Add $\mathcal{R}[index]$ to A
- Delete all requests $\mathcal{R}[index \dots n]$ whose $s(r_i) \leq f(r_{index})$

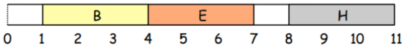
  $\triangleright$ Delete means move index to first index with $s(r_i) > f(r_{index})$

- Total runtime: $O(n)$

Algorithm : Interval Scheduling Algorithm ($\mathcal{R}$)

$A \leftarrow \emptyset$
while $\mathcal{R} \neq \emptyset$ **do**
  Select request r_x with smallest finishing time from $\mathcal{R}$
  Remove from $\mathcal{R}$ all requests conflicting with r_x
  $A \leftarrow A \cup \{r_x\}$
return A





source: https://stumash.github.io/Algorithm_Notes/greedy/intervals/intervals.html