Dynamic Programming

- Weighted Interval Scheduling
- Dynamic Programming Formulation

Imdad ullah Khan

Weighted Interval Scheduling: Dynamic Programming

Input: A set $\mathcal{R}$ of requests
Output: A compatible subset $S \subset \mathcal{R}$ of max total weight

Let $\mathcal{R}$ be sorted by finishing time

- OPT-SET(k): a max weight compatible subset of $\mathcal{R}[1 \ldots k]$
- OPT-VAL(k): the total weight of OPT-SET(k)

Out goal is to find OPT-SET(n) (and OPT-VAL(n))

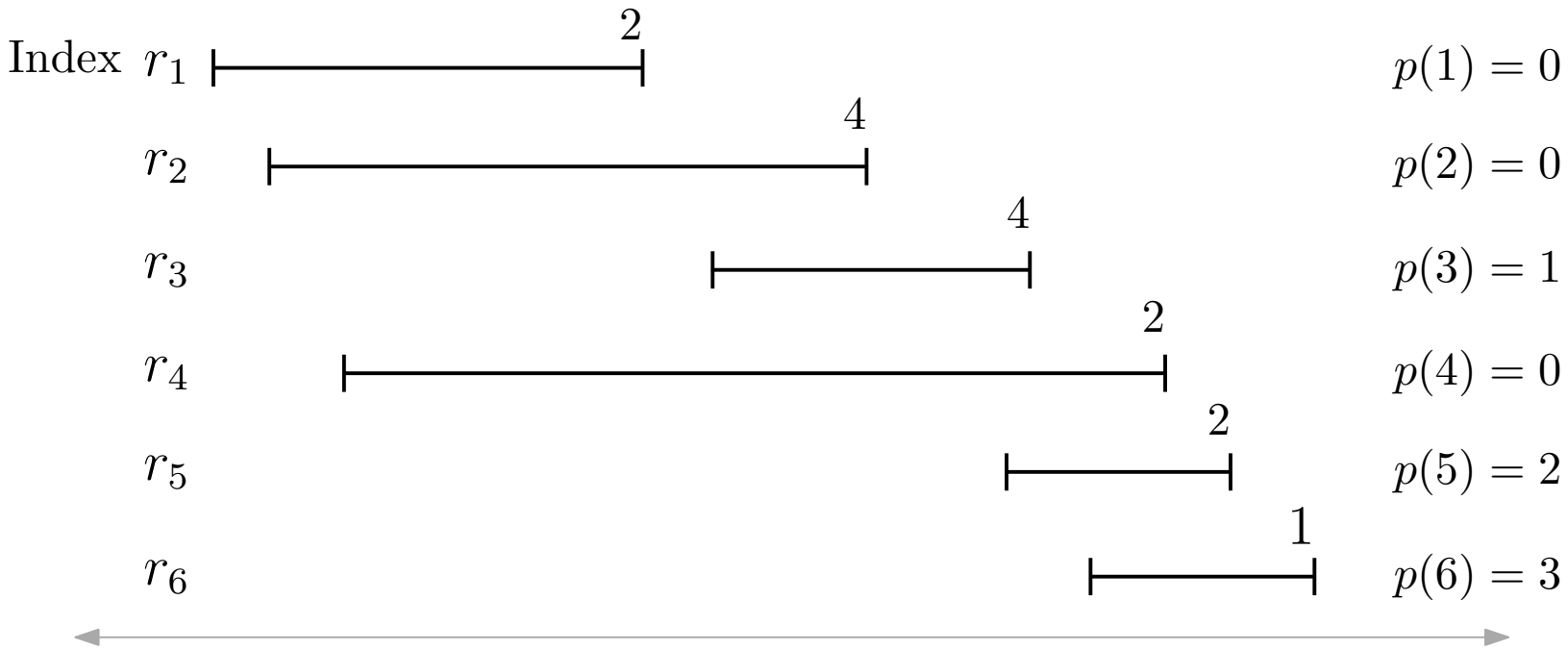
Weighted Interval Scheduling: Dynamic Programming

Input: A set $\mathcal{R}$ of requests
Output: A compatible subset $S \subset \mathcal{R}$ of max total weight

Let $\mathcal{R}$ be sorted by finishing time

Let $p(j)$ be the largest index $i < j$ with r_i compatible with r_j

▷ $p(j)$ is index of latest finishing interval before r_j starts



Weighted Interval Scheduling: Dynamic Programming

Argue about structure of the solution (though we can't compute it)

See how solution is composed of that to smaller subproblems

- Either r_n is part of the optimal solution, OPT-SET(n)
  - $v(n)$ is counted in OPT-VAL(n)
  - $r_{p(n)+1}, r_{p(n)+2}, \ldots, r_{n-1}$ are not in OPT-SET(n)
  - They finish before $f(n)$ and after $s(n)$
  - Analyze OPT-SET$(p(n))$
- Or r_n is not part of the optimal solution, OPT-SET(n)
  - $v(n)$ is not counted in OPT-VAL(n)
  - $r_{n-1}, r_{n-2}, \ldots$ may or may not be included
  - Analyze solution to $\mathcal{R} \setminus \{r_n\}$, i.e. OPT-SET$(n-1)$

Analyze OPT-SET$(n-1)$ and OPT-SET$(p(n))$?

Weighted Interval Scheduling: Dynamic Programming

Analyze OPT-SET$(n-1)$ and OPT-SET$(p(n))$?

- $r_n \notin$ OPT-SET(n)
  - OPT-SET(n) is a compatible subset in $\mathcal{R}[1 \ldots n-1]$
  - OPT-SET(n) is a max weight compatible subset in $\mathcal{R}[1 \ldots n-1]$
  - A bigger weight compatible subset in $\mathcal{R}[1 \ldots n-1]$ is also good for $\mathcal{R}$
- Or $r_n \in \mathit{OptSet}(n)$
  - $r_{p(n)+1}, r_{p(n)+2}, \ldots, r_{n-1}$ are not in OPT-SET(n)
  - OPT-SET$(n) \setminus \{r_n\}$ is a compatible subset in $\mathcal{R}[1 \ldots p(n)]$
  - OPT-SET$(n) \setminus \{r_n\}$ is a max weight compatible subset in $\mathcal{R}[1 \ldots p(n)]$
  - If $O \subset \mathcal{R}[1 \ldots p(n)]$ is compatible with weight $\geq$ OPT-VAL$(n) - v(r_n)$
  - Then $O \cup \{r_n\}$ is compatible in $\mathcal{R}$ with value bigger than OPT-VAL(n)

Weighted Interval Scheduling: Dynamic Programming

Analyze $\text{OPT-SET}(n-1)$ and $\text{OPT-SET}(p(n))$?

A max weight compatible set in $\mathcal{R}$ is either

- a max weight compatible set in $\mathcal{R}[1 \ldots n-1]$ or it is
- r_n union with a max weight compatible subset in $\mathcal{R}[1 \ldots p(n)]$

$$\text{OPT-VAL}(n) = max \begin{cases} \text{OPT-VAL}(p(n)) + v(n) & \text{if } r_n \in \text{OPT-SET}(n) \\ \text{OPT-VAL}(n-1) & \text{if } r_n \notin \text{OPT-SET}(n) \end{cases}$$

In general

$$\text{OPT-VAL}(k) = \max \begin{cases} \text{OPT-VAL}(p(k)) + v(k) & \text{if } r_k \in \text{OPT-SET}(k) \\ \text{OPT-VAL}(k-1) & \text{if } r_k \notin \text{OPT-SET}(k) \end{cases}$$

Weighted Interval Scheduling: Dynamic Programming

Recursion ?

- If $r_k \in \text{OPT-SET}(k)$, then find max compatible set in $\mathcal{R}[1 \ldots p(k)]$
- If $r_k \notin \text{OPT-SET}(k)$, then find max compatible set in $\mathcal{R}[1 \ldots k-1]$

We don't know the bases cases and which branch to take

$$\text{OPT-VAL}(k) = \max \begin{cases} 0 & \text{if } k = 0 \\ \text{OPT-VAL}(p(k)) + v(k) & \text{if } v_k \in \text{OPT-SET}(k) \\ \text{OPT-VAL}(k-1) & \text{if } v_k \notin \text{OPT-SET}(k) \end{cases}$$

Try both branches and select the one that is bigger?

Weighted Interval Scheduling: Dynamic Programming

$$\text{OPT-VAL}(k) = \max \begin{cases} 0 & \text{if } k = 0 \\ \text{OPT-VAL}(p(k)) + v(k) & \text{if } v_k \in \text{OPT-SET}(k) \\ \text{OPT-VAL}(k-1) & \text{if } v_k \notin \text{OPT-SET}(k) \end{cases}$$

Algorithm Recursive $\text{OPT-VAL}(k)$ Computation

$\text{SORT-BY-FIN-TIME}(\mathcal{R})$

For each r_i find $p(i)$ ▷ One binary search for each

function $\text{OPT-VAL}(k)$

  if $k = 0$ **then**

    return 0

  else

    return $\max\{\text{OPT-VAL}(k-1), \text{OPT-VAL}(p(k)) + v(k)\}$

Weighted Interval Scheduling: Dynamic Programming

- Implementing the above recurrence relation
- Proof of correctness by induction using the above argument
- Only compute OPT-VAL(n), can be extended to get OPT-SET(n)
- Need book keeping and backtracking
- Exponential runtime due to unnecessary repeated calls
- Pictures it's recursion tree and identify repetitions
- Use Memoization
- Use Bottom-Up Approach