Greedy Interval Coloring

- Interval Coloring - Degree, Depth, Lower Bound
- Greedy Interval Coloring
- Interval Coloring with Unknown Depth

Imdad ullah Khan

Interval Coloring: Introduction

- You have ~~a single~~ multiple mono-task resources

  ▷ e.g. lecture rooms or a research equipment

- and multiple requests to use a resource

  Each request specifies a start time and finish time

- ~~Problem is to schedule (accept/reject) the requests~~
- Problem is to map requests to resources
- All requests mapped to one resource must be compatible
- ~~The goal is to accept the maximum number of requests~~
- The goal is to use minimum number of resources

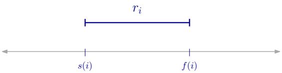
Each resource or part is referred to as a color

Also called interval partitioning

Interval Coloring: Problem Formulation

- $\mathcal{R} = \{r_1, r_2, \ldots, r_n\}$ (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**

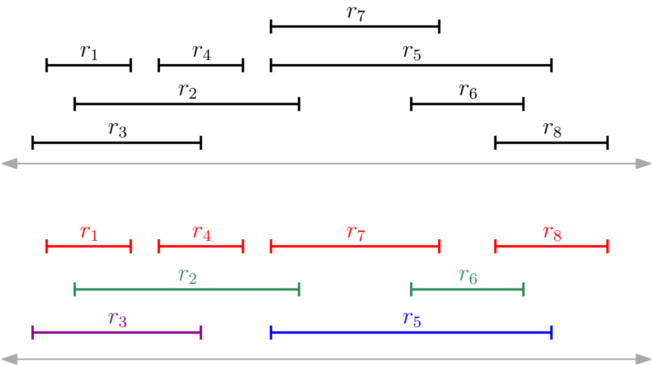
$$\underbrace{s(i) < f(i) \quad < \quad s(j) < f(j)}_{r_i \text{ is to the left of } r_j} \qquad \text{OR} \qquad \underbrace{s(j) < f(j) \quad < \quad s(i) < f(i)}_{r_i \text{ is to the right of } r_j}$$

A set is compatible if all pairs in it are compatible

Interval Coloring: Problem Formulation

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

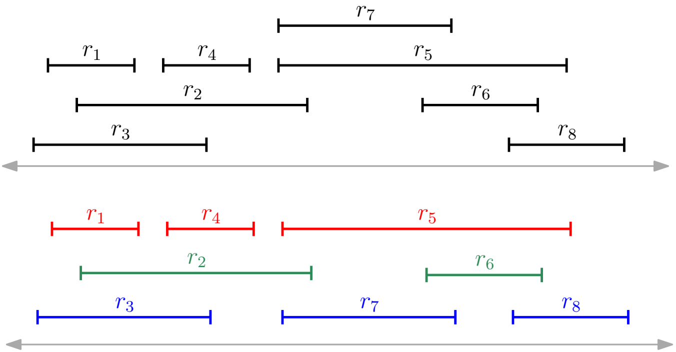
Output: A partition of $\mathcal{R}$ with smallest number of compatible subsets



Interval Coloring: Problem Formulation

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

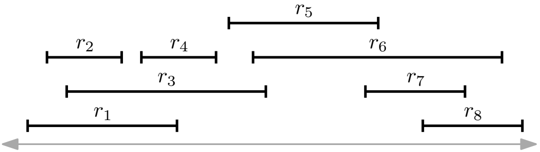
Output: A partition of $\mathcal{R}$ with smallest number of compatible subsets



Interval Coloring: Lower Bound

Degree of an interval is the number of other intervals conflicting with it

Depth of a set of intervals is the largest number of intervals passing through a point in time



$depth(\mathcal{R}) = 3$

$deg(r_1) = 3$
$deg(r_2) = 2$
$deg(r_3) = 5$
$deg(r_4) = 2$
$deg(r_5) = 3$

- Intervals passing through the depth realizing point(s) are *'parallel'*
- Number of resources (parts/colors) is at least the depth of $\mathcal{R}$

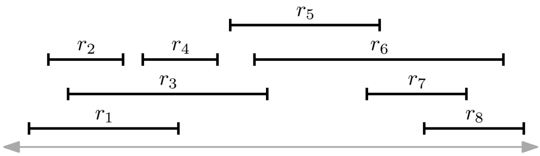
$D = depth(\mathcal{R})$ is a lower bound on the number of colors needed

Greedy Interval Coloring

We give an algorithm to color $\mathcal{R}$ with D colors

- Let $r_1, r_2, \ldots, r_n$ be intervals sorted by $s(\cdot)$
- $\mathcal{R}_p(r_j)$: set of intervals with $s(i) \leq s(j)$ that are conflicting with r_j

  $\triangleright$ $\mathcal{R}_p(r_j)$ is set of preceding intervals that conflict with r_j



$\mathcal{R}_p(r_1) = \{\ \}$

$\mathcal{R}_p(r_2) = \{r_1\}$

$\mathcal{R}_p(r_3) = \{r_1, r_2\}$

$\mathcal{R}_p(r_4) = \{r_1, r_3\}$

$\mathcal{R}_p(r_5) = \{r_3\}$

Greedy Interval Coloring

Algorithm : Interval Coloring Algorithm $(\mathcal{R}, D)$

$C \leftarrow \{c_1, \ldots, c_D\}$ $\triangleright$ set of D colors to assign

Let $r_1, r_2, \ldots, r_n$ be $\mathcal{R}$ sorted by $s(\cdot)$

for $j = 1$ to n **do**

  $C' \leftarrow C \setminus \{$colors used for any $r_i \in \mathcal{R}_p(r_j)\}$

  if $C' \neq \emptyset$ **then**

    color r_j with a $c_l \in C'$

  else

    Leave r_j uncolored

Greedy Interval Coloring: Correctness

Algorithm : Interval Coloring Algorithm $(\mathcal{R}, D)$

$C \leftarrow \{c_1, \ldots, c_D\}$ ▷ set of D colors to assign
Let $r_1, r_2, \ldots, r_n$ be $\mathcal{R}$ sorted by $s(.)$
for $j = 1$ to n **do**
  $C' \leftarrow C \setminus \{\text{colors used for any } r_i \in \mathcal{R}_p(r_j)\}$
  if $C' \neq \emptyset$ **then**
    color r_j with a $c_l \in C'$
  else
    Leave r_j uncolored

Every request in $\mathcal{R}$ gets a non-conflicting color if any

- Let r_a and r_b be conflicting requests with $s(r_a) < s(r_b)$
- while coloring r_b color of r_a was excluded by construction

Greedy Interval Coloring: Correctness

Algorithm : Interval Coloring Algorithm $(\mathcal{R}, D)$

$C \leftarrow \{c_1, \ldots, c_D\}$ ▷ set of D colors to assign
Let $r_1, r_2, \ldots, r_n$ be $\mathcal{R}$ sorted by $s(.)$
for $j = 1$ to n **do**
  $C' \leftarrow C \setminus \{\text{colors used for any } r_i \in \mathcal{R}_p(r_j)\}$
  if $C' \neq \emptyset$ **then**
    color r_j with a $c_l \in C'$
  else
    Leave r_j uncolored

Every request in $\mathcal{R}$ does get a color

- Suppose r_b did not get a color: for r_b, $C' = \emptyset$ ▷ $|\mathcal{R}_p(r_b)| \geq D$
- For every $r_j \in \mathcal{R}_p(r_b)$, $s(r_j) < s(r_b) < f(r_j)$ ▷ (conflict definition)
- At point $s(r_b)$, all requests in $\{r_b\} \cup \mathcal{R}_p(r_b)$ are active
- Hence depth would be greater than D ▷ a contradiction!

Greedy Interval Coloring: Optimality

Lower bound: Any algorithm must use at least D colors

Upper bound: This algorithm uses at most D colors

Greedy Interval Coloring: Implementation

Algorithm : Interval Coloring Algorithm $(\mathcal{R}, D)$

$C \leftarrow \{c_1, \dots, c_D\}$
Let $r_1, r_2, \dots, r_n$ be $\mathcal{R}$ sorted by $s(.)$
for $j = 1$ to n **do**
  $C' \leftarrow C \setminus \{$colors used for any $r_i \in \mathcal{R}_p(r_j)\}$
  color r_j with a $c_l \in C'$

Naive implementation 1 : Pre-compute $\mathcal{R}_p(r_j)$ for every j

- Takes $O(n^2)$ in precomputation
- Large space complexity

Greedy Interval Coloring: Implementation

Algorithm : Interval Coloring Algorithm $(\mathcal{R}, D)$

$C \leftarrow \{c_1, \ldots, c_D\}$
Let $r_1, r_2, \ldots, r_n$ be $\mathcal{R}$ sorted by $s(\cdot)$
for $j = 1$ to n **do**
  $C' \leftarrow C \setminus \{$colors used for any $r_i \in \mathcal{R}_p(r_j)\}$
  color r_j with a $c_l \in C'$

Naive implementation 2 : Find $\mathcal{R}_p(r_j)$ on the go

- No need to maintain or get the set
- Just need to find an unused color
- runtime: $O(n^2)$

Greedy Interval Coloring: Implementation

Algorithm : Interval Coloring Algorithm $(\mathcal{R}, D)$

$C \leftarrow \{c_1, \ldots, c_D\}$
Let $r_1, r_2, \ldots, r_n$ be $\mathcal{R}$ sorted by $s(\cdot)$
for $j = 1$ to n **do**
  $C' \leftarrow C \setminus \{\text{colors used for any } r_i \in \mathcal{R}_p(r_j)\}$
  color r_j with a $c_l \in C'$

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

- Searching for $\mathcal{R}_p(r_j)$ needs a scan of $\mathcal{R}$, leading to $O(n^2)$ runtime
- We only need to find an "*available*" color
- Maintain information of the last interval colored by each color
- Easy to update this information when r_j is colored with c_l
- An available color is the one whose last usage is not conflicting
- Total runtime: $O(nD)$, which could be $O(n^2)$

Greedy Interval Coloring: Implementation

Algorithm : Interval Coloring Algorithm $(\mathcal{R}, D)$

$C \leftarrow \{c_1, \ldots, c_D\}$
Let $r_1, r_2, \ldots, r_n$ be $\mathcal{R}$ sorted by $s(\cdot)$
for $j = 1$ to n **do**
  $C' \leftarrow C \setminus \{$colors used for any $r_i \in \mathcal{R}_p(r_j)\}$
  color r_j with a $c_l \in C'$

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

- Maintain information of the last interval colored by each color
- Maintain this information in a min-heap $\mathcal{H}$
- items are colors and values are $f(\cdot)$ of last request using this color
- For r_j, $c \leftarrow \textsc{ExtractMin}(\mathcal{H})$
- $\textsc{IncreaseKey}(\mathcal{H}, c, f(r_j))$
- Runtime $O(n \log n + n \log D)$

Interval Coloring: Unknown Depth

Algorithm : Interval Coloring Unknown Depth ($\mathcal{R}$)

$d \leftarrow 1$
Let $r_1, r_2, \ldots, r_n$ be $\mathcal{R}$ sorted by $s(\cdot)$
for $j = 1$ to n **do**
  if r_j can be colored by some color $c \leq d$ **then**
    Color r_i with color c
  else
    Allocate a new color $d + 1$
    $d \leftarrow d + 1$
    Color r_i with color d
return d

- Colored are named by numbers
- Allocates colors on need basis
- Prove that exactly D colors are allocated
- Proof of correctness is very similar to the known D case