Coloring Mixed and Directional Interval Graphs

GD 2022, Tokyo

Grzegorz Gutowski — Uniwersytet Jagielloński Kraków

Florian Mittelstädt, Ignaz Rutter, Joachim Spoerhase — Julius-Maximilians-Universität Würzburg

Alexander Wolff, **Johannes Zink** — Universität Passau

Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Input: directed graph G **Output:** layered drawing of G

Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Input: directed graph G **Output:** layered drawing of G

Consists of five phases:

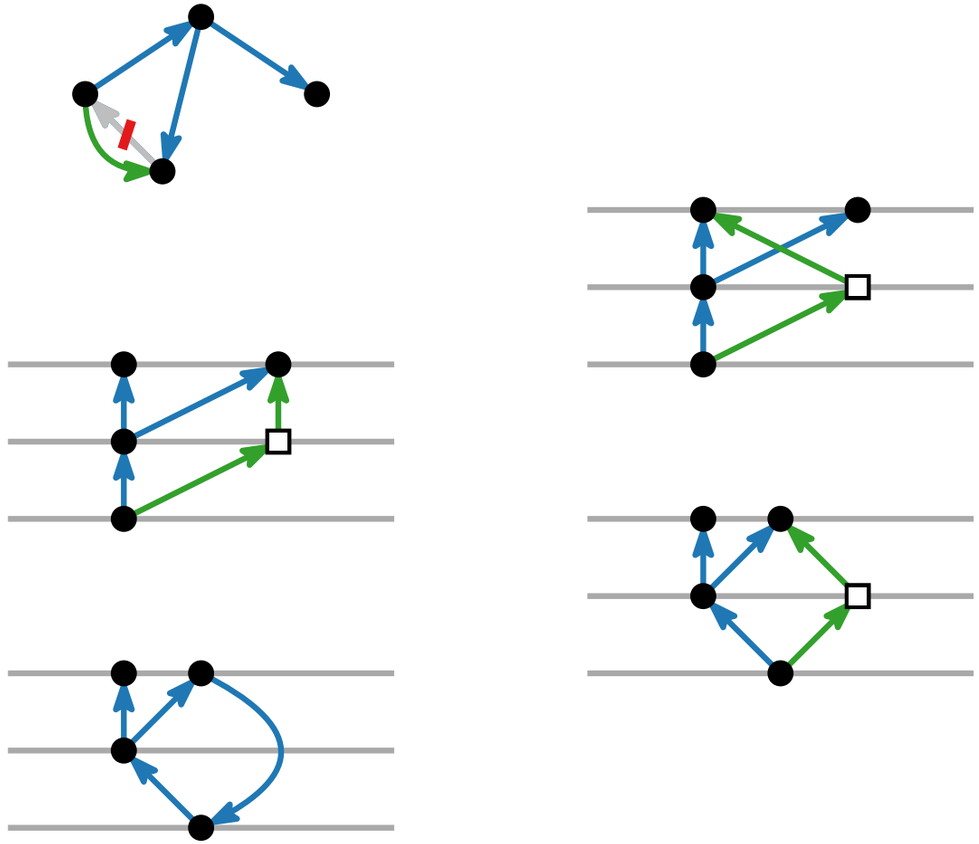
Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

Input: directed graph G **Output:** layered drawing of G

Consists of five phases:

1. cycle elimination
2. layer assignment
3. crossing minimization
4. node placement
5. edge routing



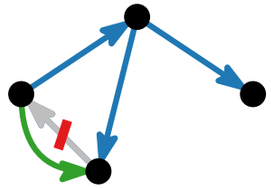
Motivation

Framework for layered graph drawing by Sugiyama, Tagawa, and Toda (1981).

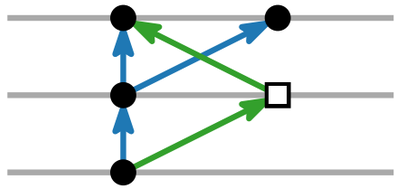
Input: directed graph G **Output:** layered drawing of G

Consists of five phases:

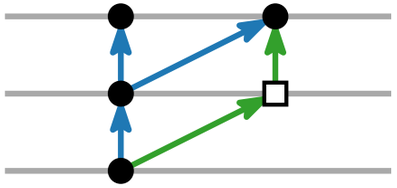
1. cycle elimination



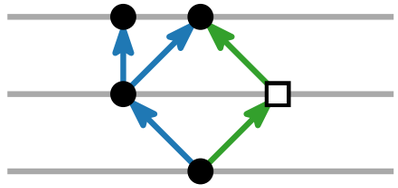
2. layer assignment



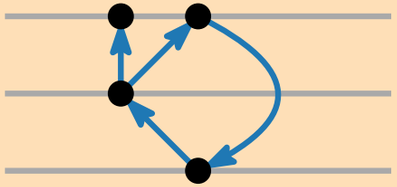
3. crossing minimization



4. node placement



5. edge routing



we want orthogonal edges!

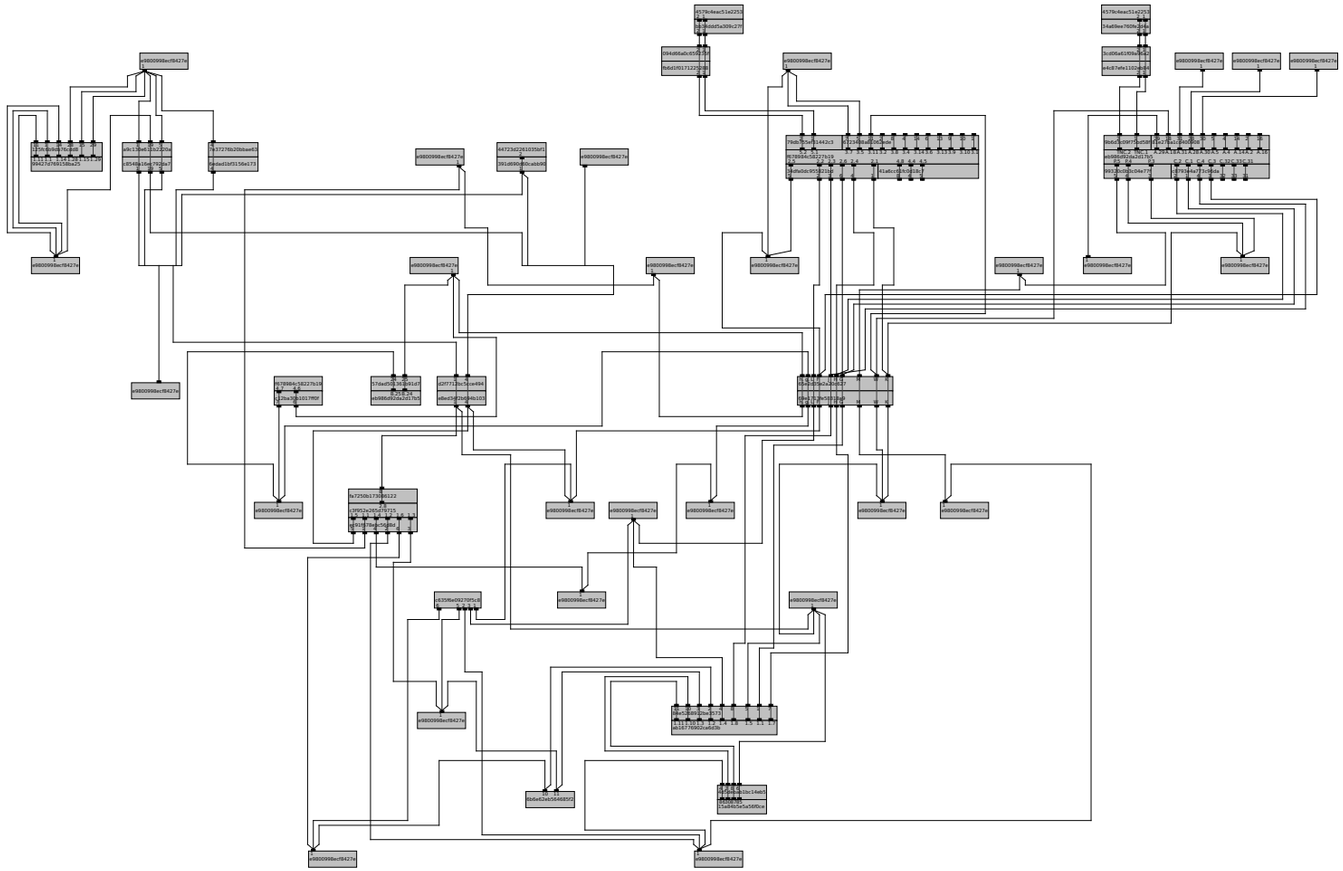
Motivation

Framework for layered g

Input: directed graph G

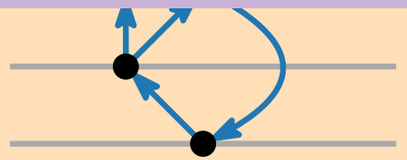
Consists of five phases:

1. cycle elimination
2. layer assignment
3. crossing minimiza
4. node placement
5. edge routing



cable plan

[Zink, Walter, Baumeister, Wolff; CGTA'22]



we want orthogonal edges!

Motivation – Layered Orthogonal Edge Routing

- it suffices to consider each pair of consecutive layers individually

Motivation – Layered Orthogonal Edge Routing

- it suffices to consider each pair of consecutive layers individually

Motivation – Layered Orthogonal Edge Routing

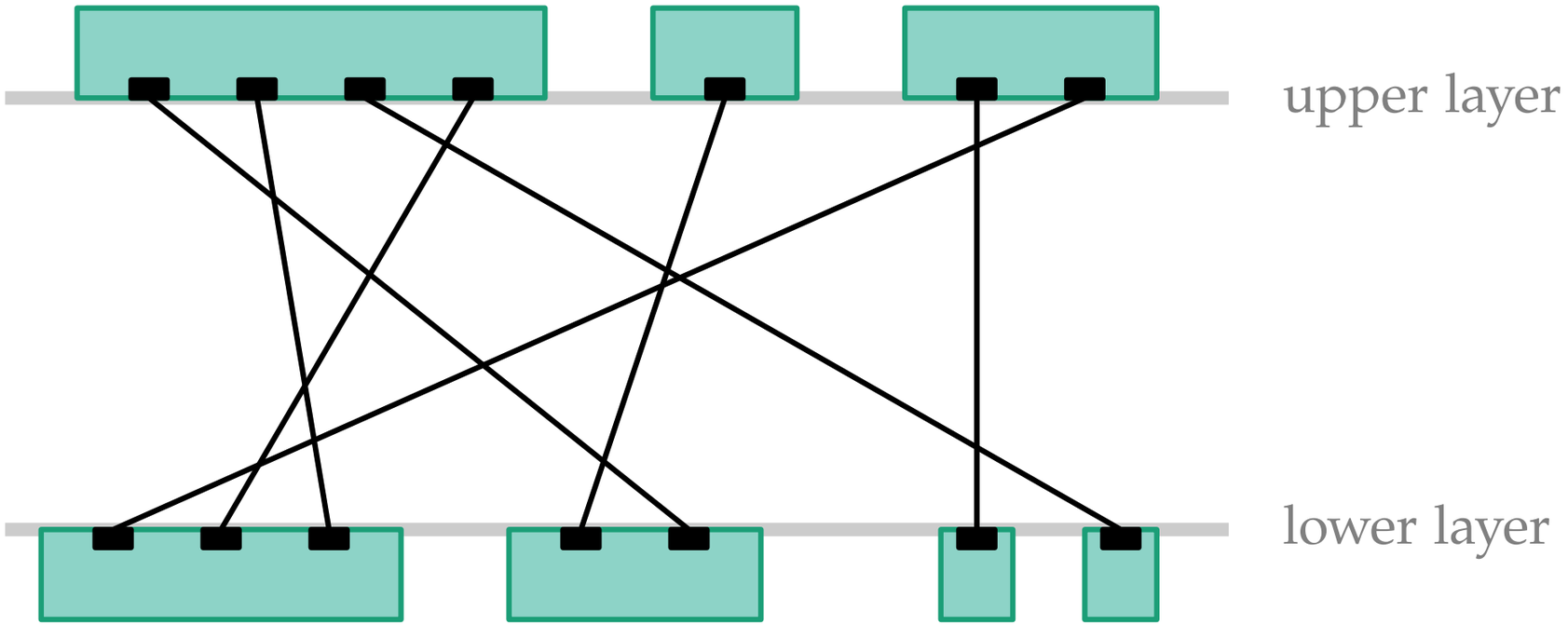
- it suffices to consider each pair of consecutive layers individually
- positions of vertices are fixed

upper layer

lower layer

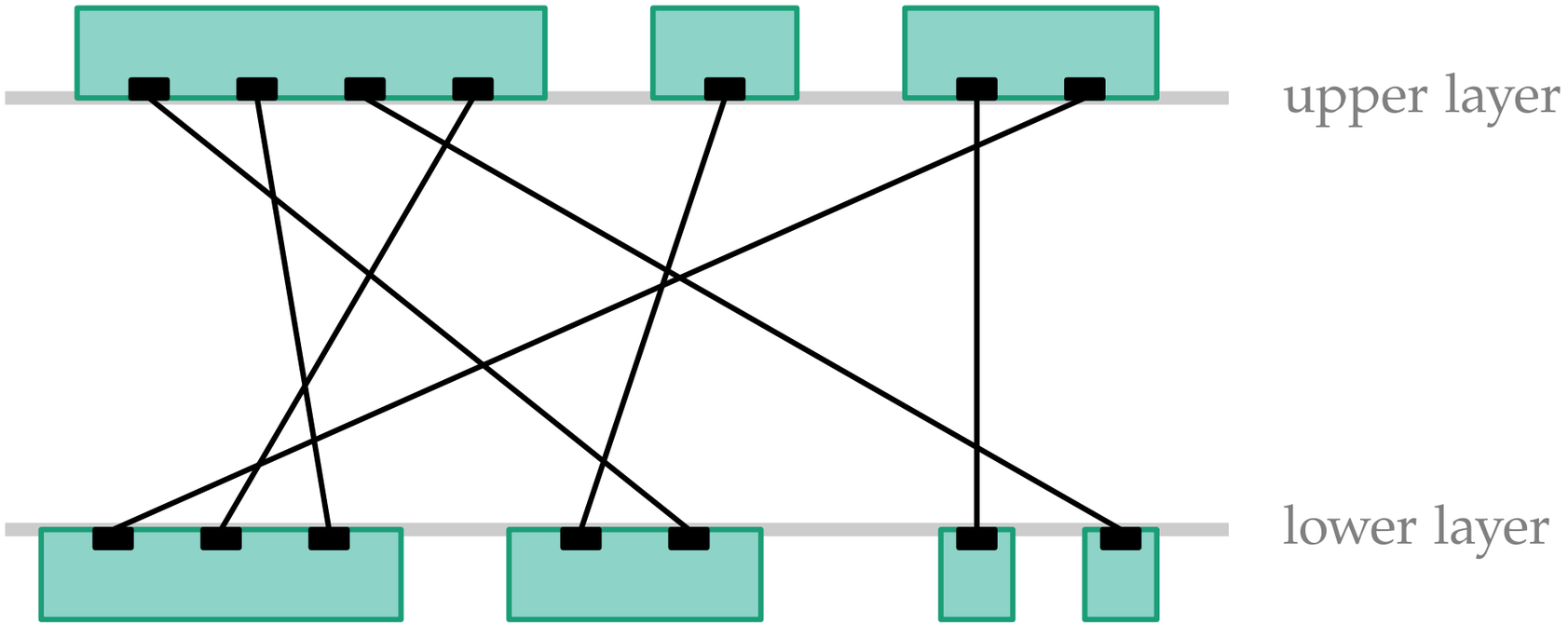
Motivation – Layered Orthogonal Edge Routing

- it suffices to consider each pair of consecutive layers individually
- positions of vertices are fixed
- no two edges share a common end point (vertices have distinct ports)



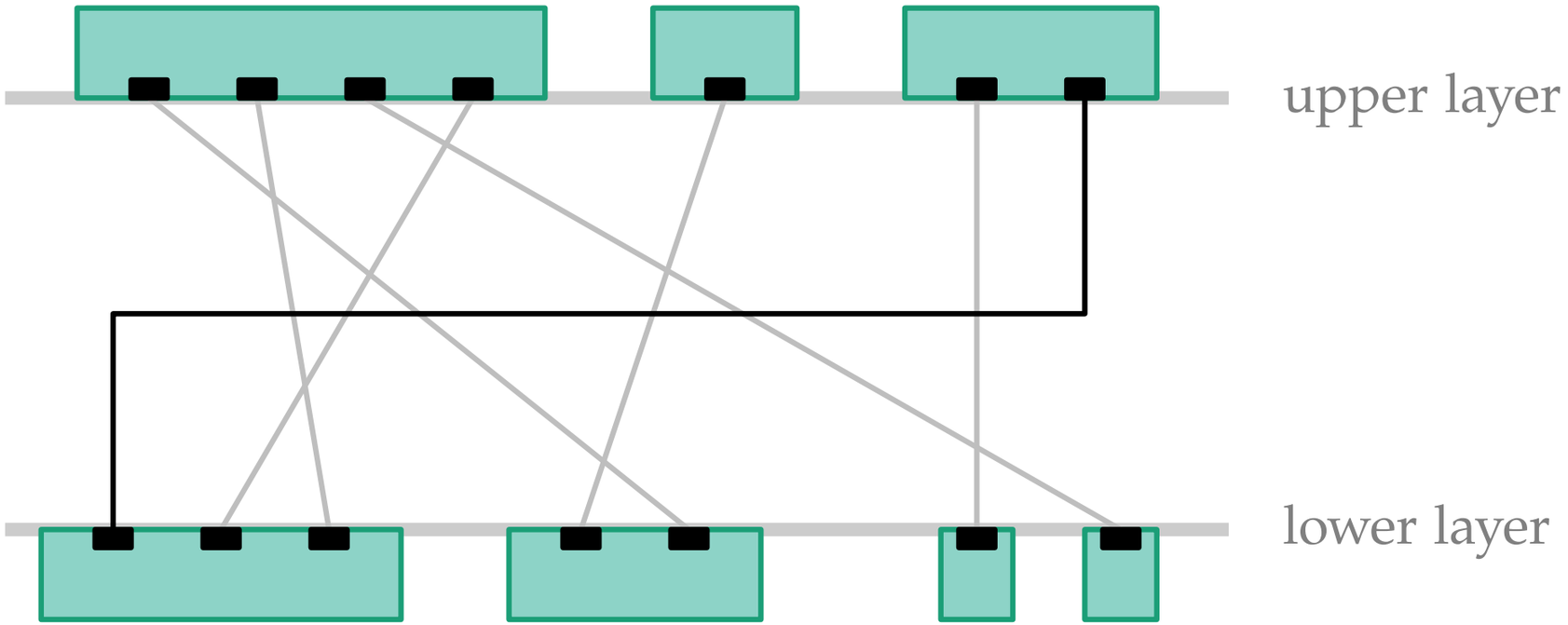
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments



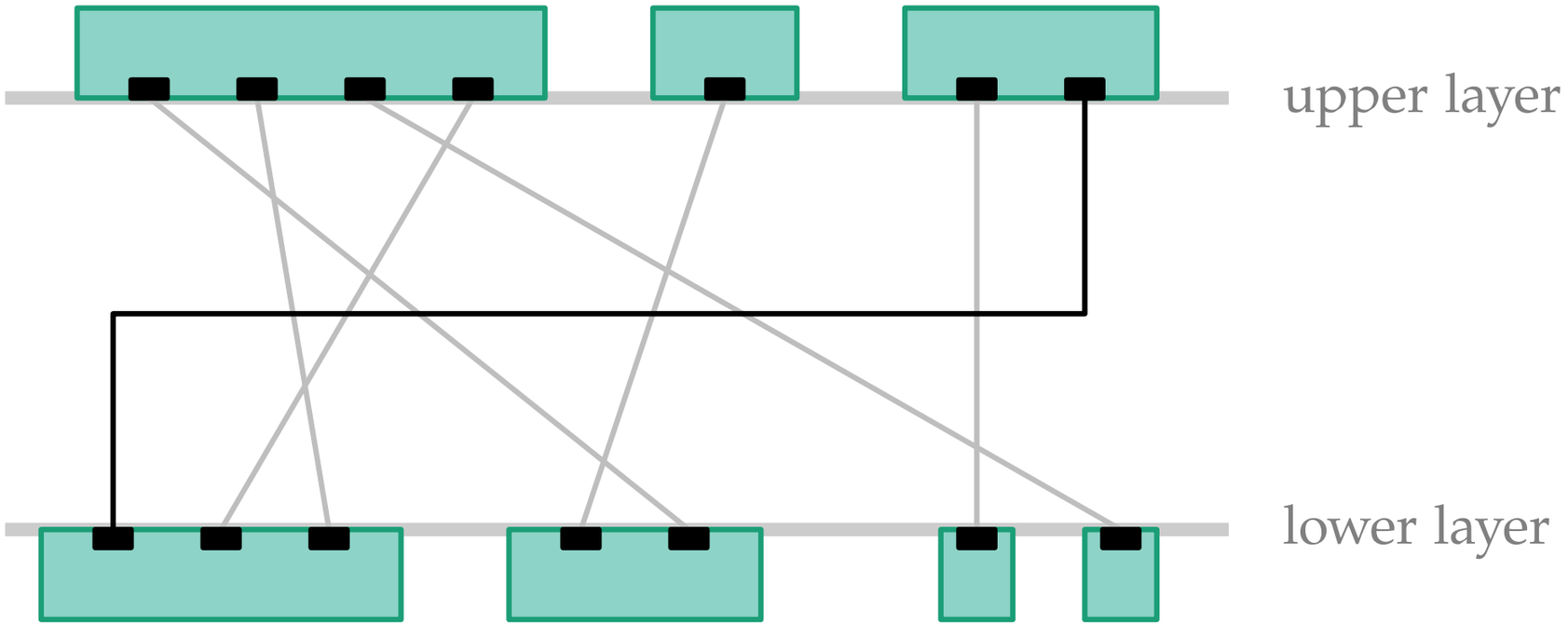
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments



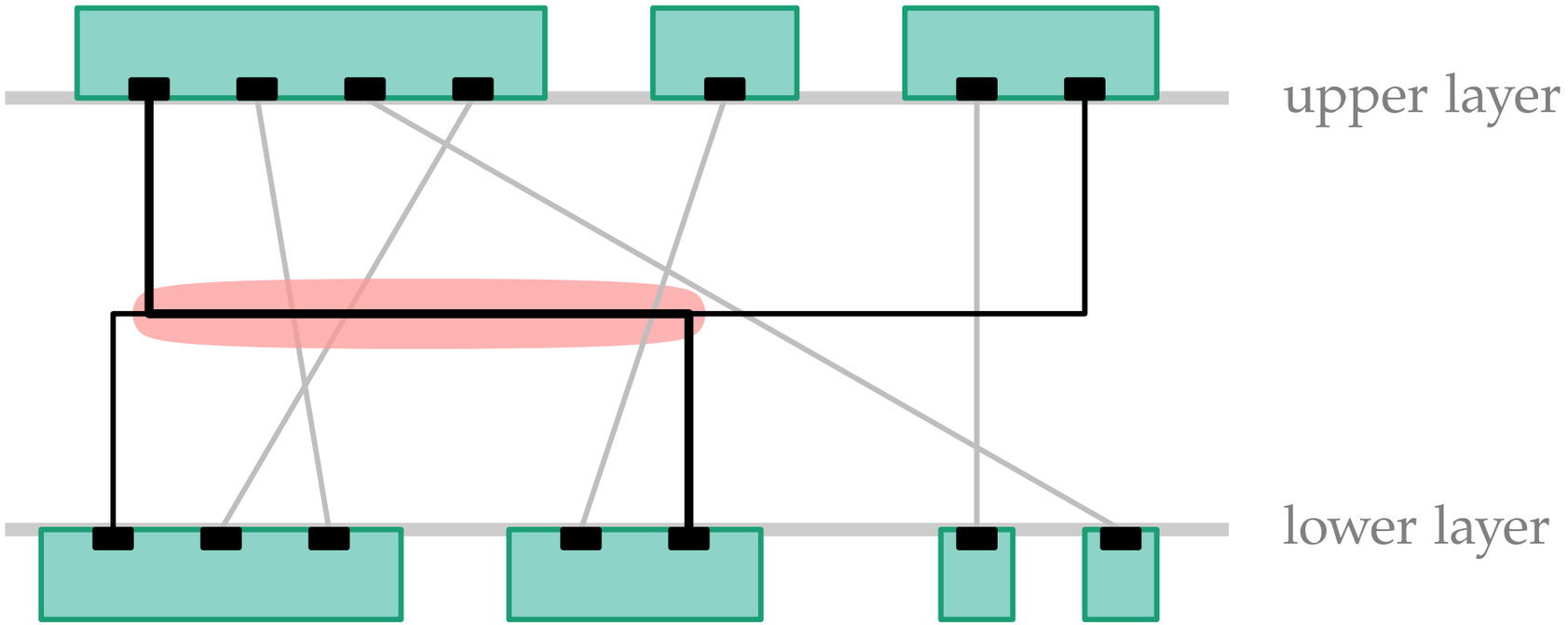
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges



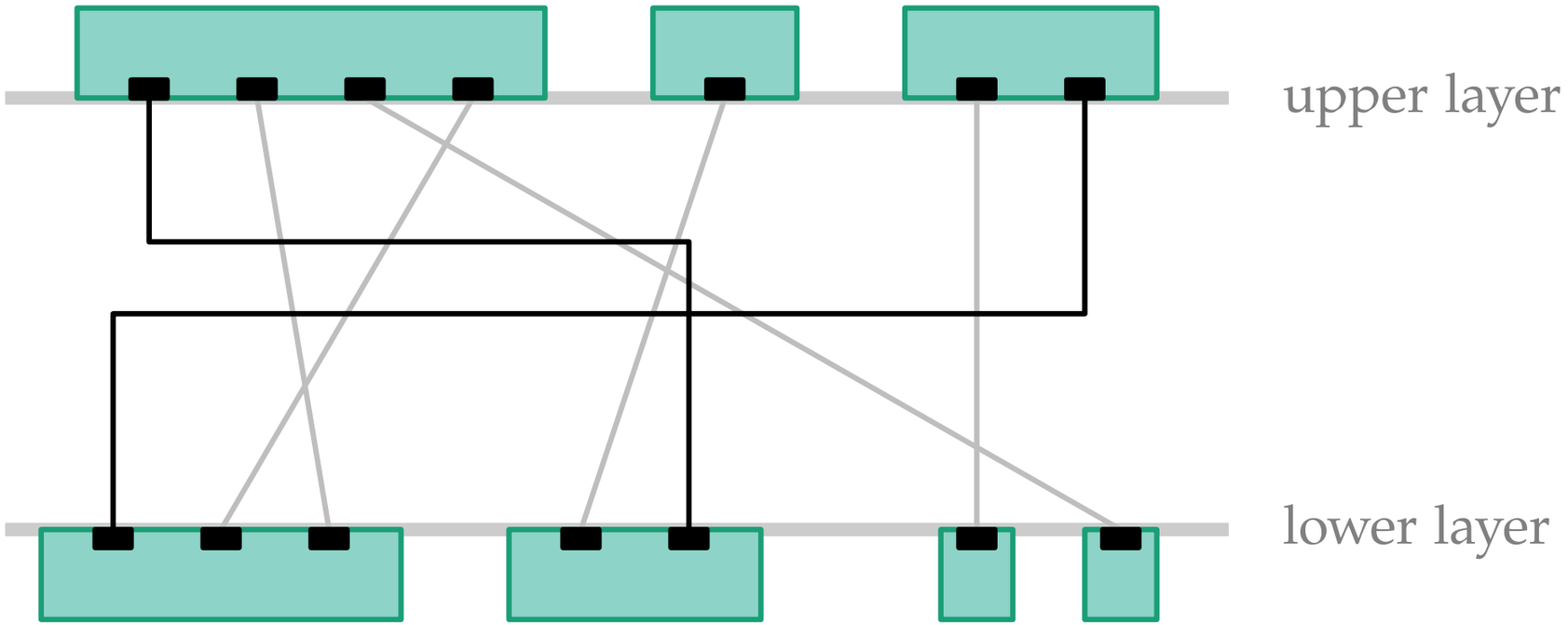
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges



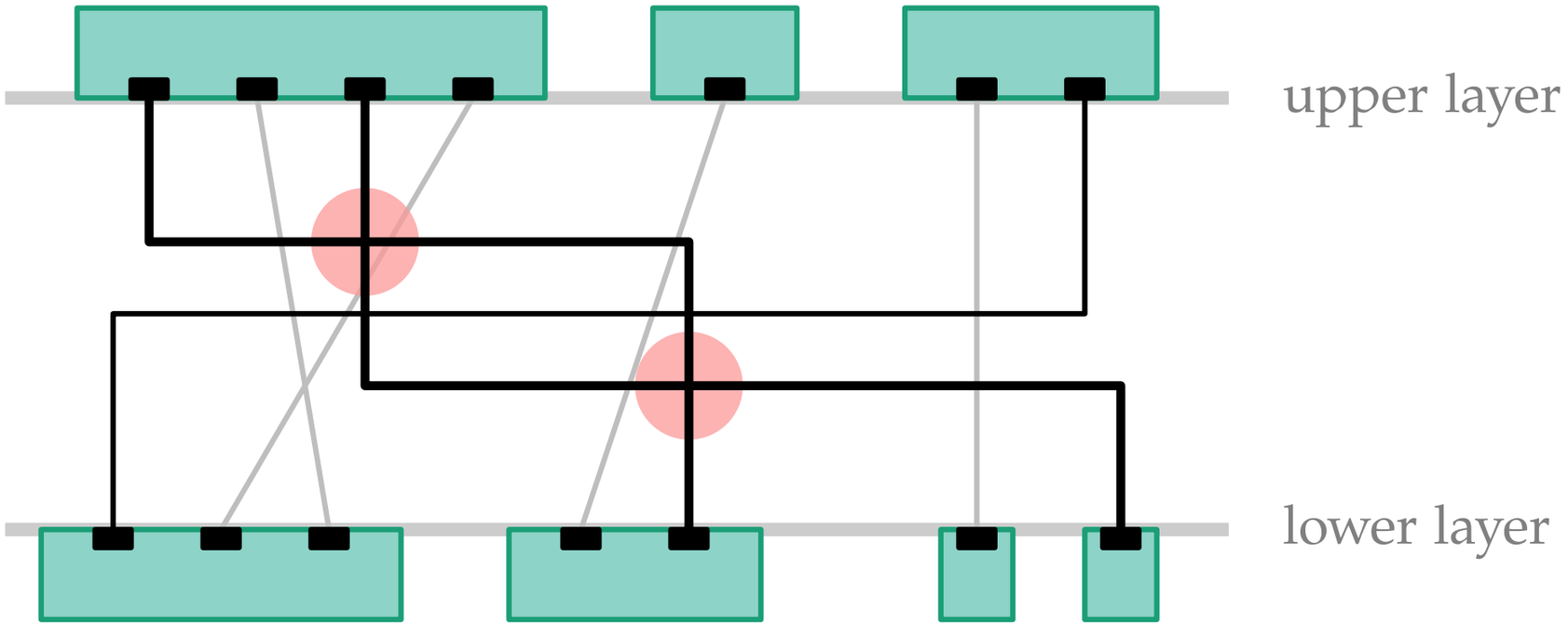
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges



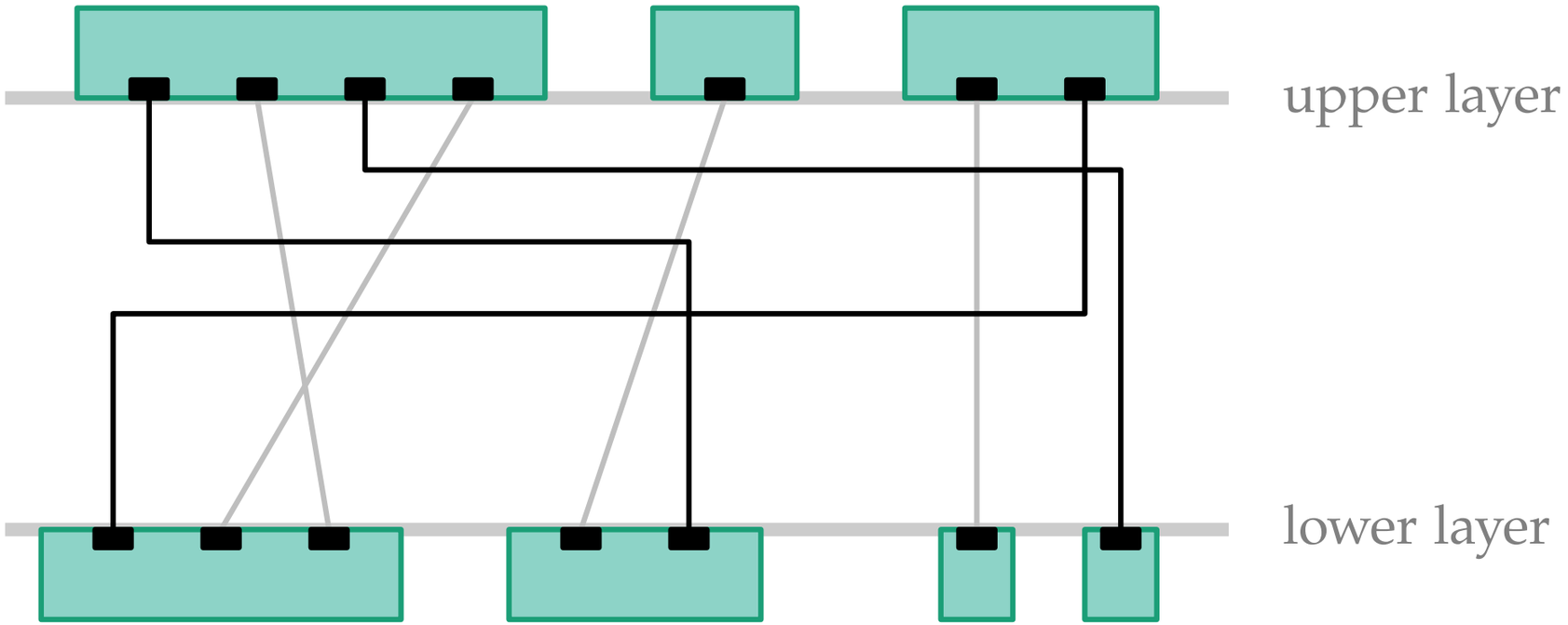
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges



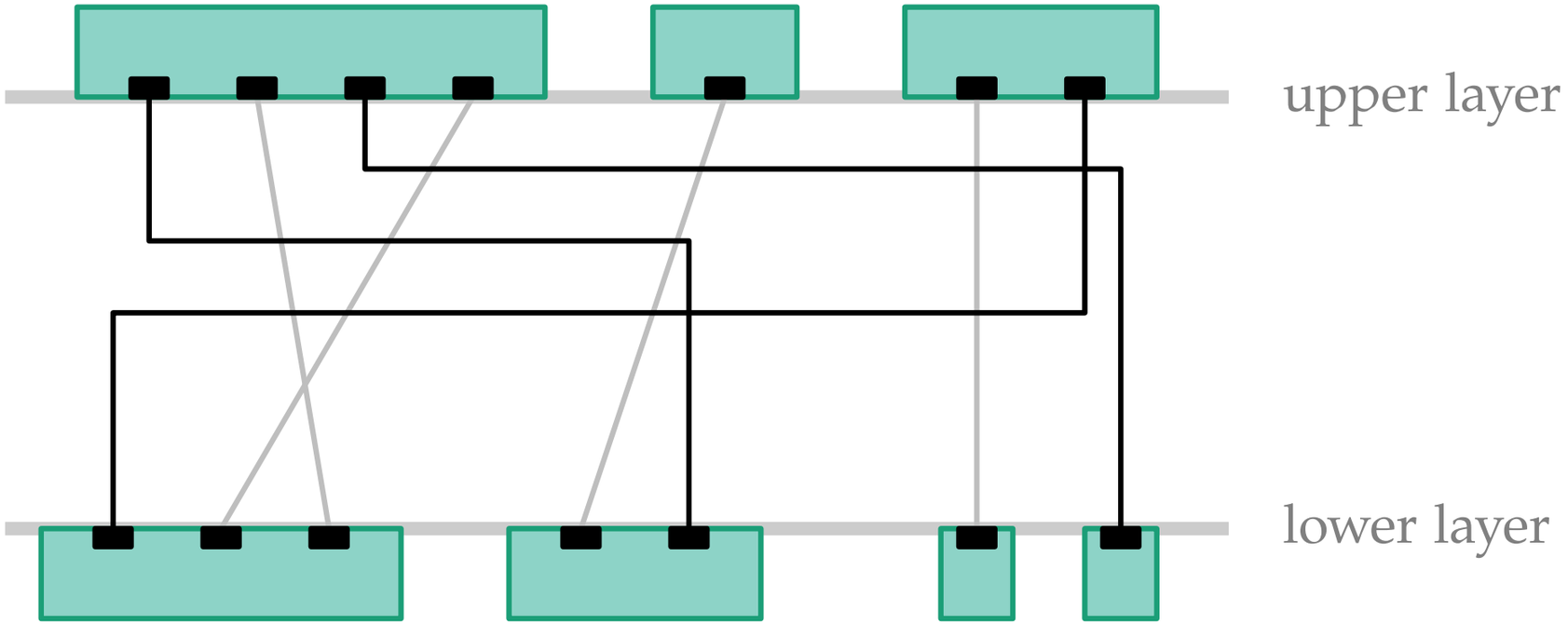
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges



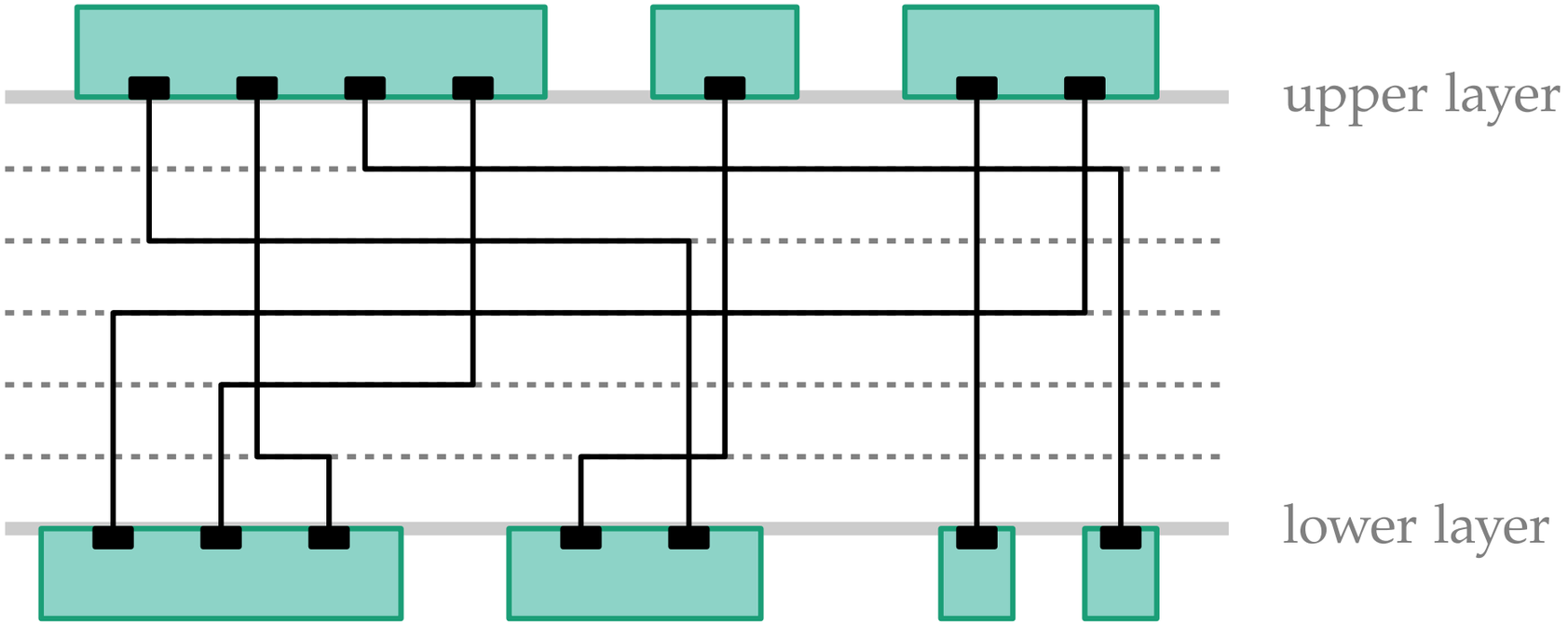
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges
- use as few horizontal intermediate layers (tracks) as possible



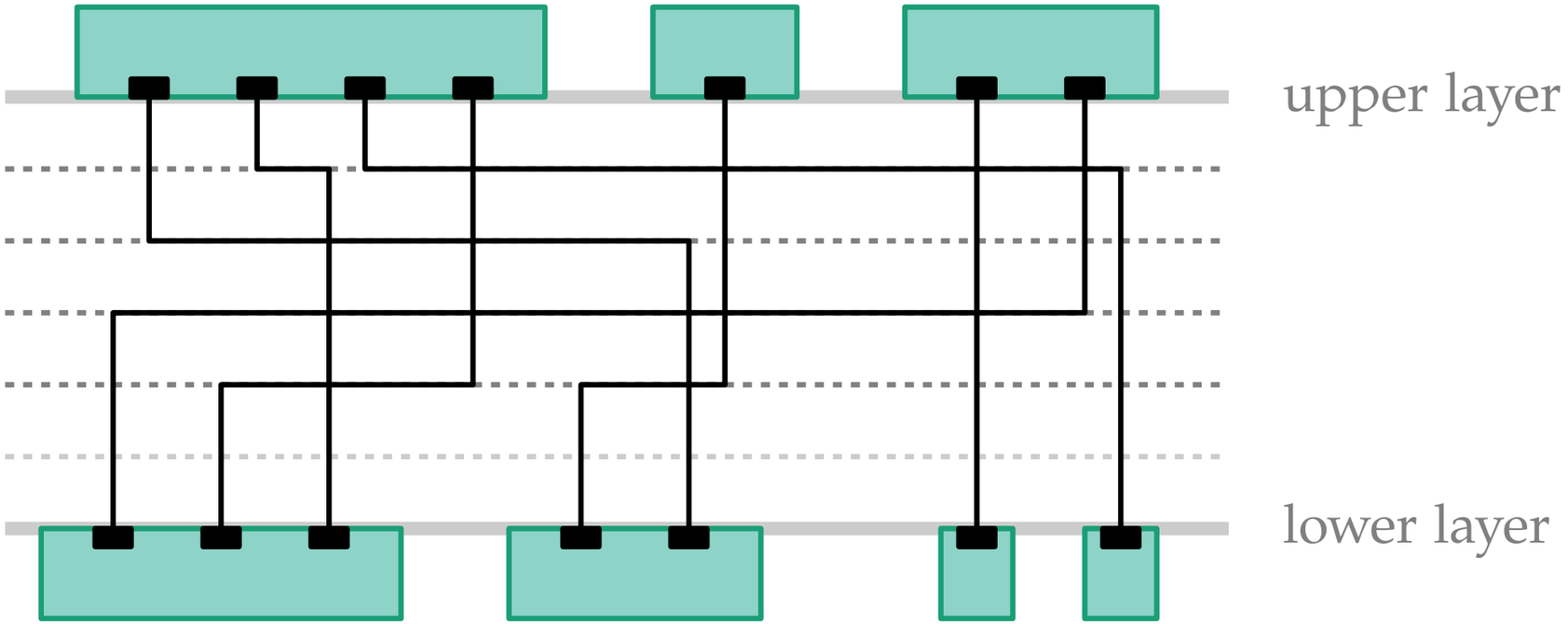
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges
- use as few horizontal intermediate layers (tracks) as possible



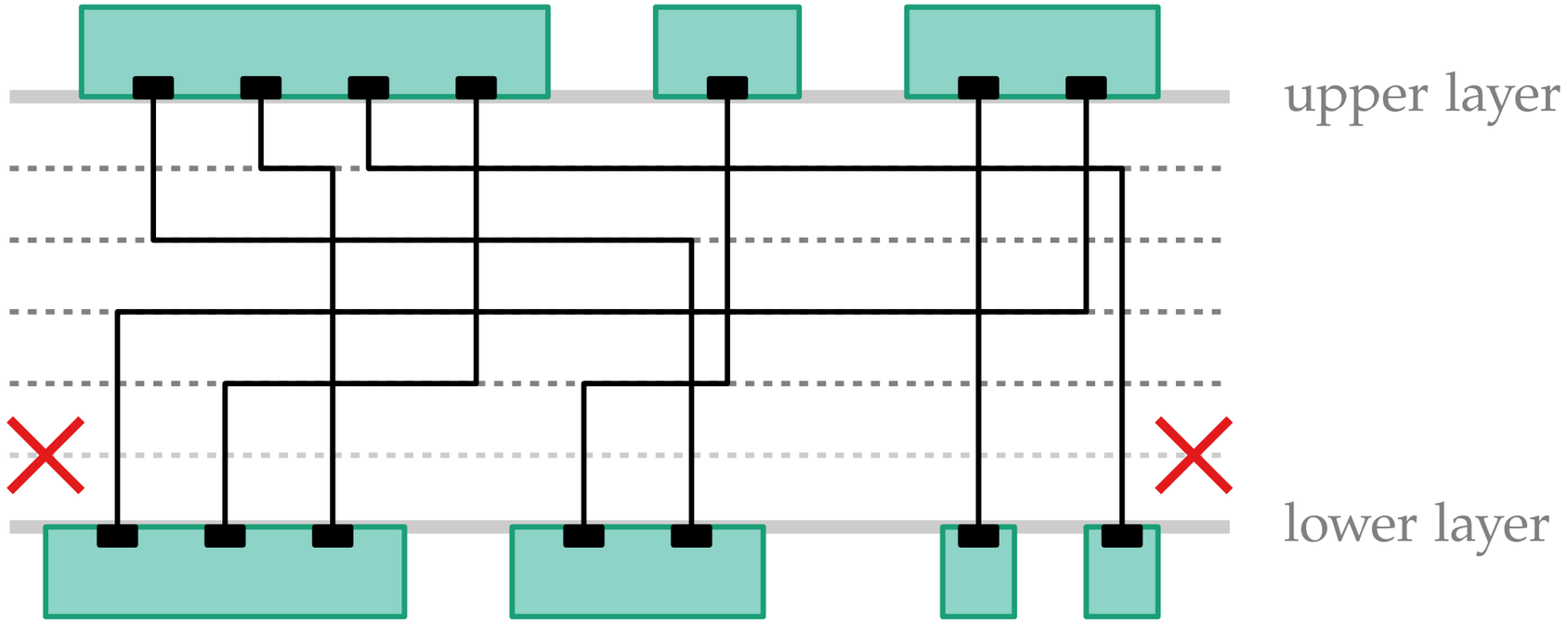
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges
- use as few horizontal intermediate layers (tracks) as possible



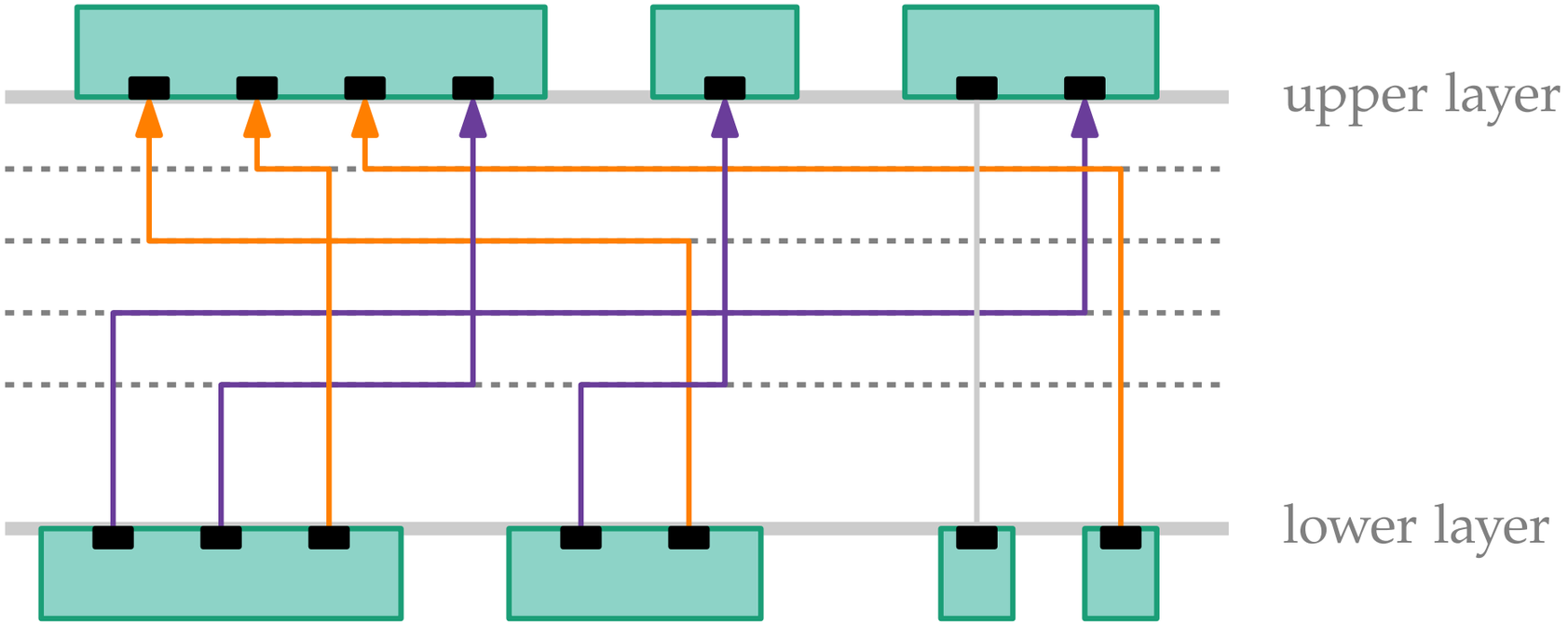
Motivation – Layered Orthogonal Edge Routing

- draw each edge with at most two vertical and one horizontal line segments
- avoid overlaps and double crossings between the same pair of edges
- use as few horizontal intermediate layers (tracks) as possible



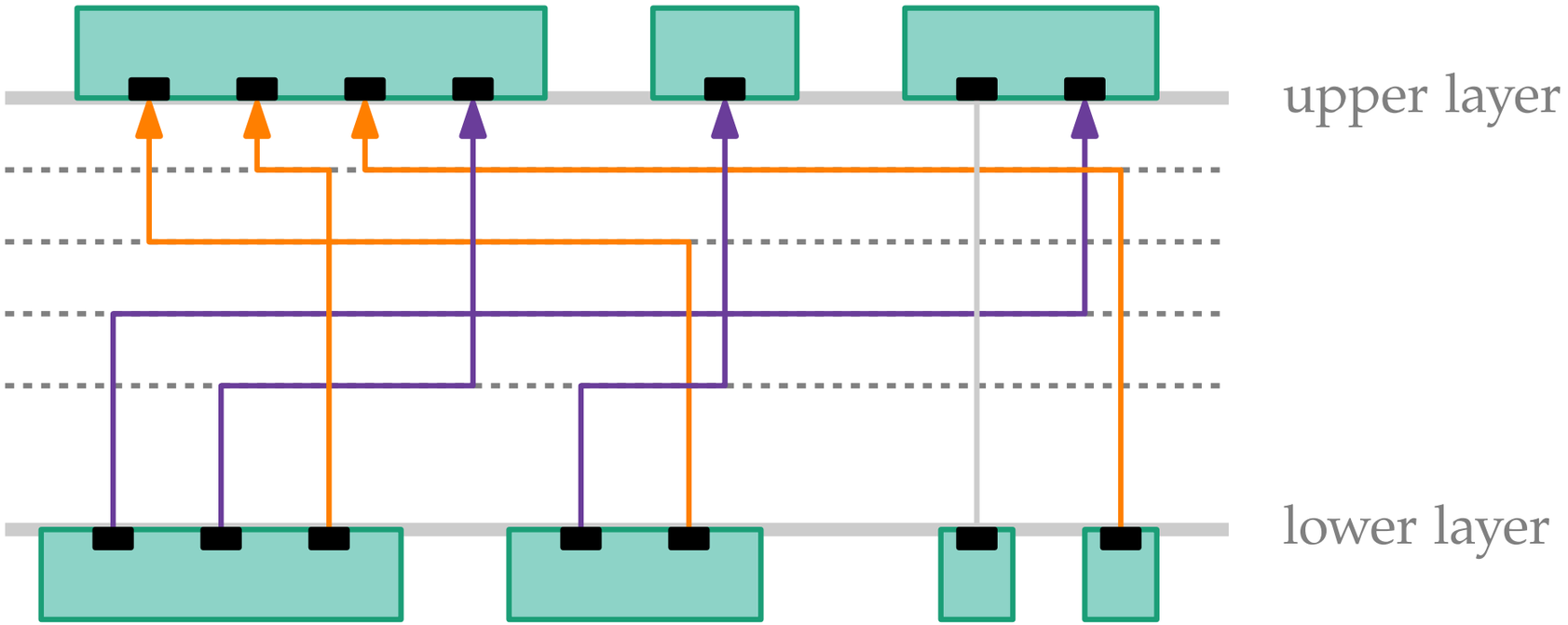
Motivation – Layered Orthogonal Edge Routing

- distinguish between *left-going* and *right-going* edges



Motivation – Layered Orthogonal Edge Routing

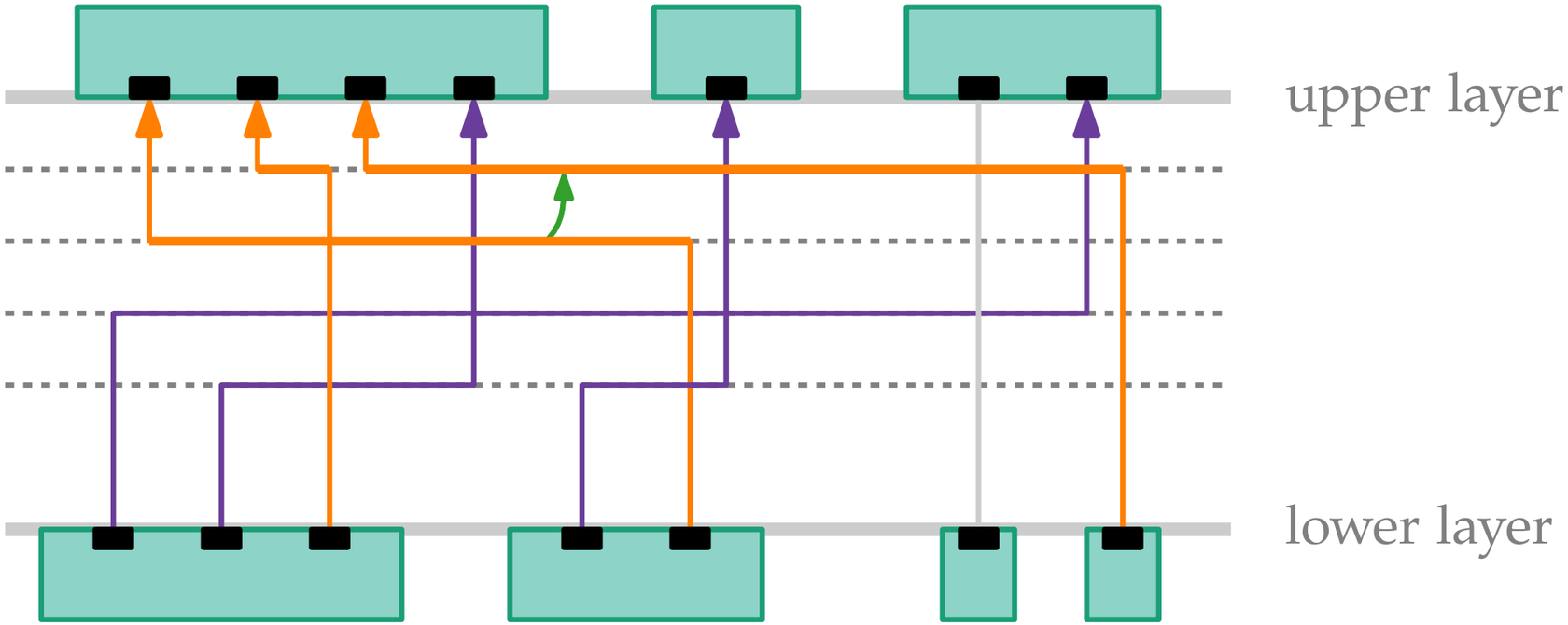
- distinguish between *left-going* and *right-going* edges
- only edges going in the same direction and overlapping partially in x-dimension can cross twice



Motivation – Layered Orthogonal Edge Routing

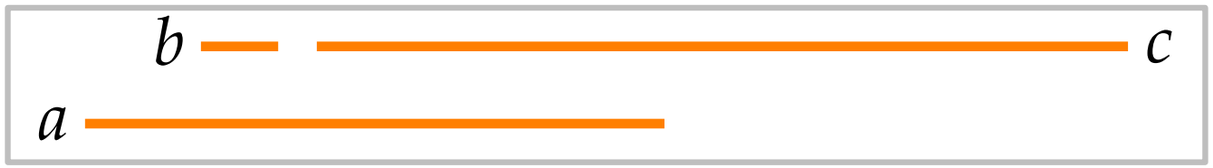
- distinguish between *left-going* and *right-going* edges
- only edges going in the same direction and overlapping partially in x-dimension can cross twice

  $\Rightarrow$ induce a vertical order for the horizontal middle segments



Definition – Directional Interval Graphs

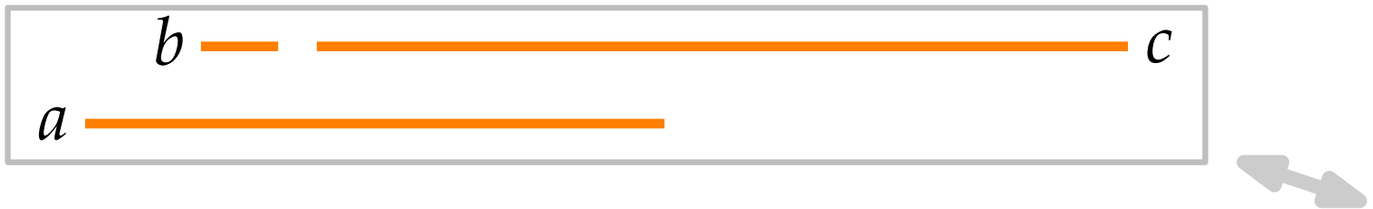
Interval representation: set of intervals



Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

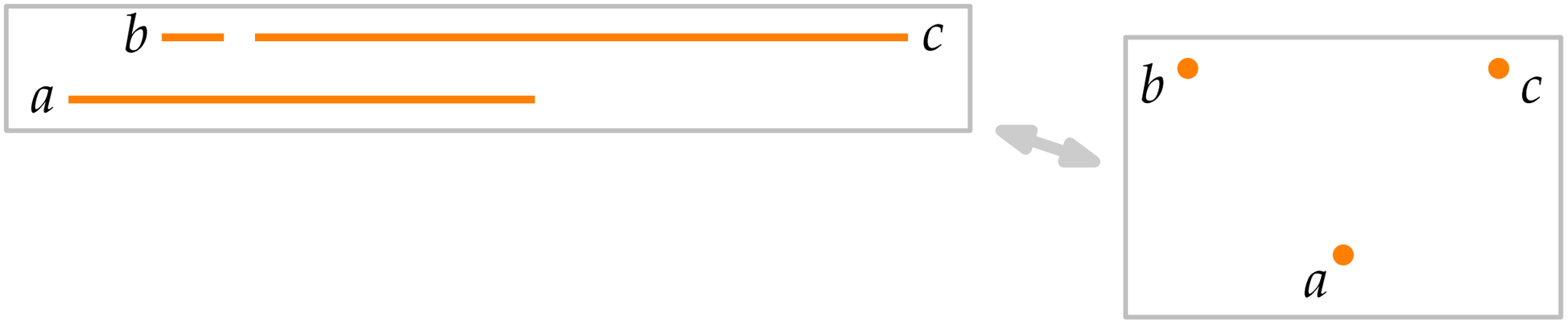


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval

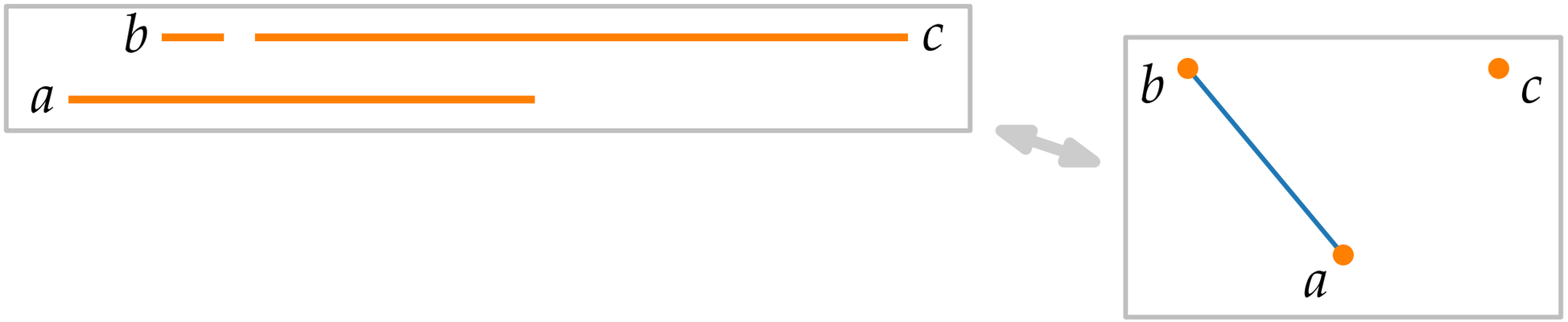


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another

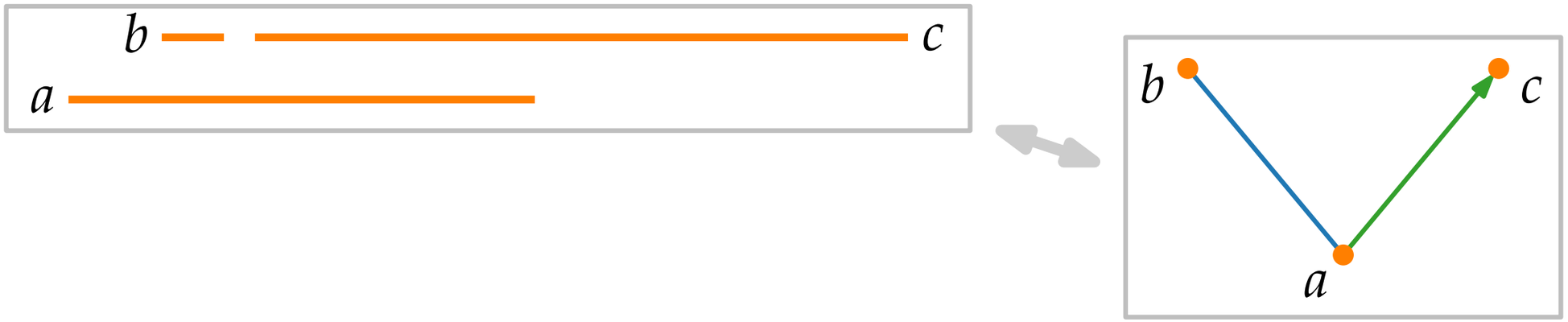


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially

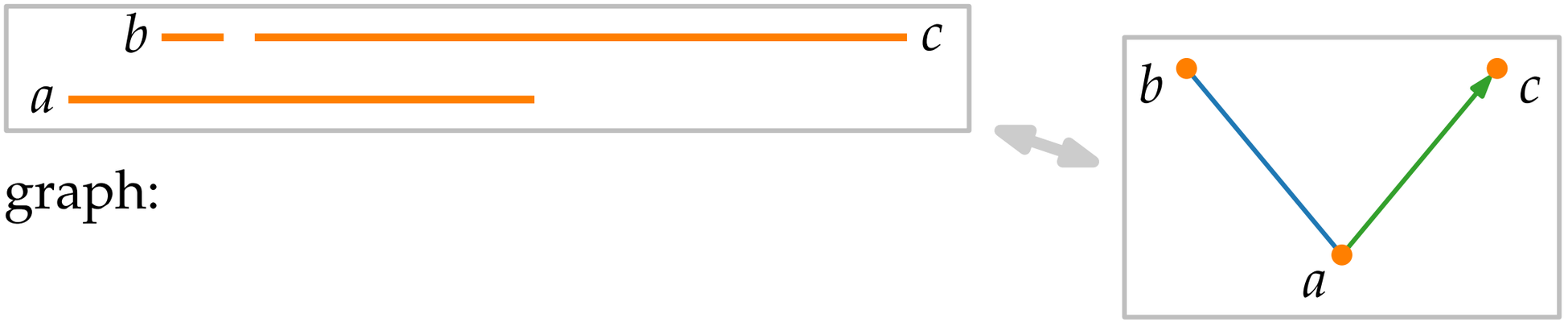


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially



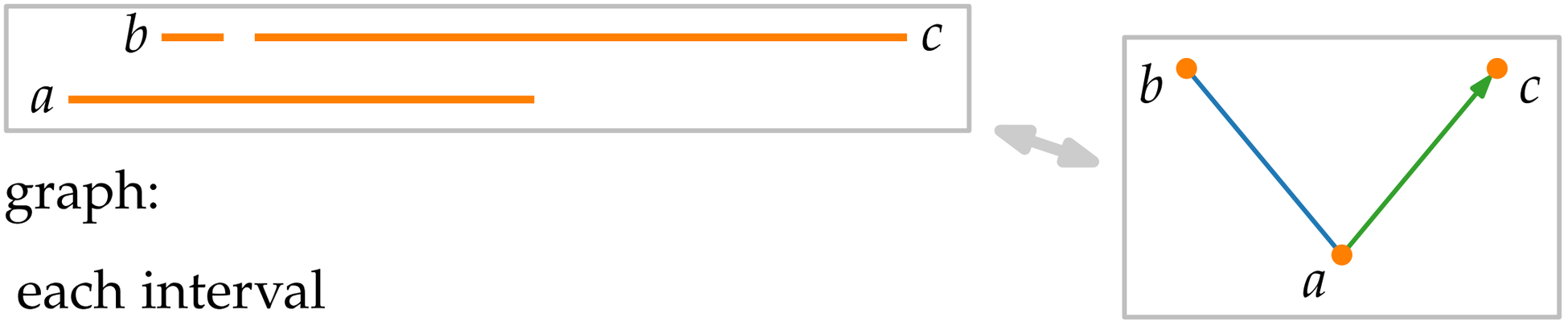
Mixed interval graph:

Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially



Mixed interval graph:

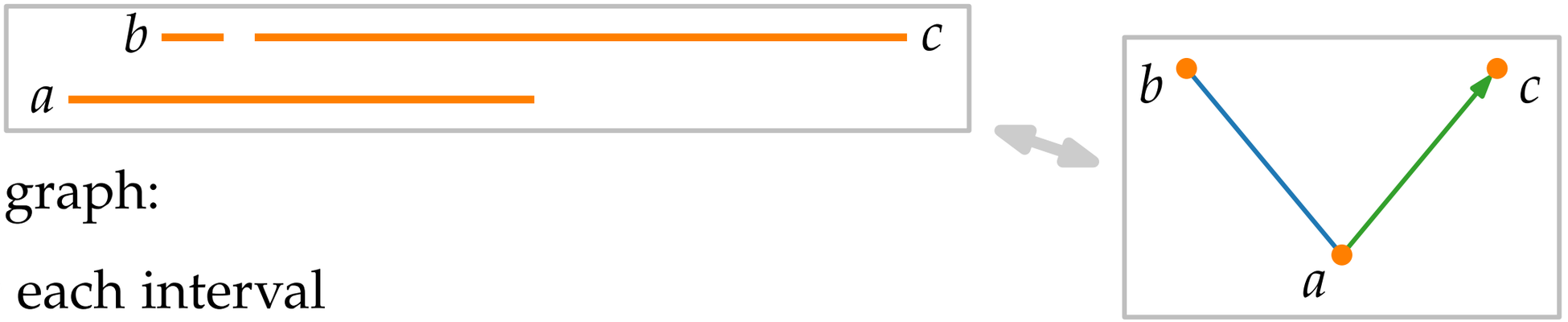
- vertex for each interval

Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval
- undirected edge if one interval contains another
- directed edge (towards the right interval) if the intervals overlap partially



Mixed interval graph:

- vertex for each interval
- for each two overlapping intervals: undirected or arbitrarily directed edge

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directed graphs (only directed edges):

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directed graphs (only directed edges):

- coloring in linear time using topological sorting

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:

[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs:

Directed graphs (only directed edges):

- coloring in linear time using topological sorting

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs:

- recognition in $O(n^2)$ time

Directed graphs (only directed edges):

- coloring in linear time using topological sorting

$n :=$ # intervals

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs:

- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

Directed graphs (only directed edges):

- coloring in linear time using topological sorting

$n :=$ # intervals

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

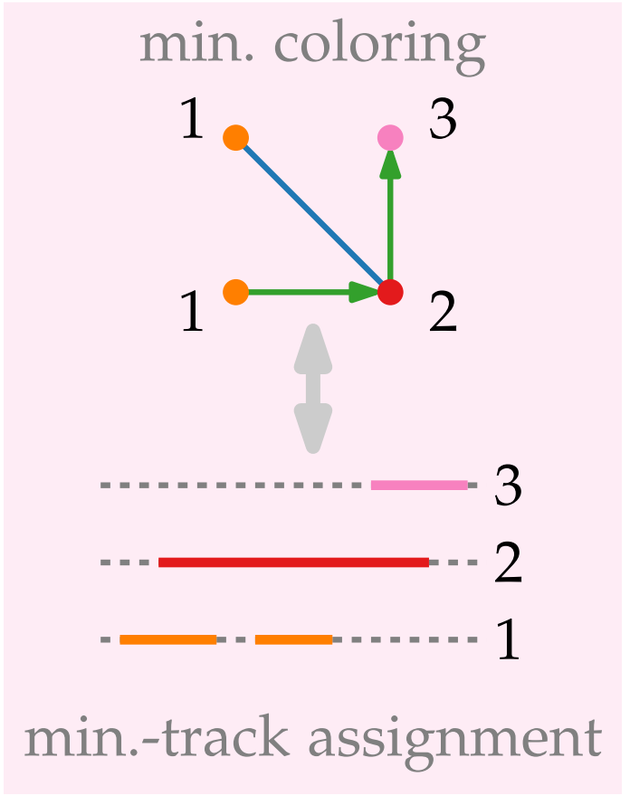
- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs:

- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm



Directed graphs (only directed edges):

- coloring in linear time using topological sorting

$n := \#$ intervals

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

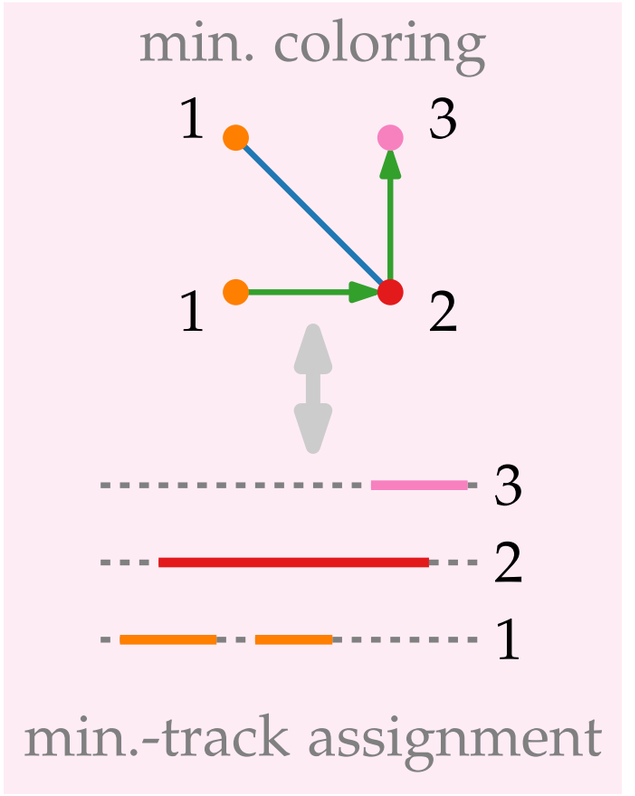
Directional interval graphs:

- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

Mixed interval graphs:

Directed graphs (only directed edges):

- coloring in linear time using topological sorting



$n :=$ # intervals

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs:

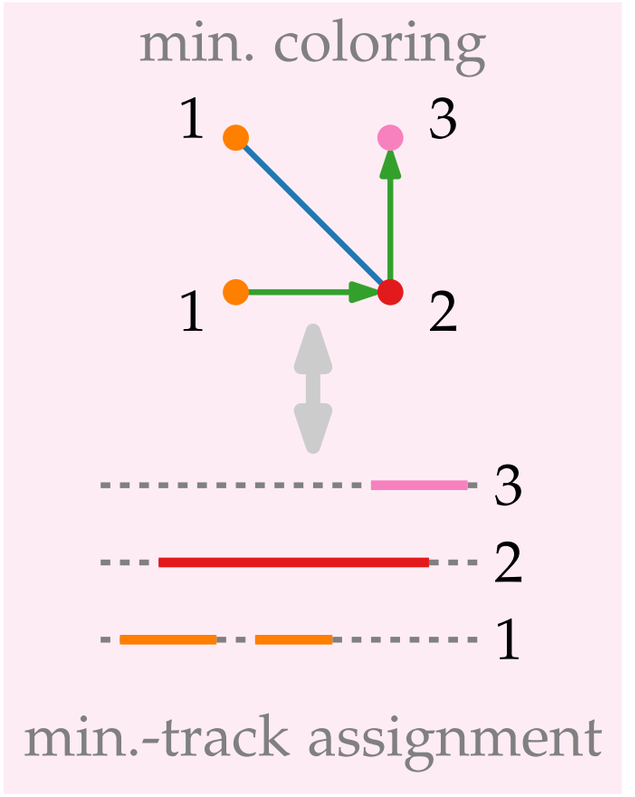
- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

Mixed interval graphs:

- coloring is NP-complete

Directed graphs (only directed edges):

- coloring in linear time using topological sorting



n := # intervals

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs: **our contribution**

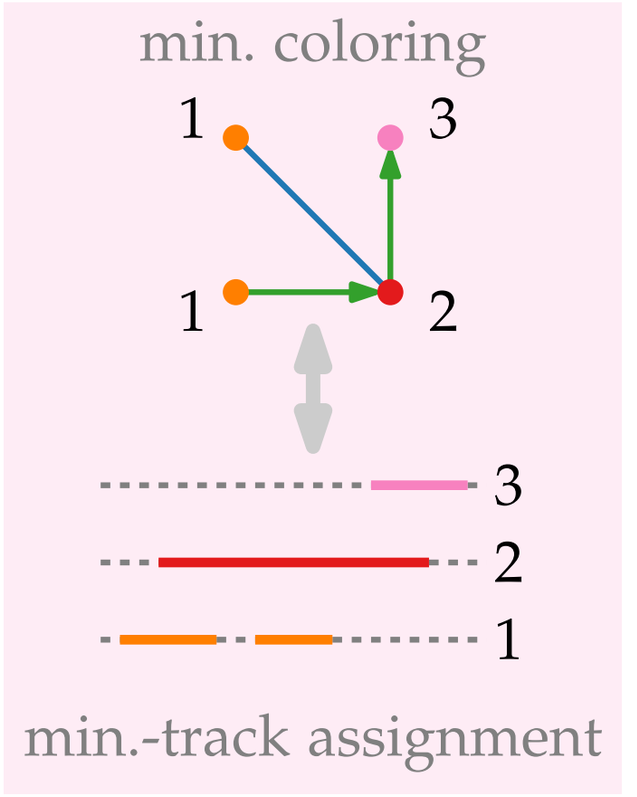
- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

Mixed interval graphs:

- coloring is NP-complete

Directed graphs (only directed edges):

- coloring in linear time using topological sorting



$n :=$ # intervals

Coloring Mixed Graphs

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- ⋆ undirected edge uv: $c(u) \neq c(v)$,
- ⋆ directed edge uv: $c(u) < c(v)$,
- ⋆ $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs: **our contribution**

- recognition in $O(n^2)$ time ← not in this talk
- coloring in $O(n \log n)$ time by a greedy algorithm

Mixed interval graphs:

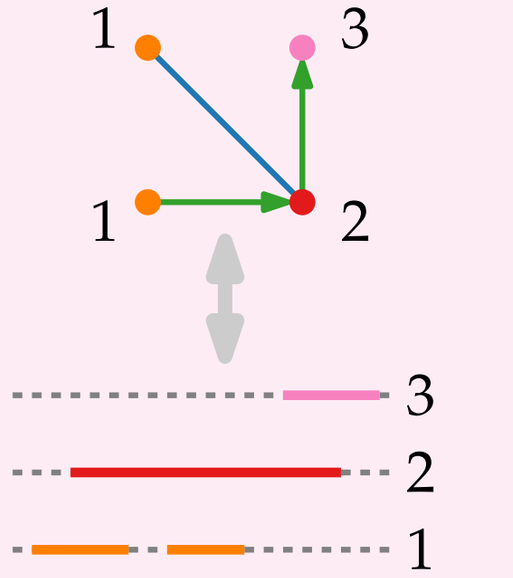
- coloring is NP-complete

agenda for this talk

Directed graphs (only directed edges):

- coloring in linear time using topological sorting

min. coloring



min.-track assignment

$n :=$ # intervals

Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

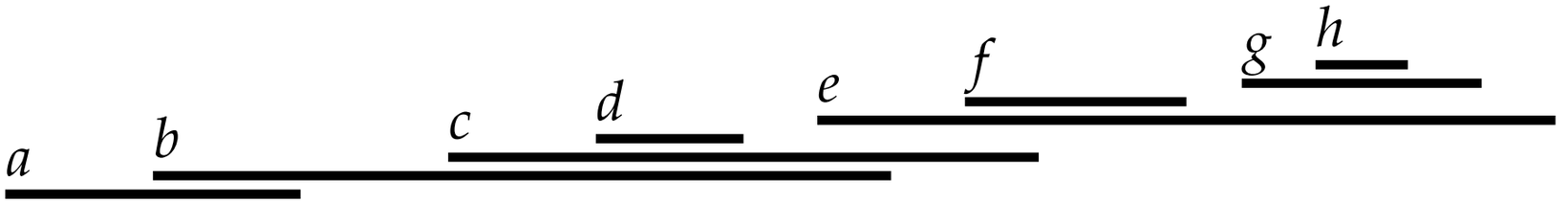
1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

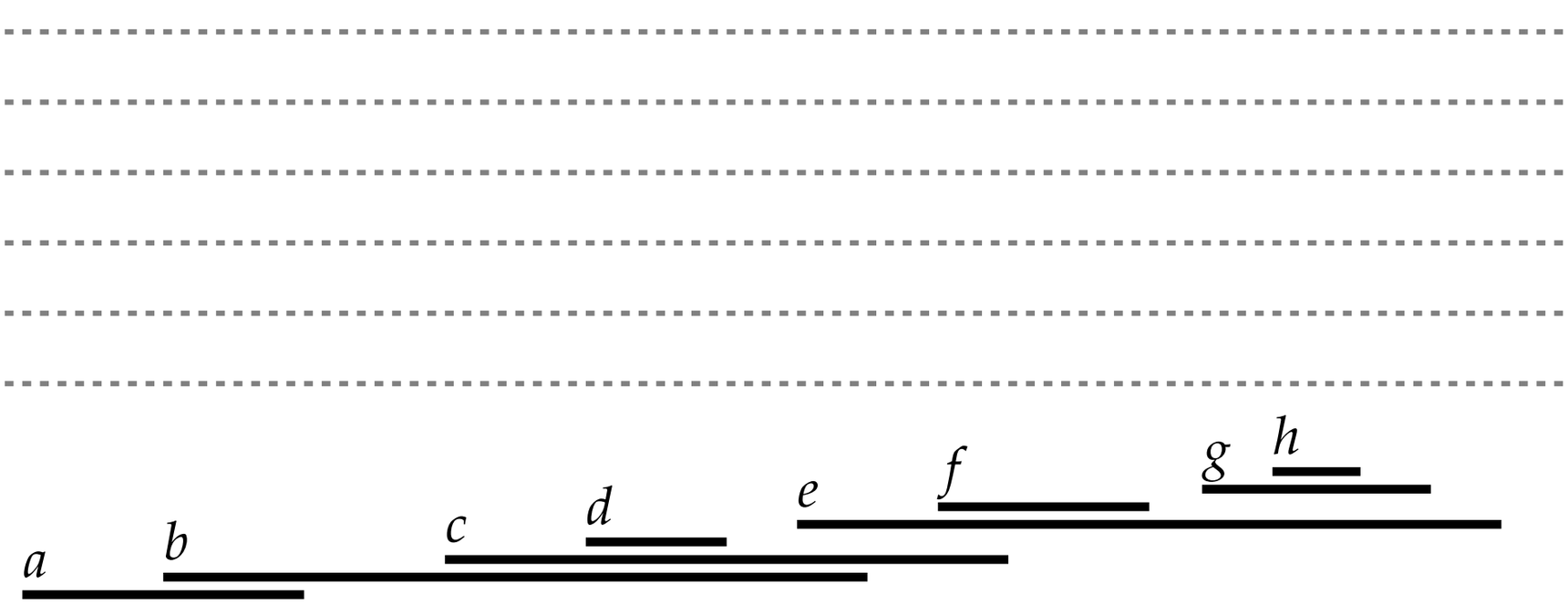


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

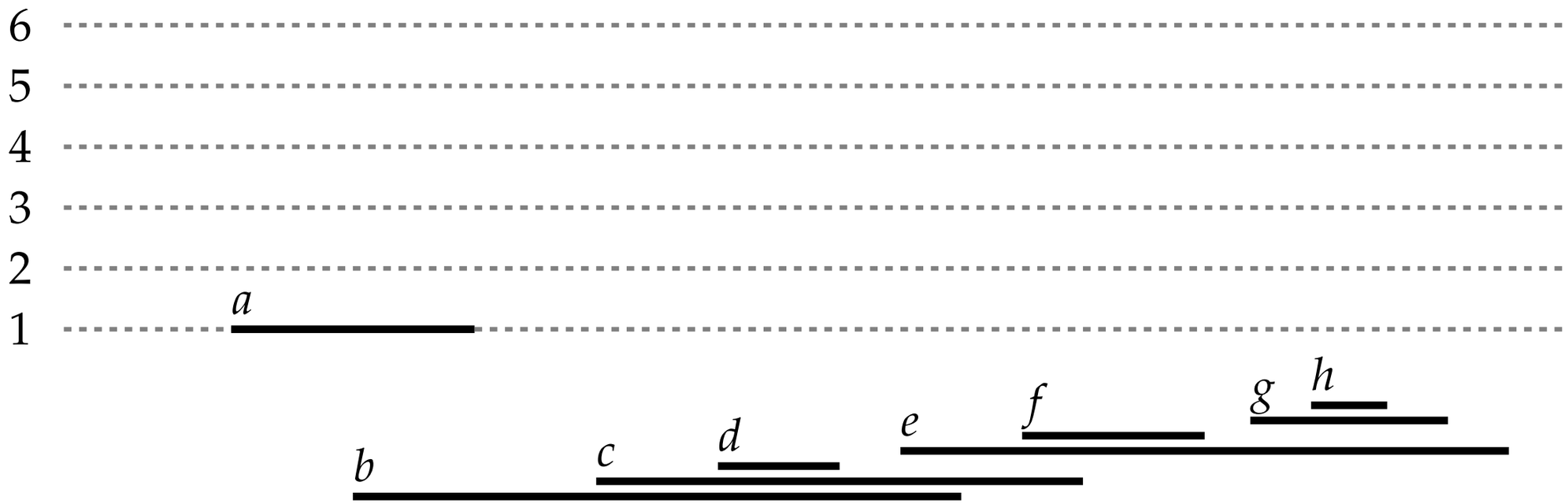


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

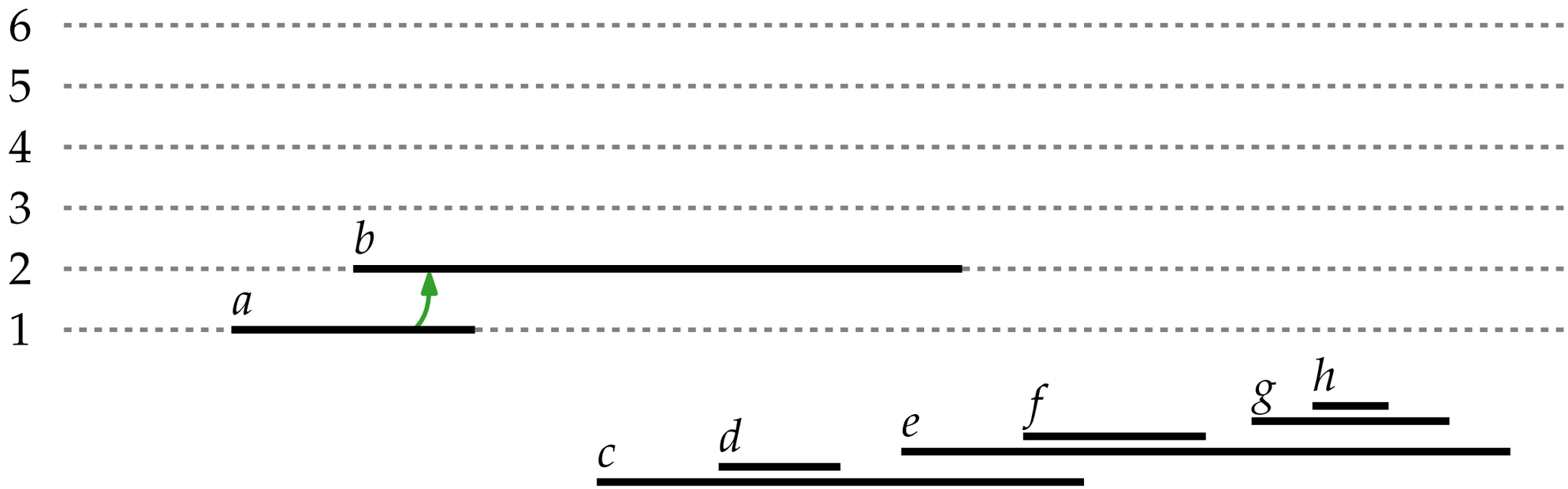


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

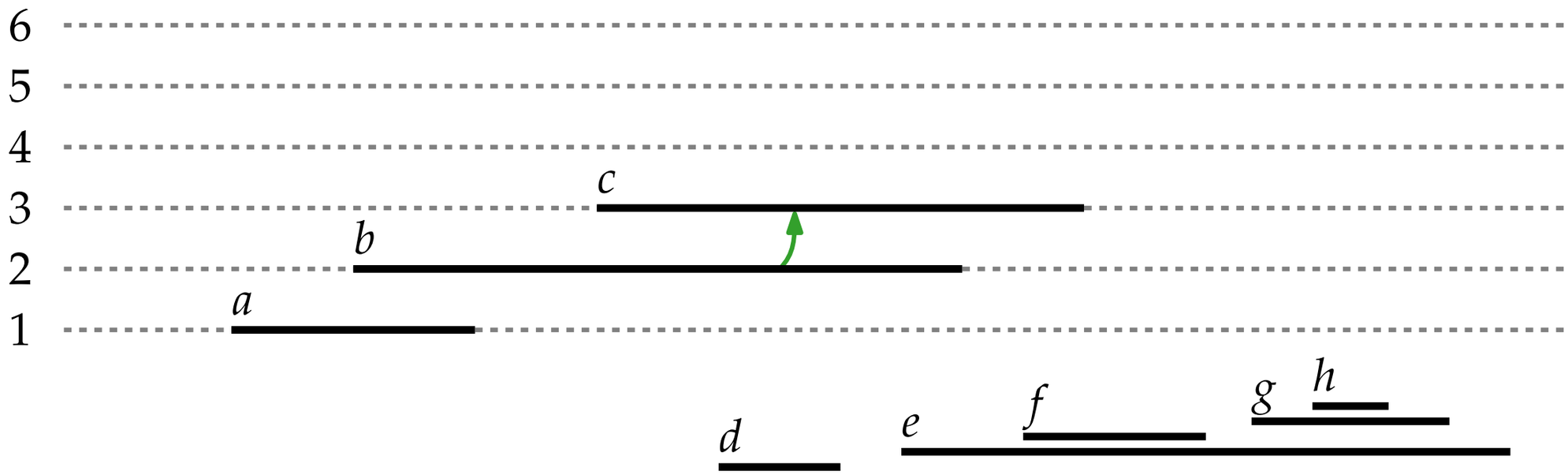


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

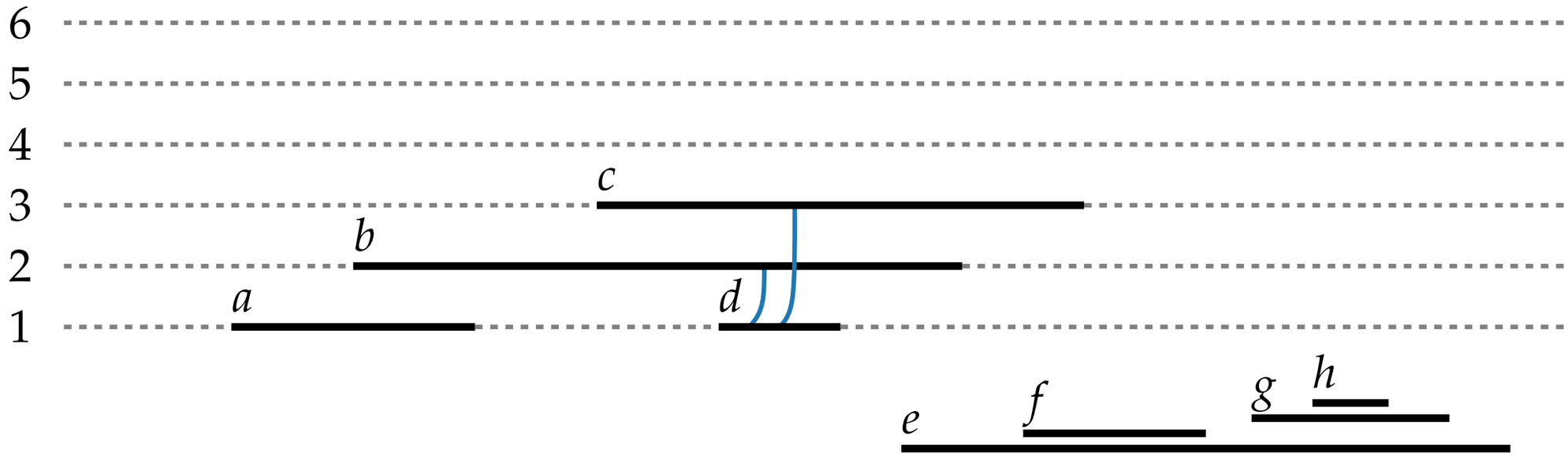


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

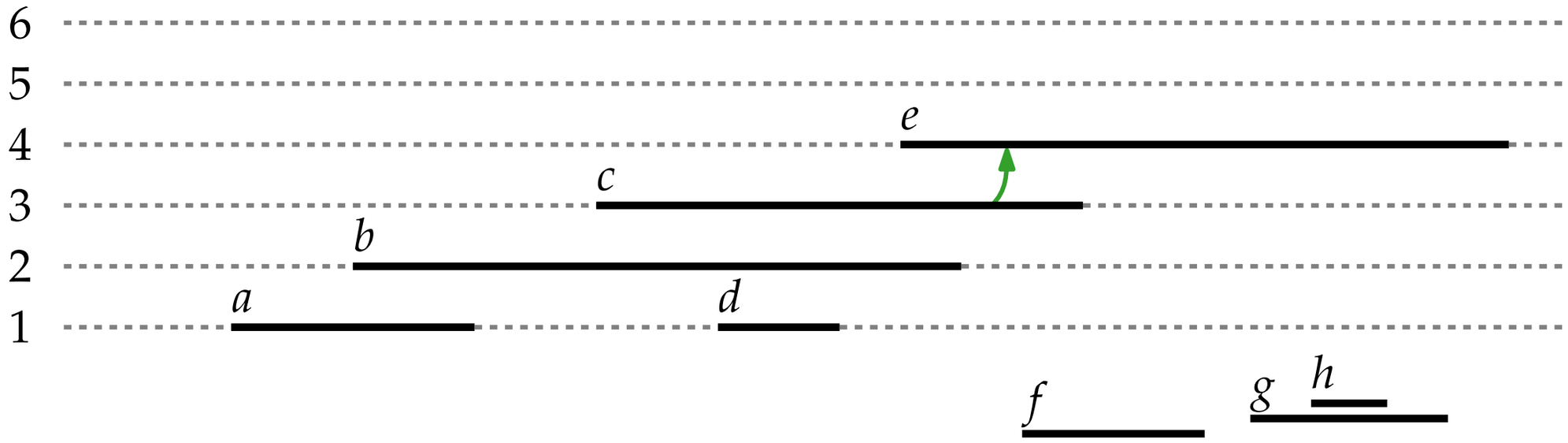


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

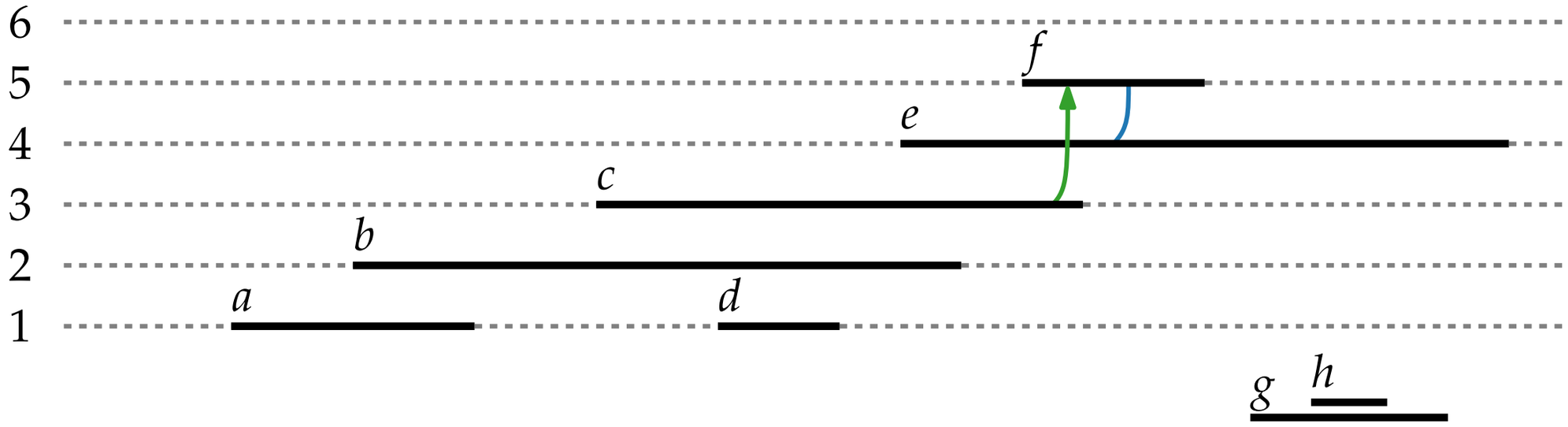


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

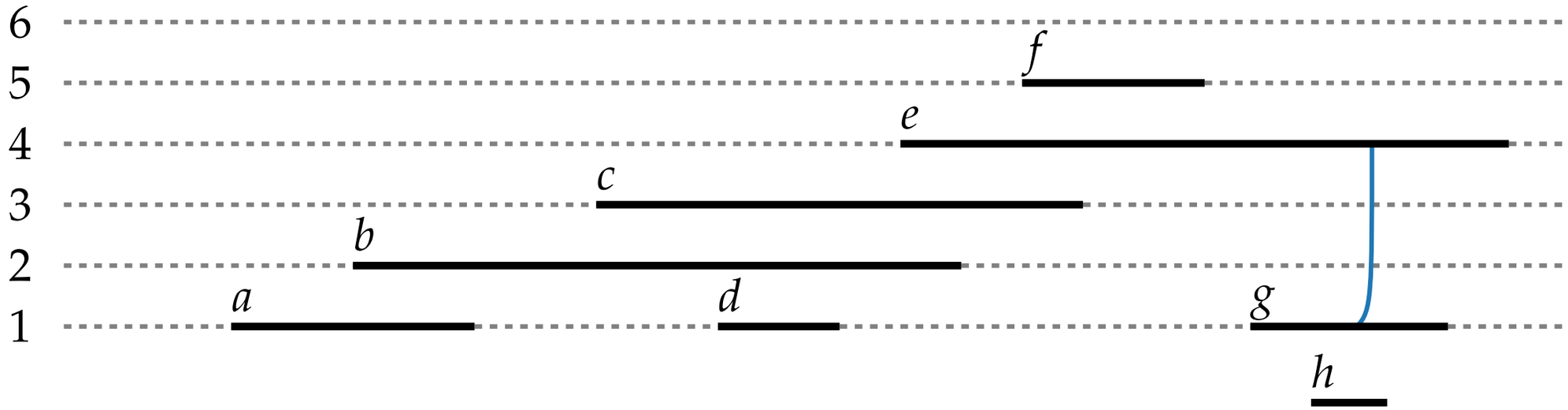


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

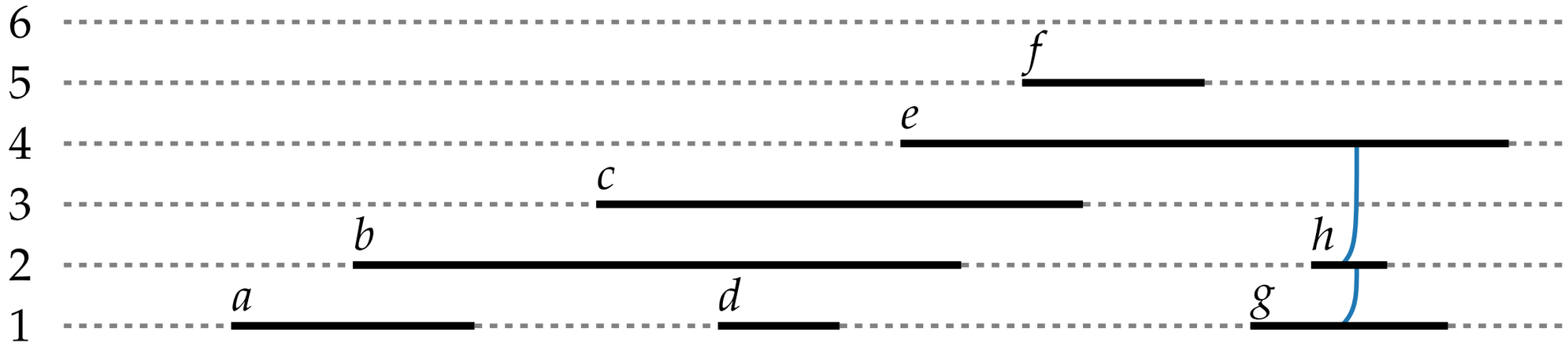


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

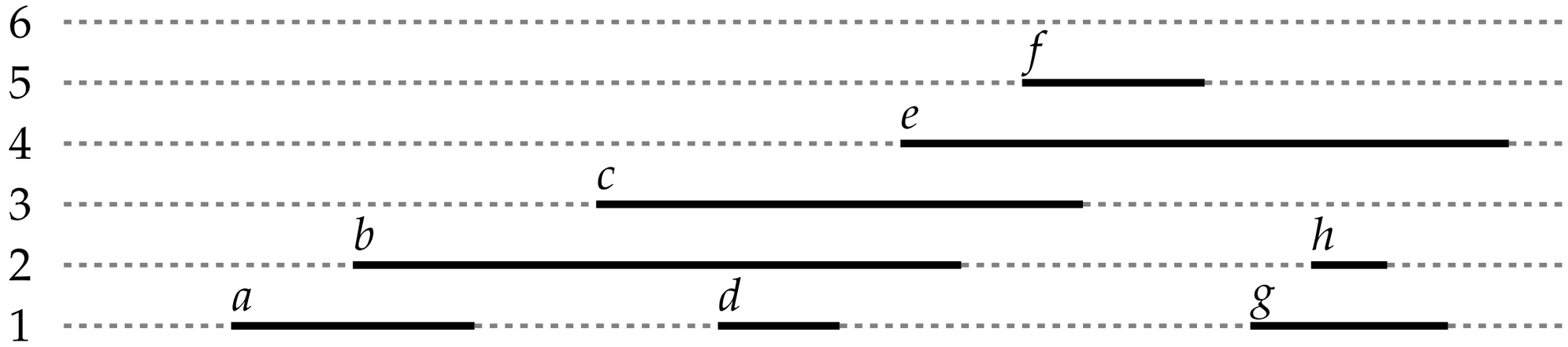


Coloring Directional Interval Graphs

Given: an interval representation of a directional interval graph G

`GreedyColoring:`

1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges



Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let G^+ be the *transitive closure* of G
  (the graph obtained by exhaustively adding transitive directed edges to G).

Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let G^+ be the *transitive closure* of G
  (the graph obtained by exhaustively adding transitive directed edges to G).

- Show: the size of a largest clique in G^+ equals the maximum color m in c.

Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let G^+ be the *transitive closure* of G
  (the graph obtained by exhaustively adding transitive directed edges to G).

- Show: the size of a largest clique in G^+ equals the maximum color m in c.

  $\Rightarrow$ The coloring c uses the minimum number of colors.

Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.

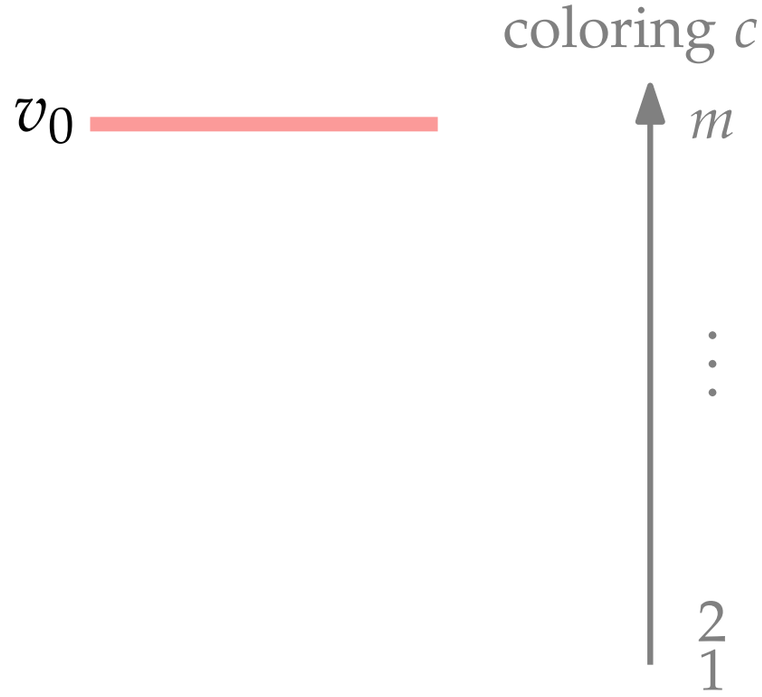
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.



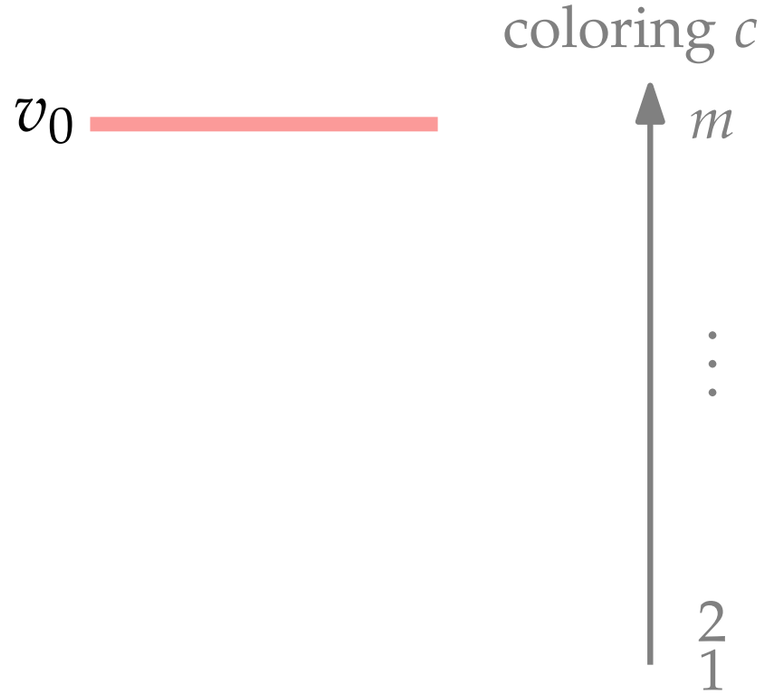
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.



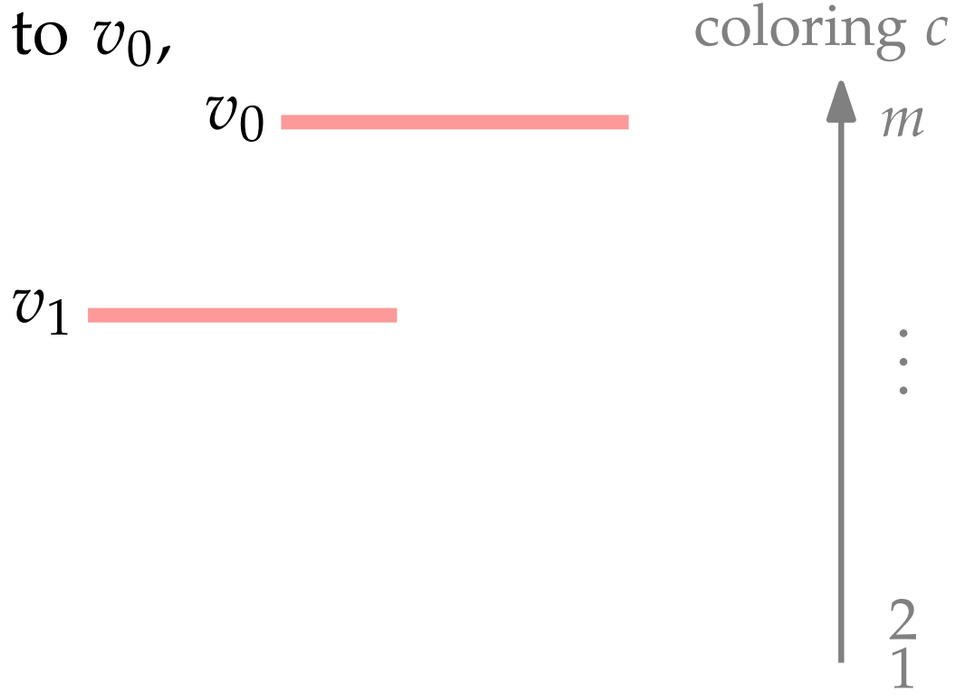
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.



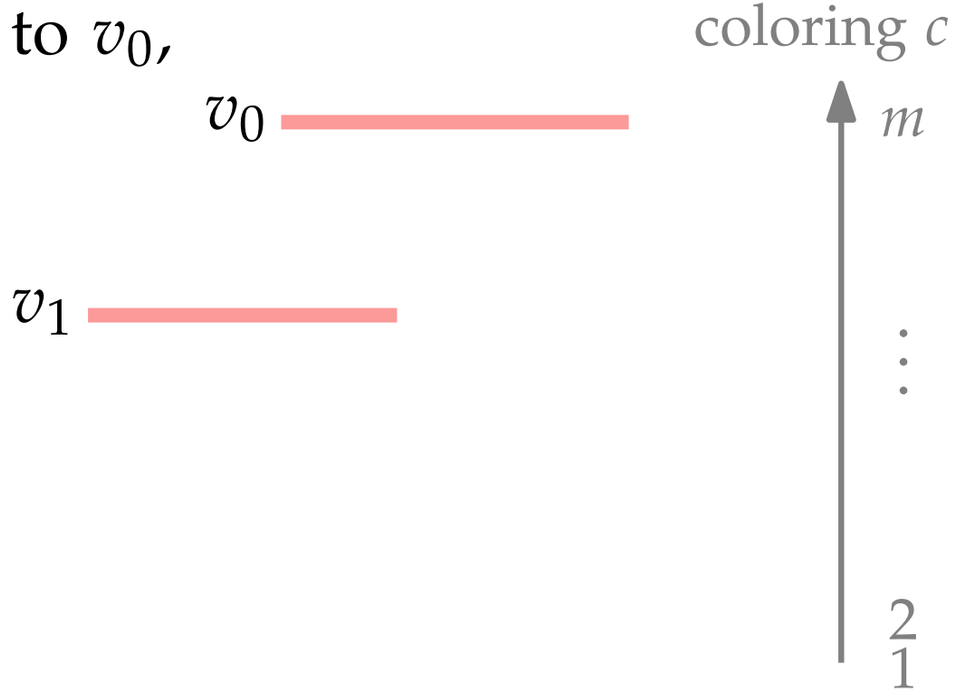
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.



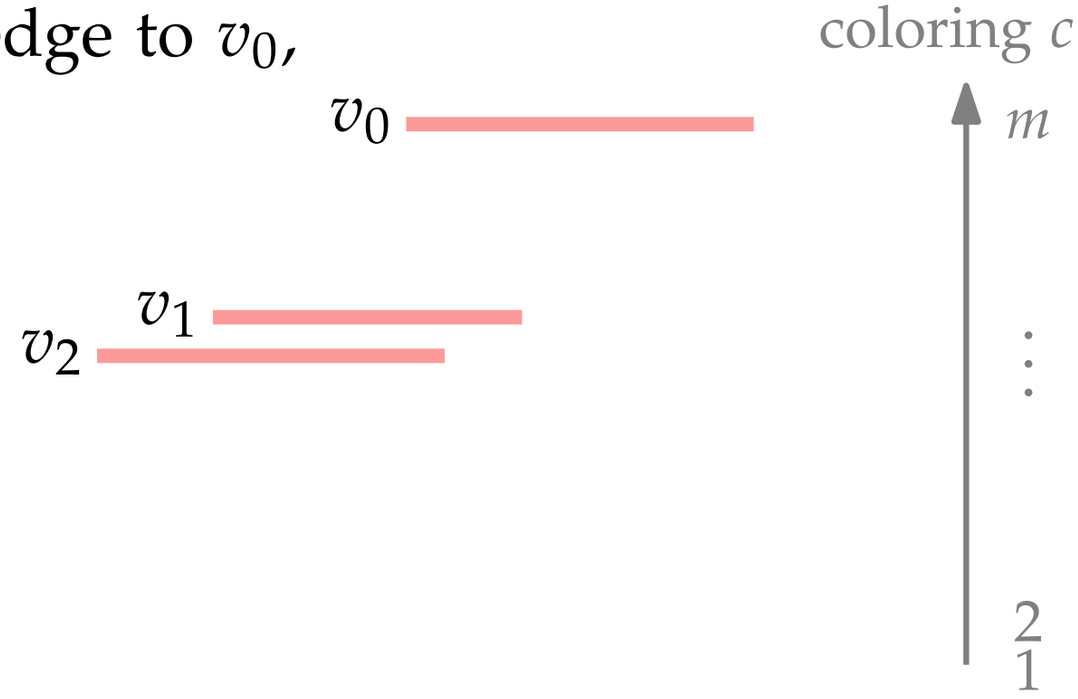
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.



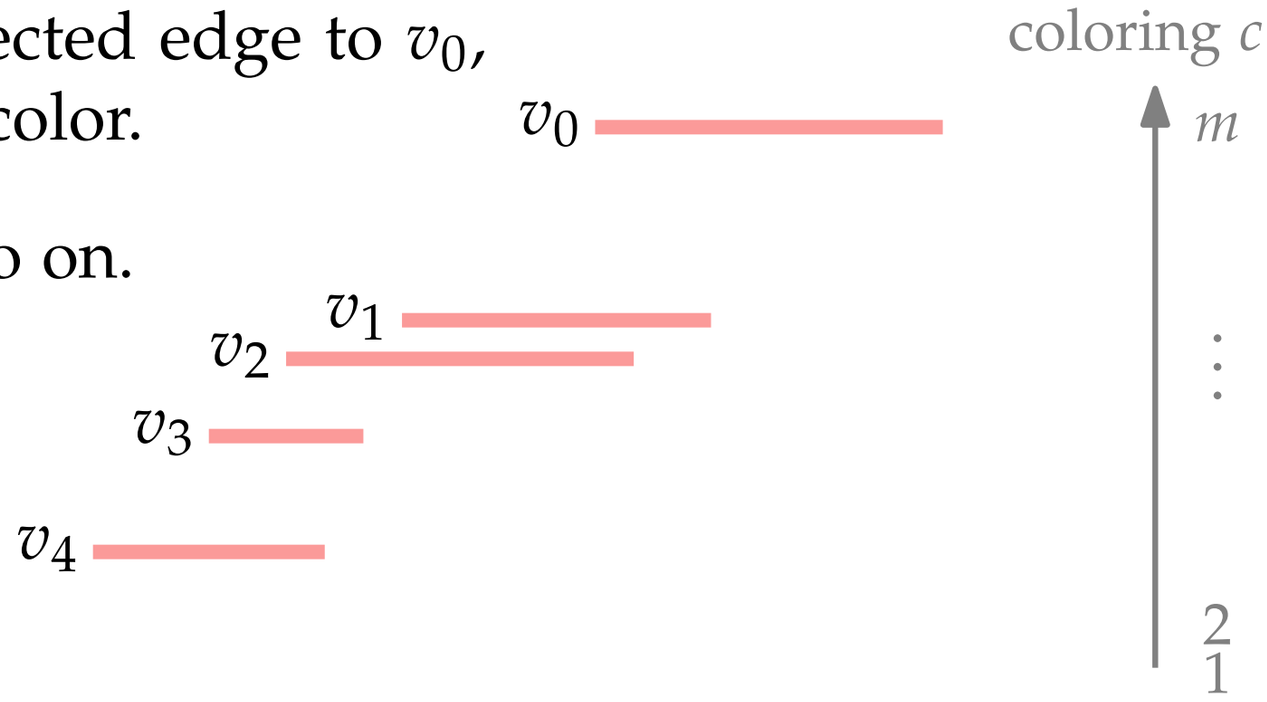
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.



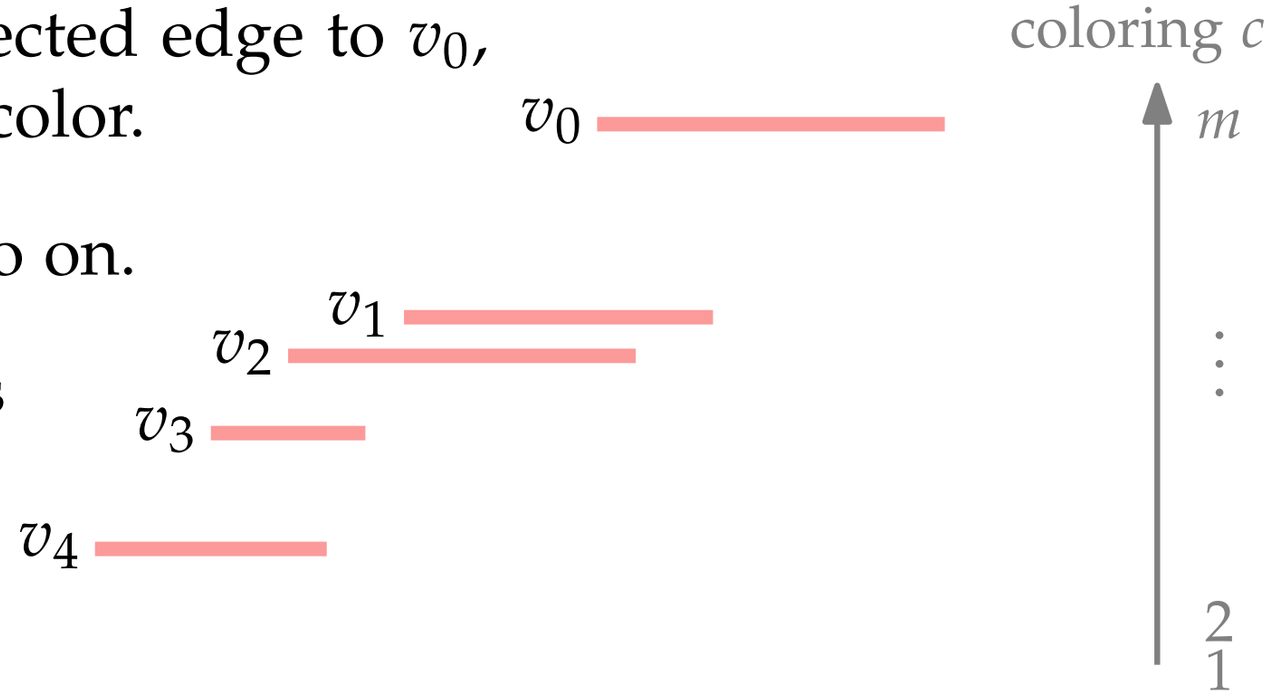
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.



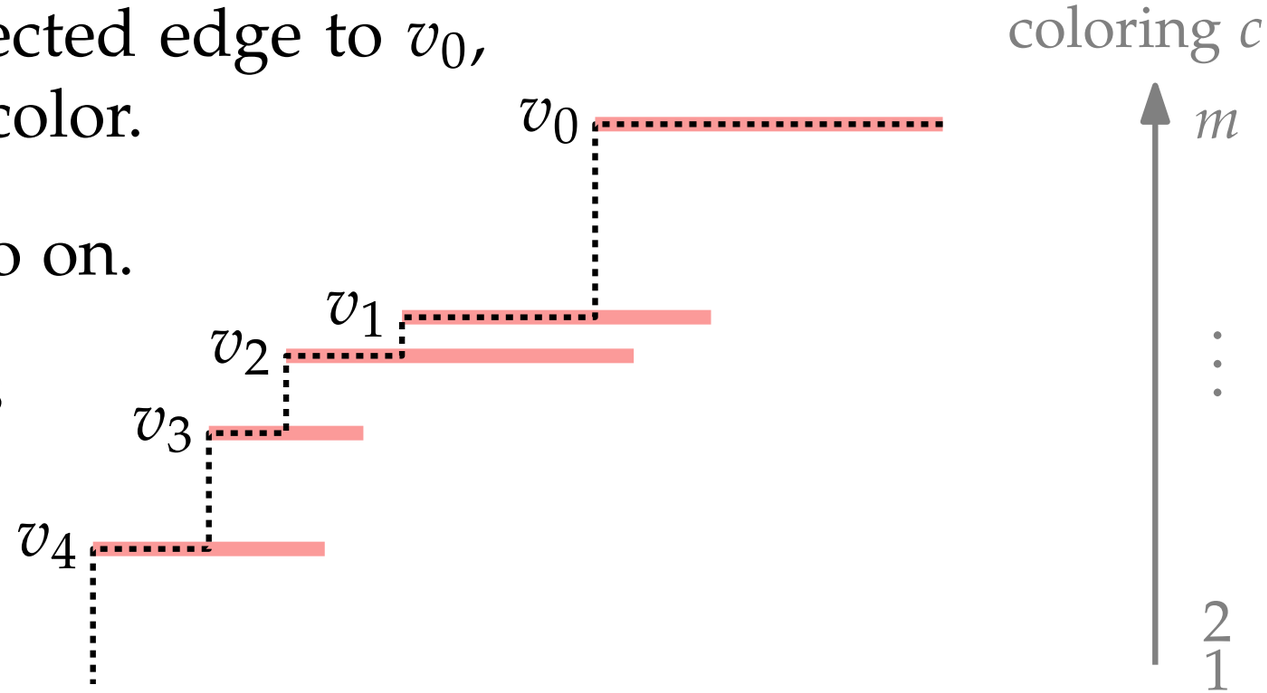
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.



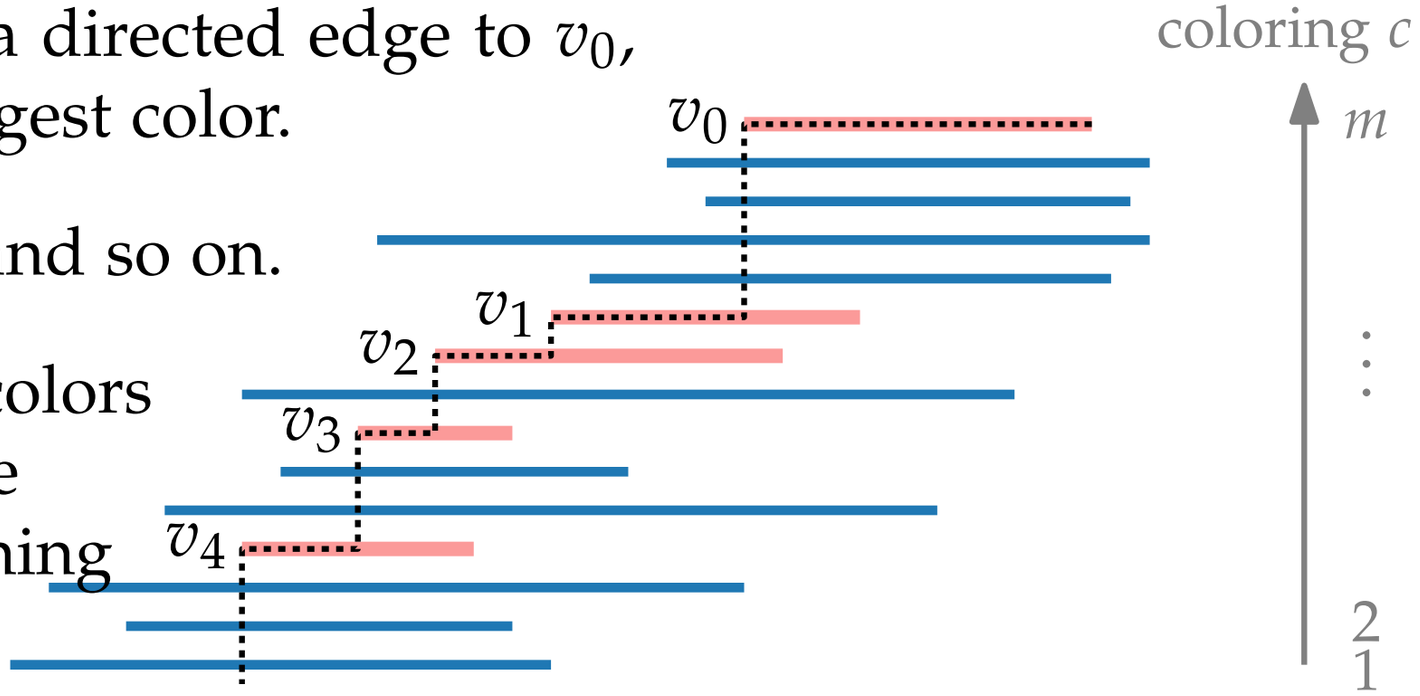
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.



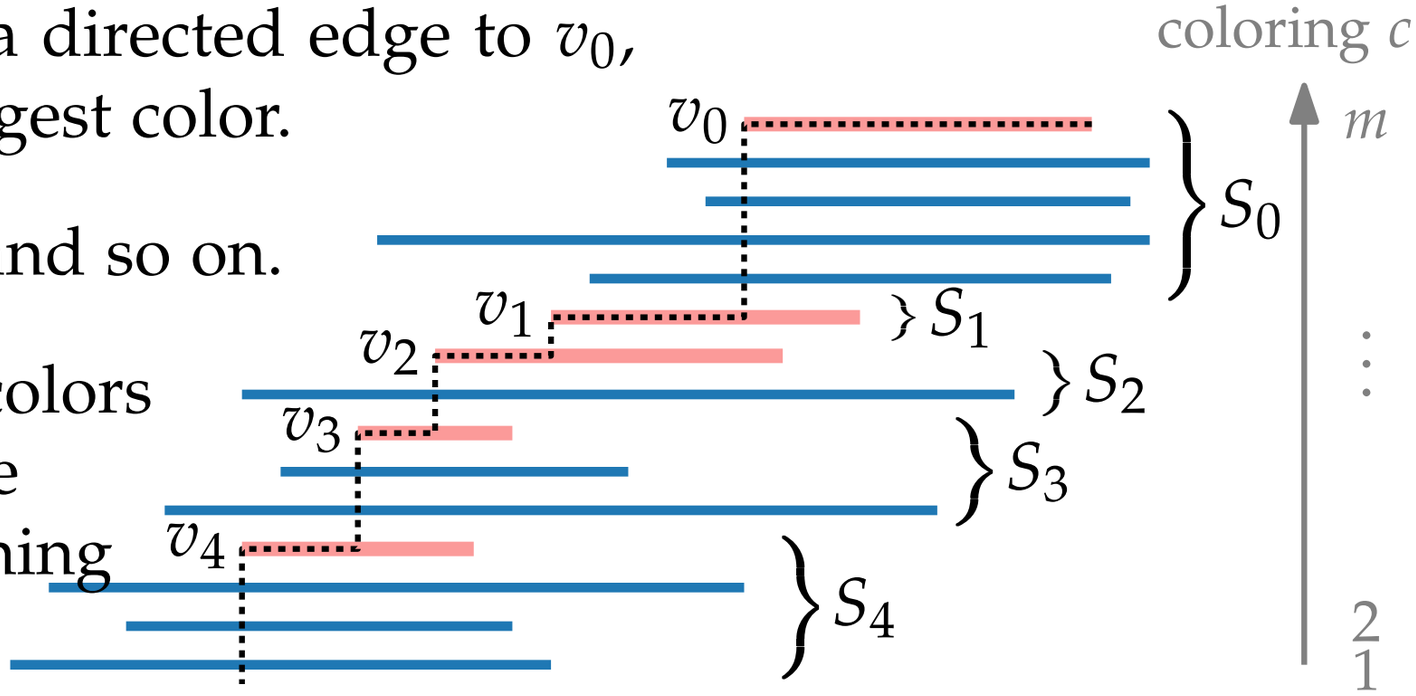
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.
- Among all intervals having a directed edge to v_0, let v_1 be the one with the largest color.
- Similarly, define v_2 w.r.t. v_1 and so on.
- By the greedy strategy, the colors between $c(v_i)$ and $c(v_{i+1})$ are occupied by intervals containing the left endpoint of v_i.



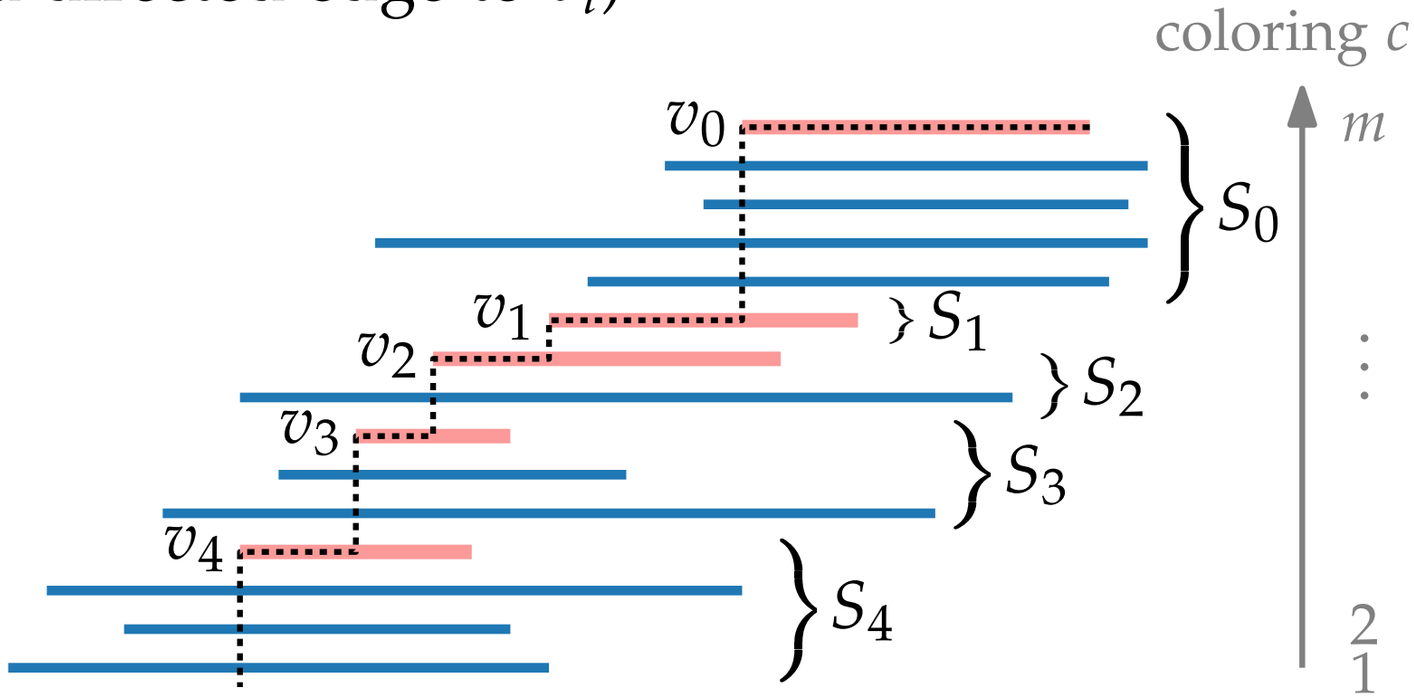
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)



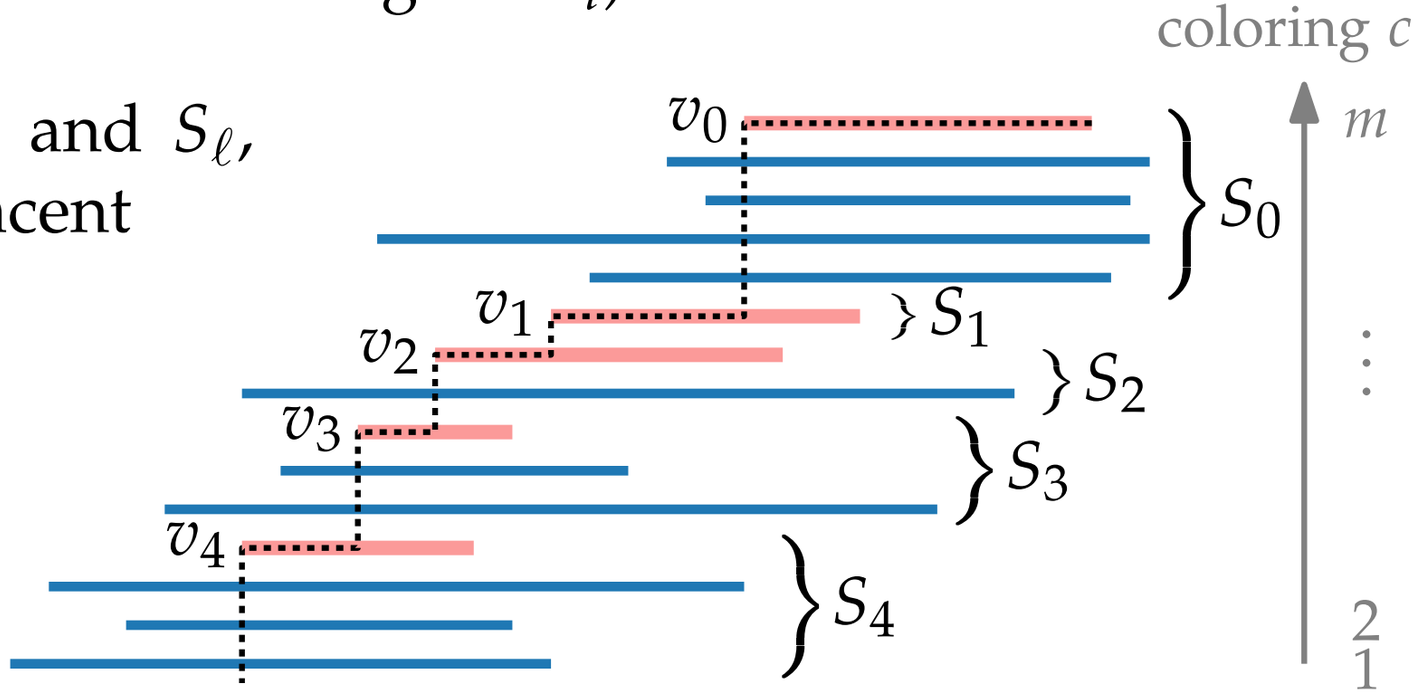
Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i. (otherwise they would have a directed edge to v_i)
- **Claim:** for any two steps S_i and S_ℓ, every pair of intervals is adjacent in the transitive closure G^+.



Coloring Directional Interval Graphs

Theorem 1:

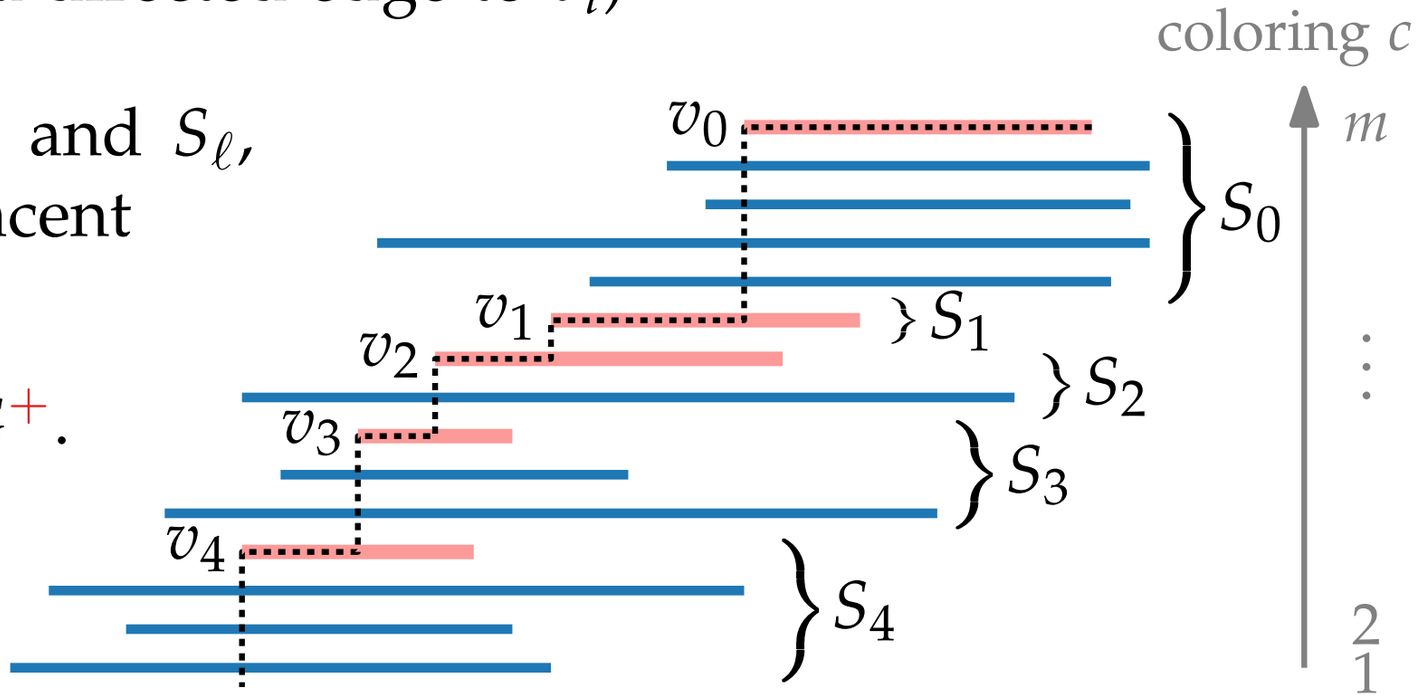
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)

- **Claim:** for any two steps S_i and S_ℓ, every pair of intervals is adjacent in the transitive closure G^+.

  $\Rightarrow$ $S = \bigcup S_i$ is a clique in G^+.



Coloring Directional Interval Graphs

Theorem 1:

A coloring c computed by `GreedyColoring` has the minimum number of colors.

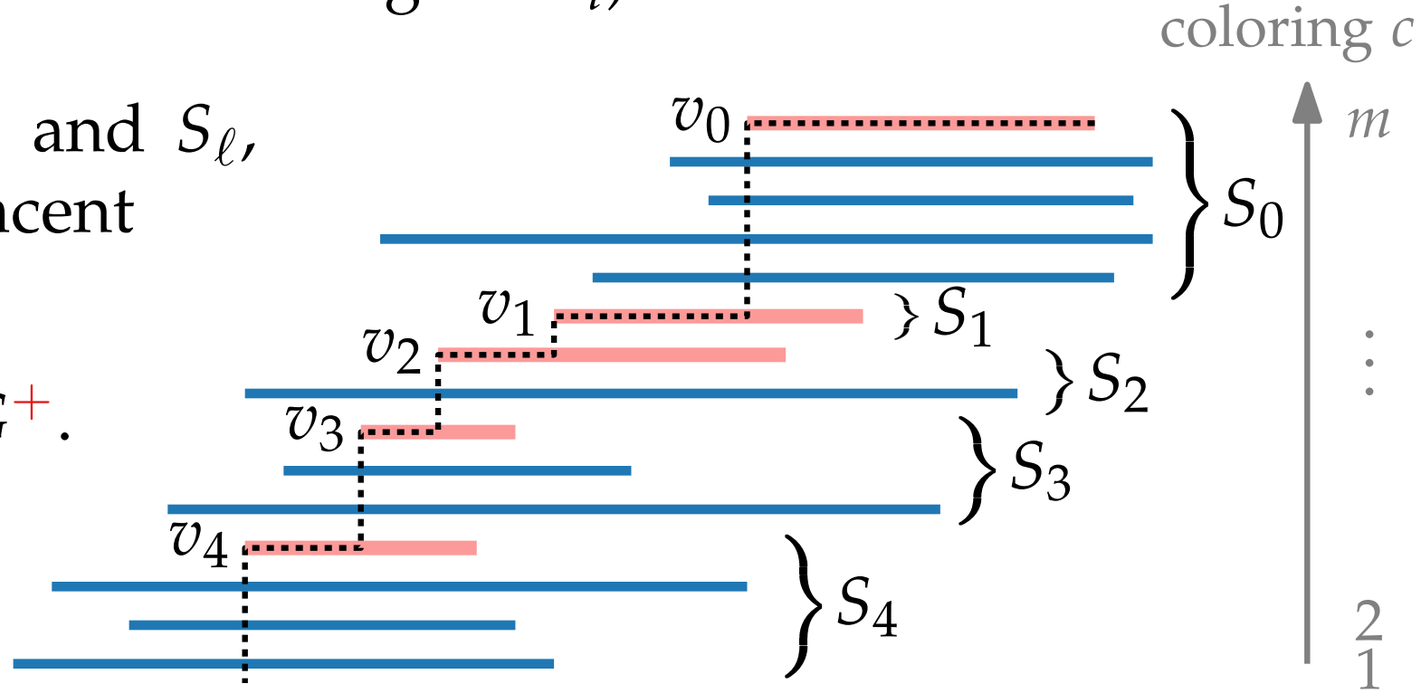
Proof sketch:

- Hence, for every *step* S_i, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)

- **Claim:** for any two steps S_i and S_ℓ, every pair of intervals is adjacent in the transitive closure G^+.

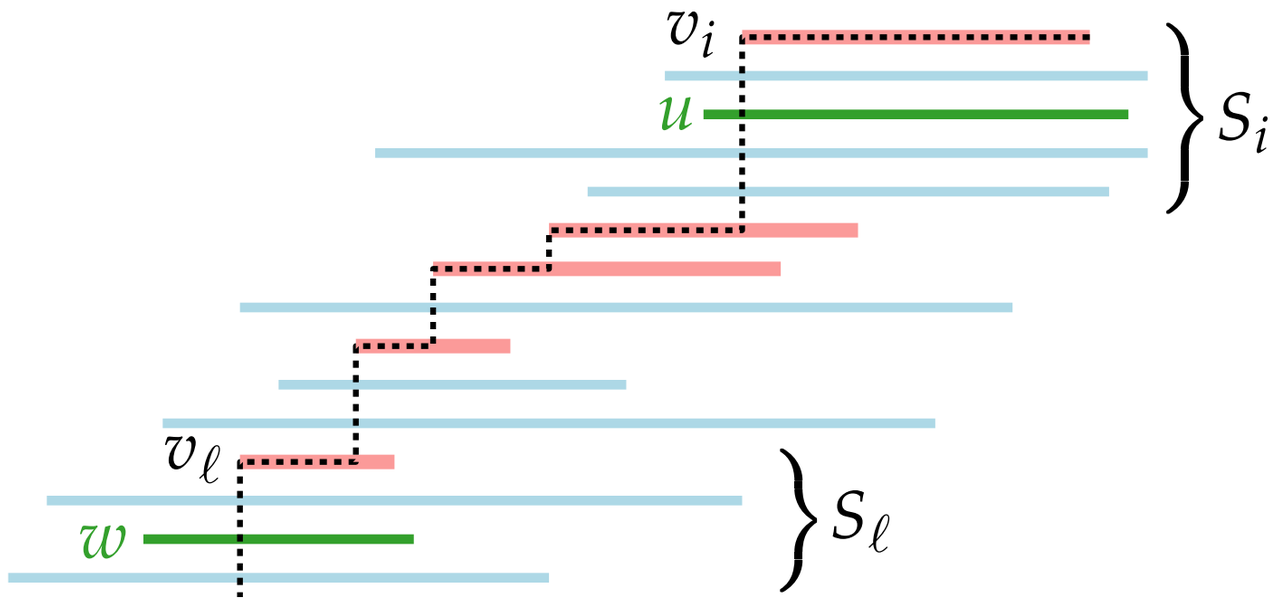
  $\Rightarrow$ $S = \bigcup S_i$ is a clique in G^+.

  $\Rightarrow$ S alone requires m colors in G. $\square$



Proof of the Claim

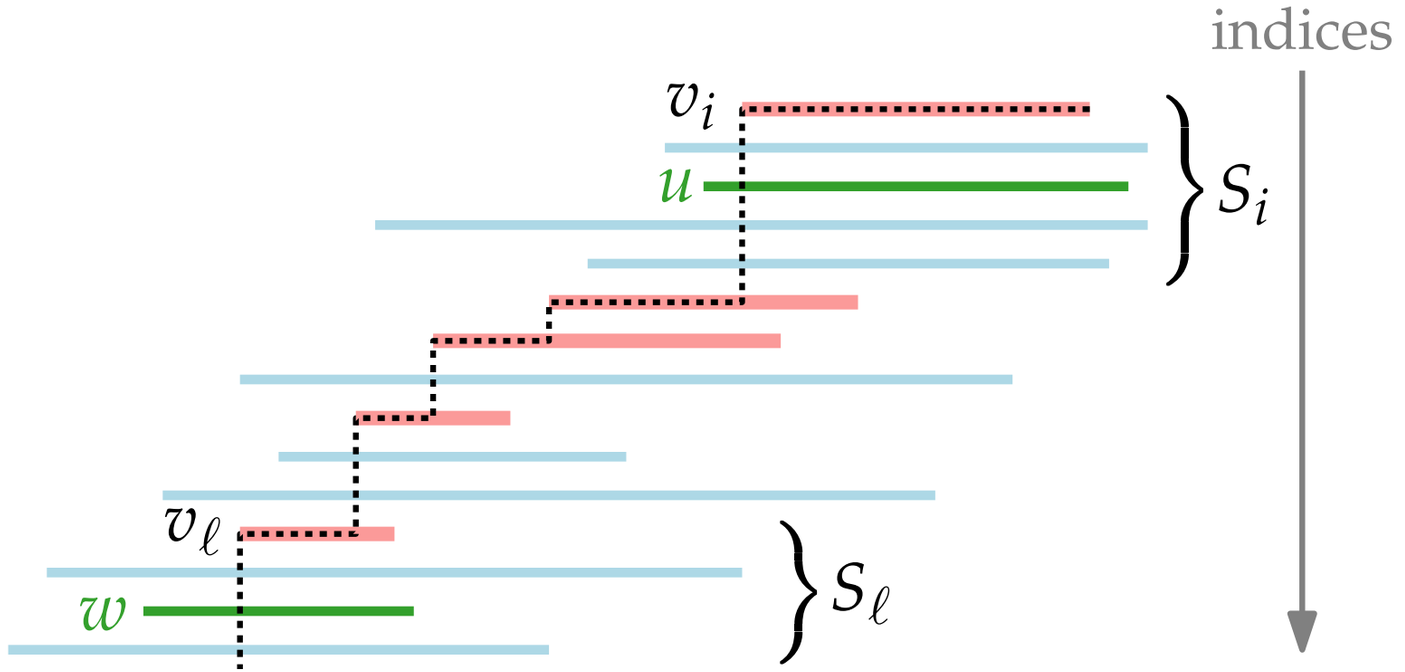
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

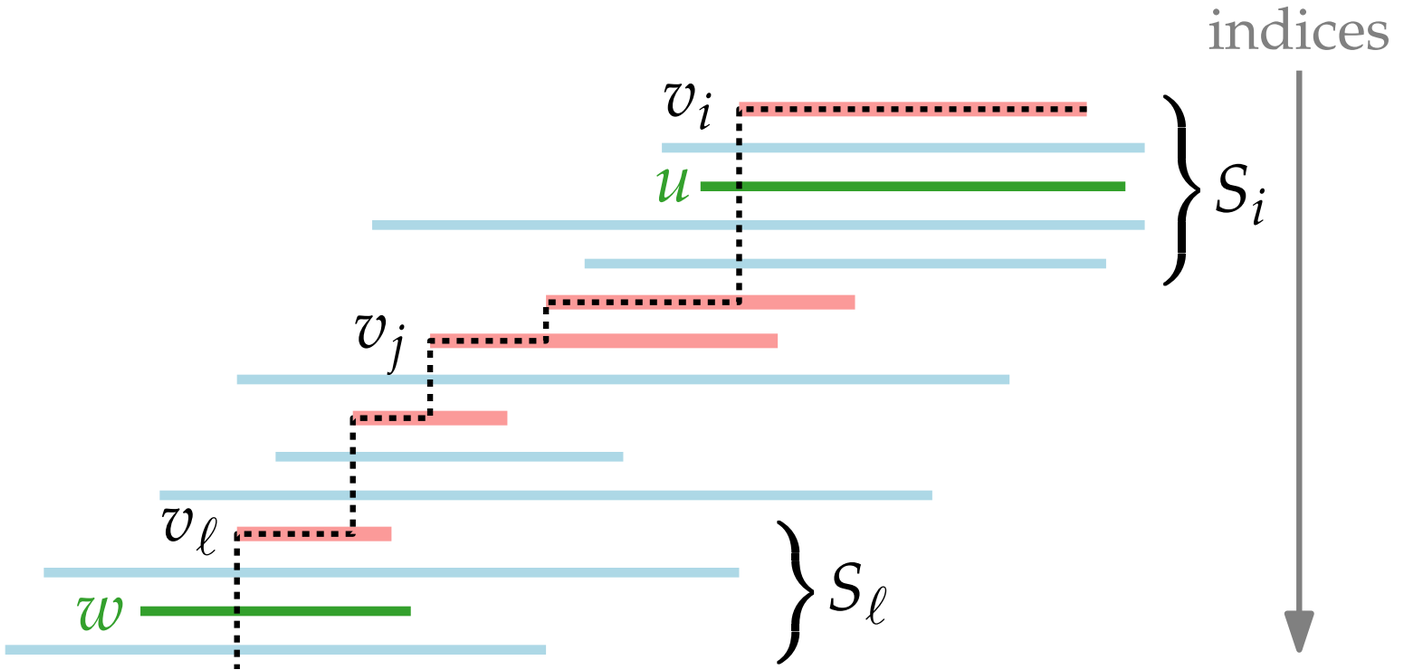


Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.

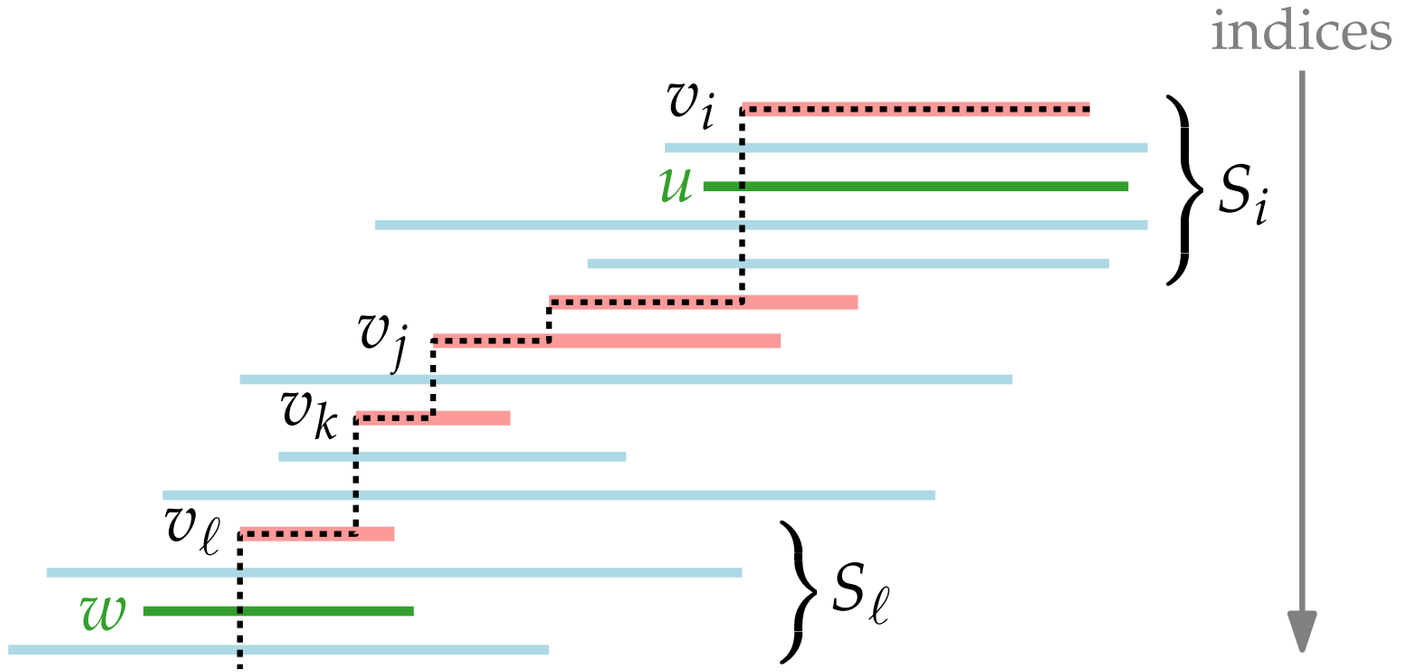


Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.



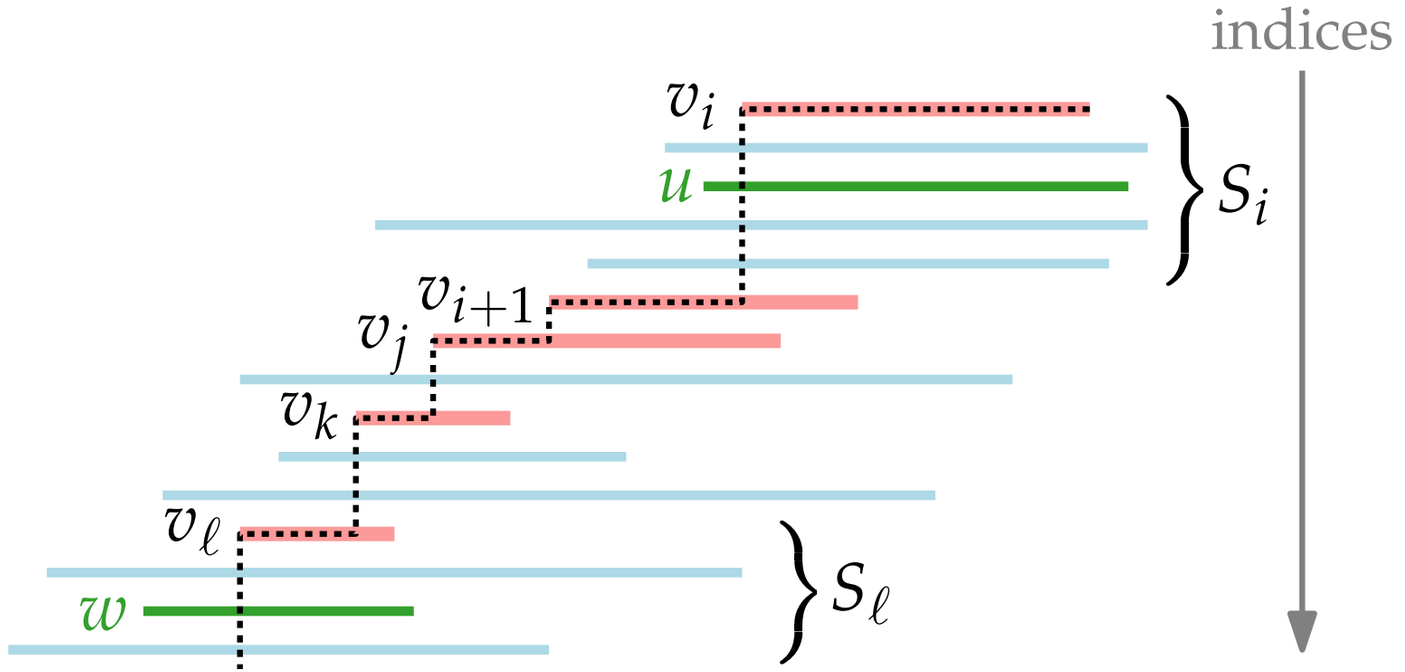
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$u \cap v_{i+1} \neq \emptyset$



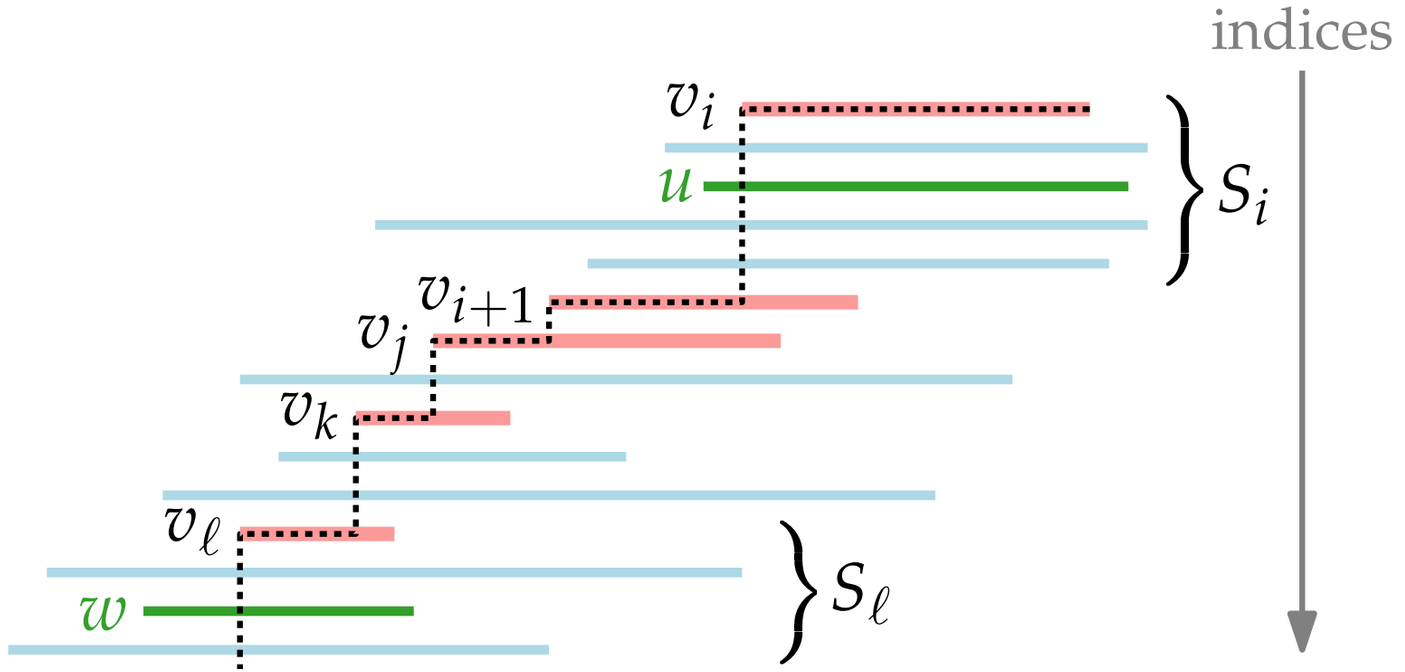
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$u \cap v_{i+1} \neq \emptyset$
$w \cap v_{\ell-1} \neq \emptyset$



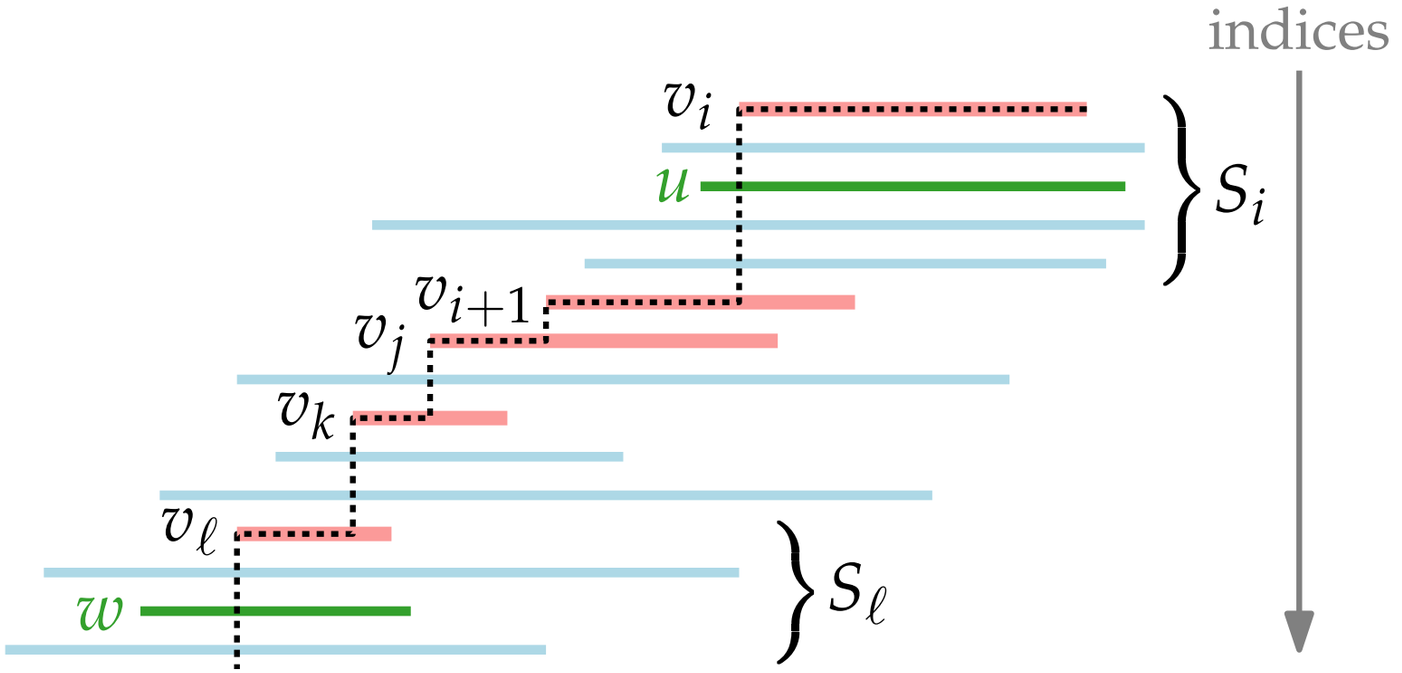
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \Rightarrow u \cap w = \emptyset$$



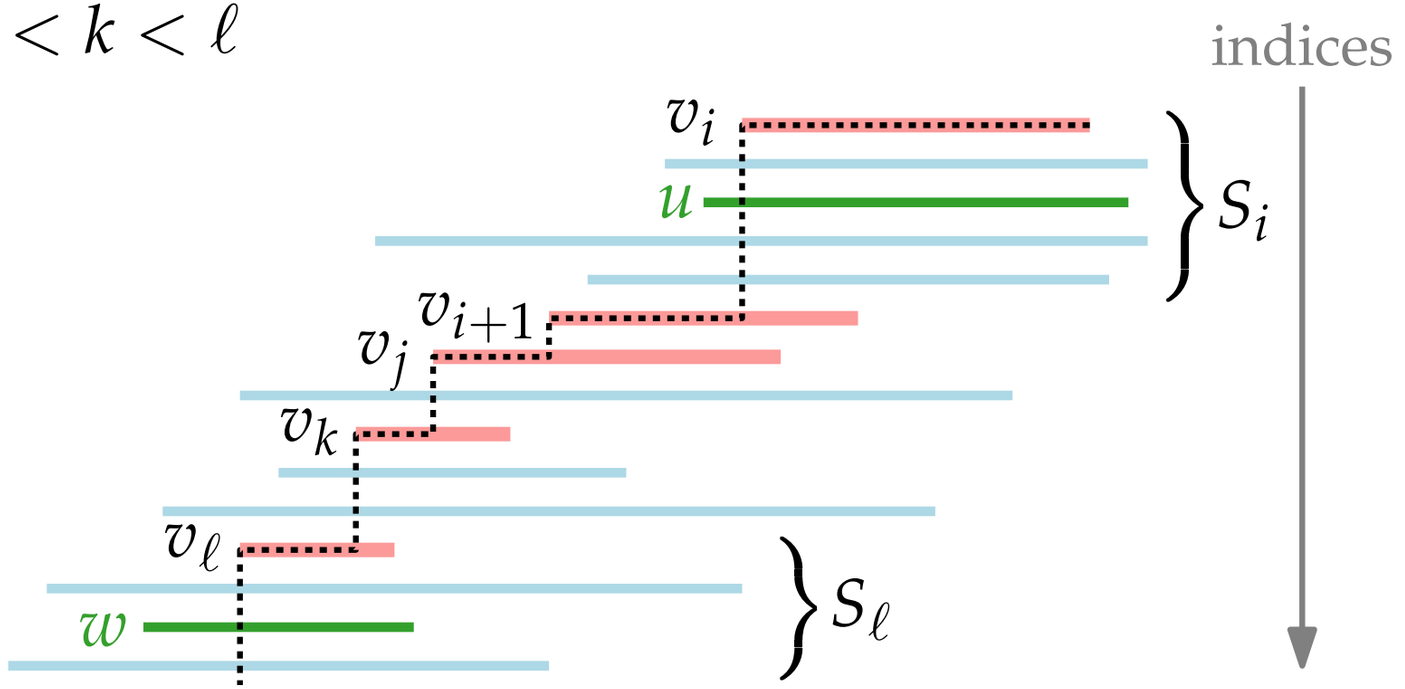
Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$



Proof of the Claim

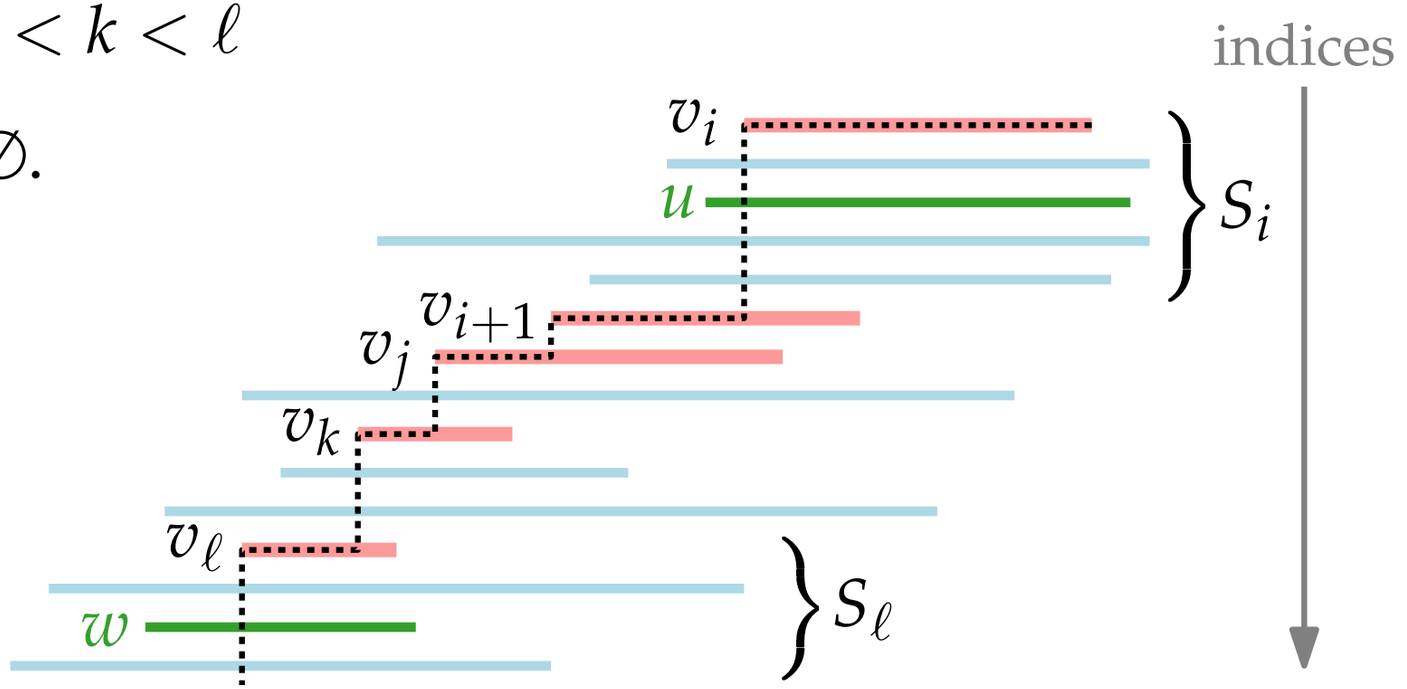
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.



Proof of the Claim

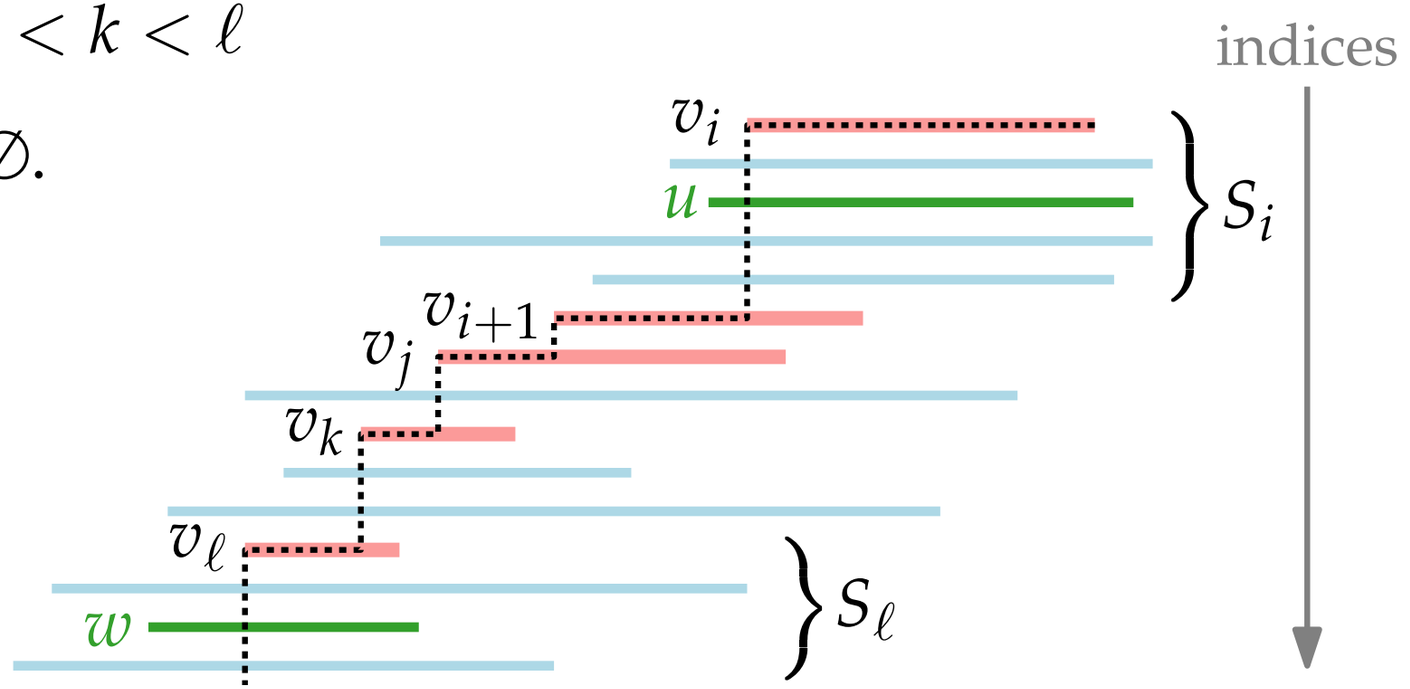
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap



Proof of the Claim

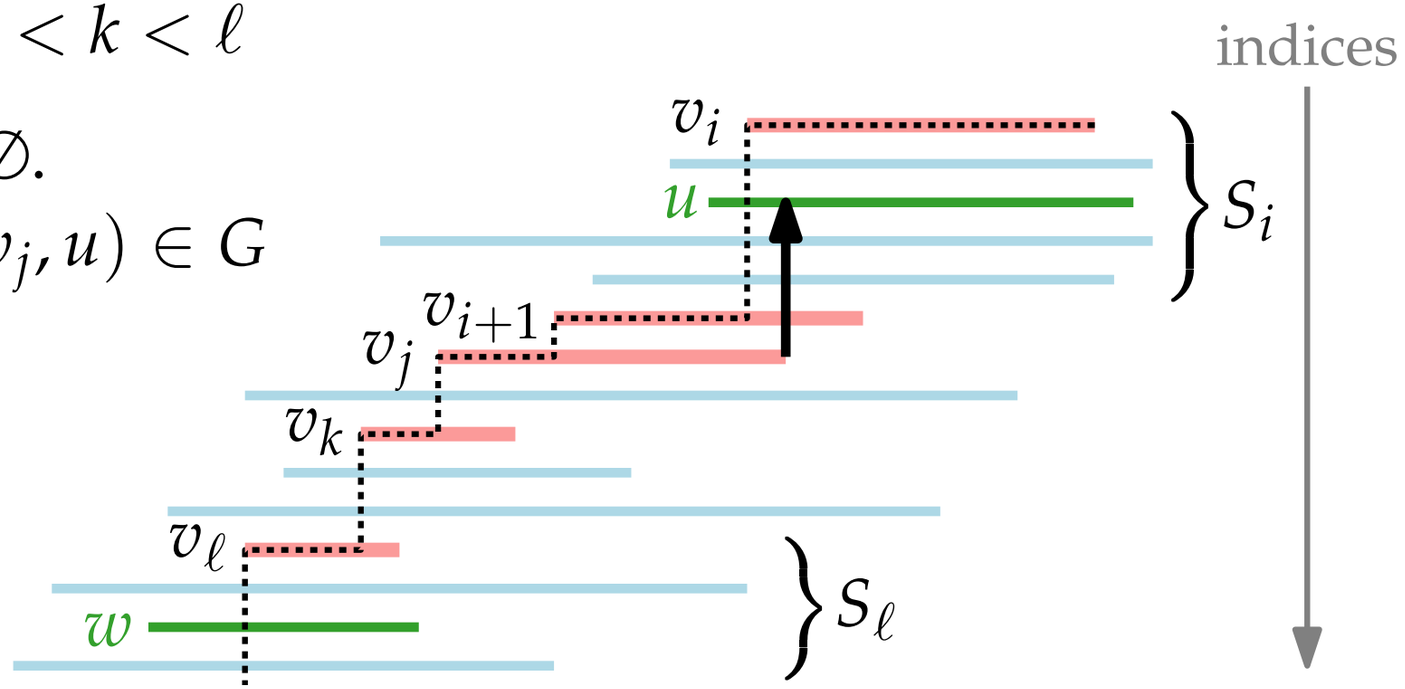
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap $\Rightarrow (v_j, u) \in G$



Proof of the Claim

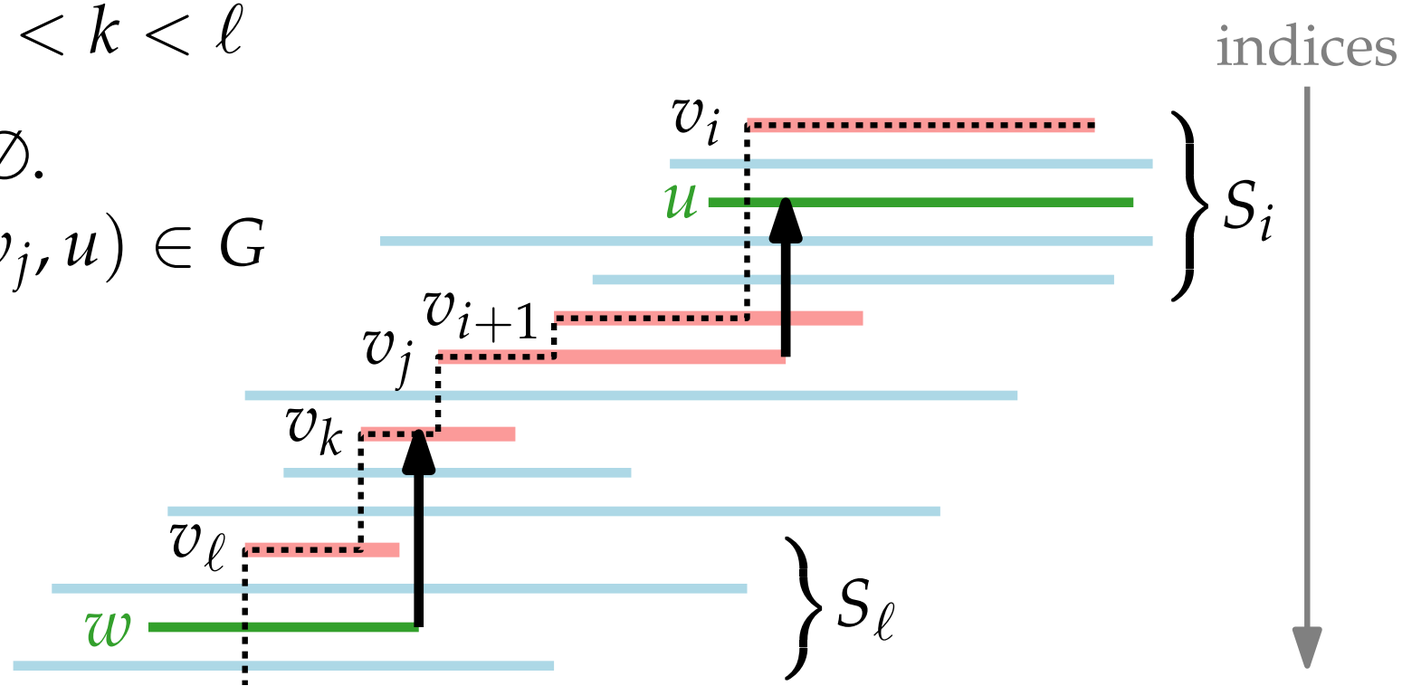
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

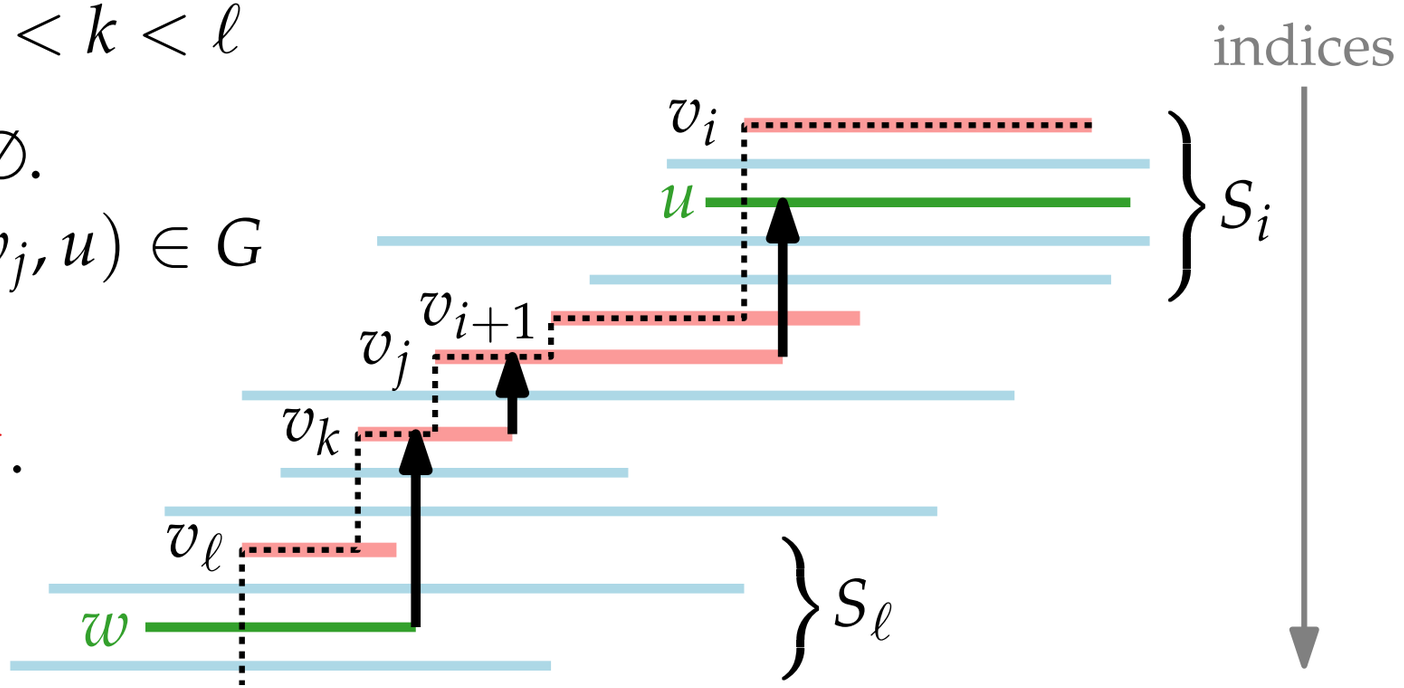
Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

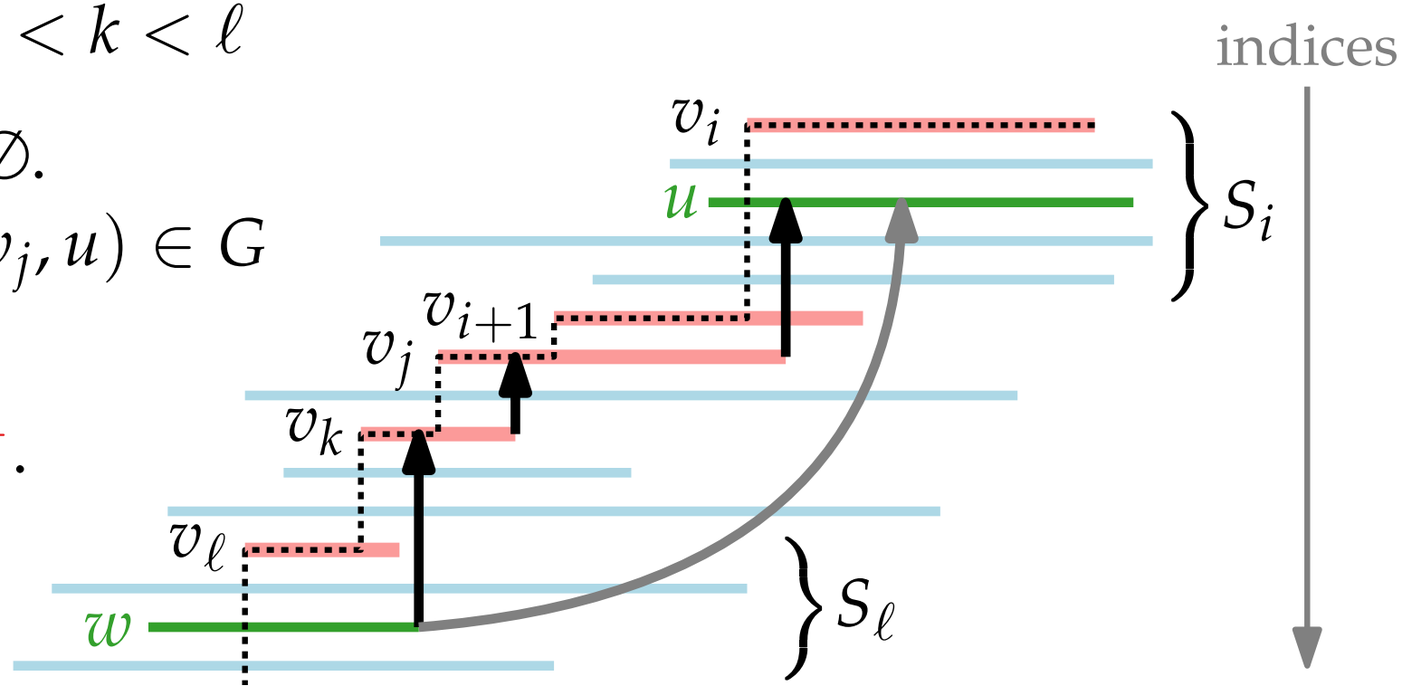
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.

Transitivity $\Rightarrow$ claim.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

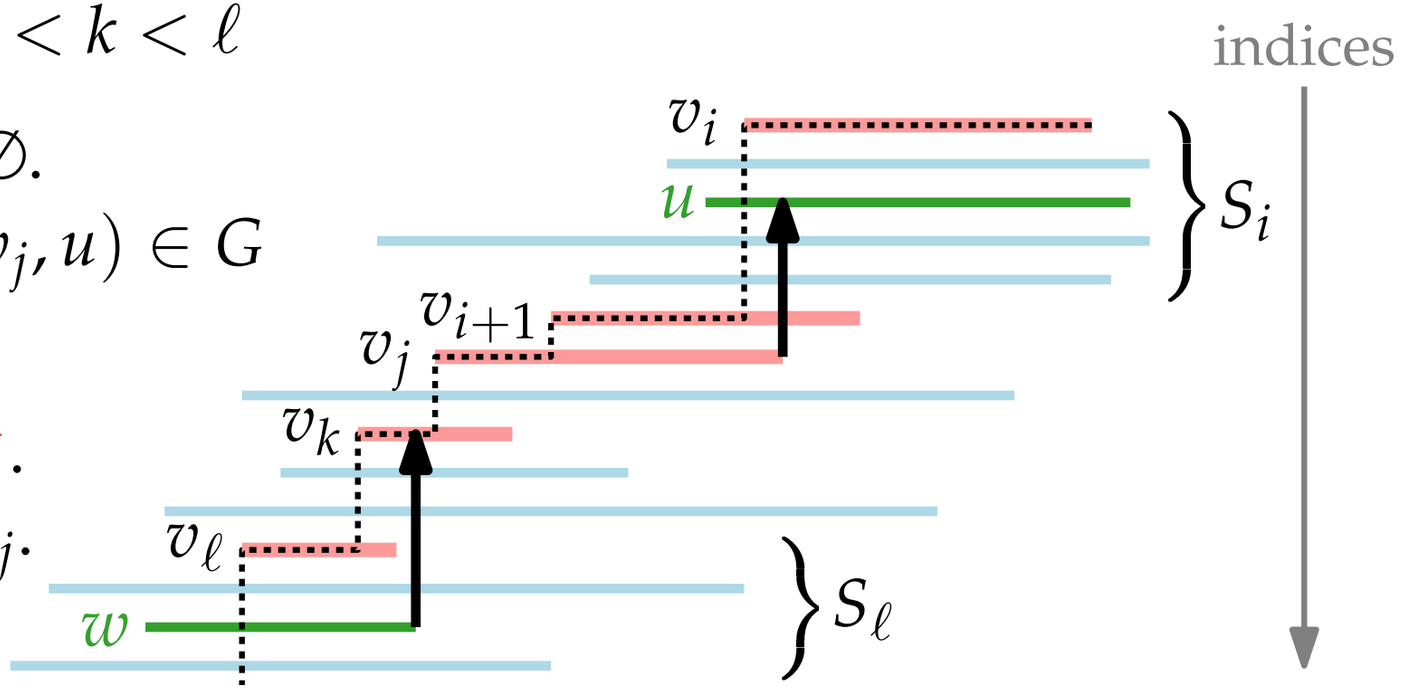
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.
If $j \geq k$, then w overlaps v_j.

Transitivity $\Rightarrow$ claim.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

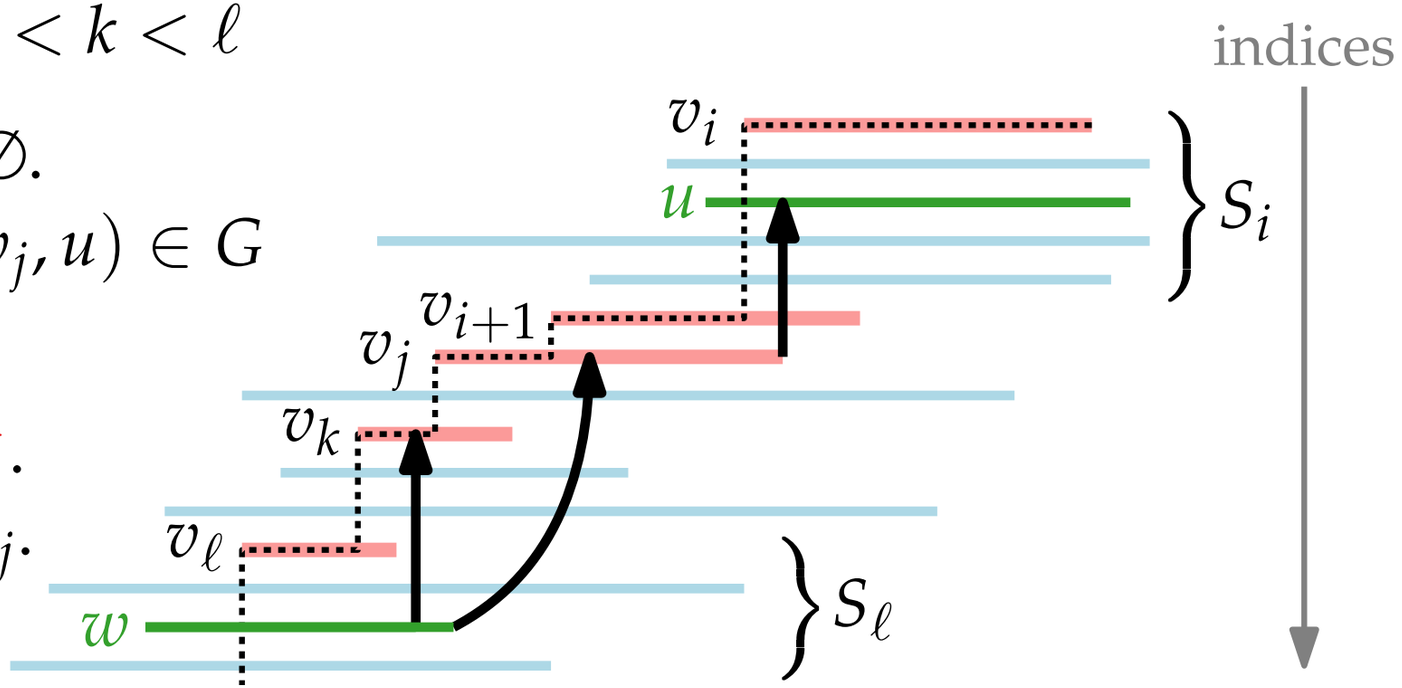
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.
If $j \geq k$, then w overlaps v_j.

Transitivity $\Rightarrow$ claim.



Proof of the Claim

Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

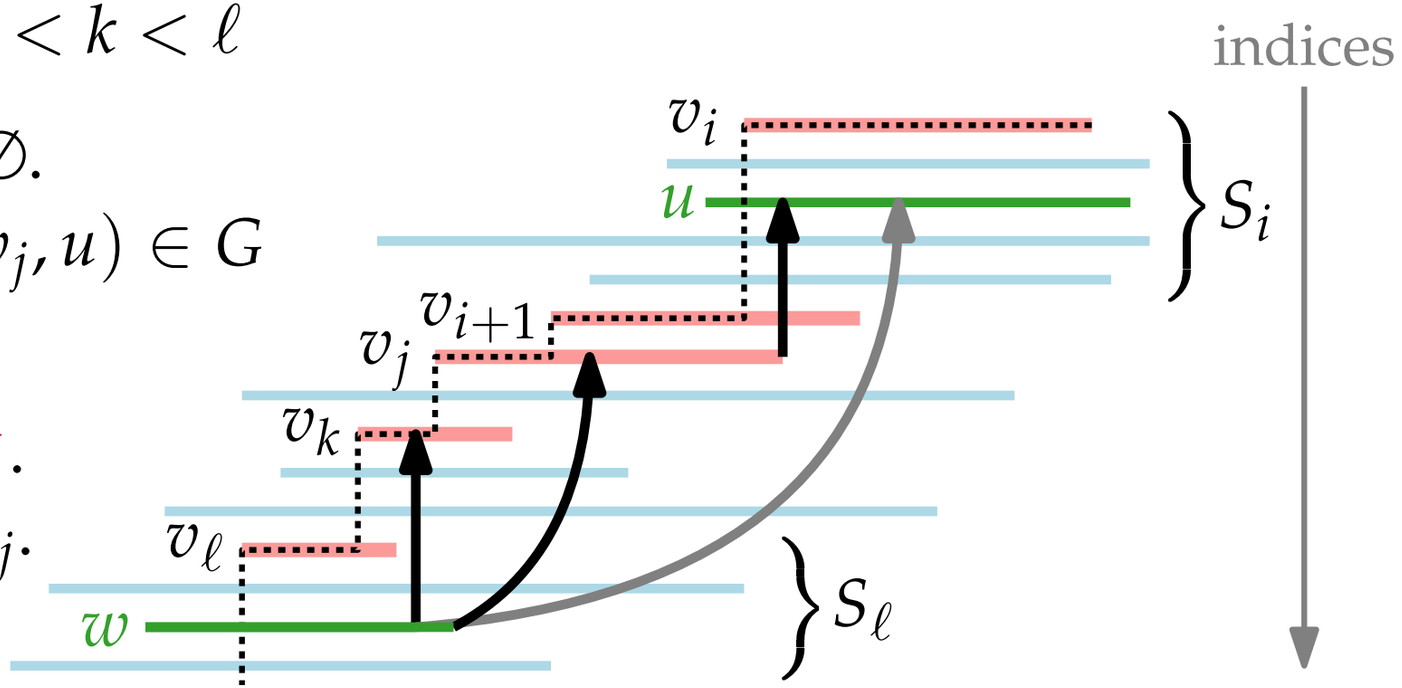
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\left.\begin{array}{l} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array}\right\} \underset{u \cap w = \emptyset}{\Rightarrow} \begin{array}{l} i < j < \ell \\ i < k < \ell \end{array}$$

By definition, $u \cap v_{j+1} = \emptyset$.
$\Rightarrow u$ and v_j overlap $\Rightarrow (v_j, u) \in G$
Similarly, $(w, v_k) \in G$.

If $j < k$, then $(v_k, v_j) \in G^+$.
If $j \geq k$, then w overlaps v_j.

Transitivity $\Rightarrow$ claim.



Overview

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- undirected edge uv: $c(u) \neq c(v)$,
- directed edge uv: $c(u) < c(v)$,
- $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs: **our contribution**

- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

Mixed interval graphs:

- coloring is NP-complete

Directed graphs (only directed edges):

- coloring in linear time using topological sorting

$n :=$ # intervals

Overview

Find a graph coloring $c\colon V \to \mathbb{N}$ such that:
[Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97]

- undirected edge uv: $c(u) \neq c(v)$,
- directed edge uv: $c(u) < c(v)$,
- $\max_{v \in V} c(v)$ is minimized.

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs: **our contribution**

- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

Mixed interval graphs:

- coloring is NP-complete

Directed graphs (only directed edges):

- coloring in linear time using topological sorting

$n :=$ # intervals

Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

variable gadget for each variable v_i:

Coloring Mixed Interval Graphs

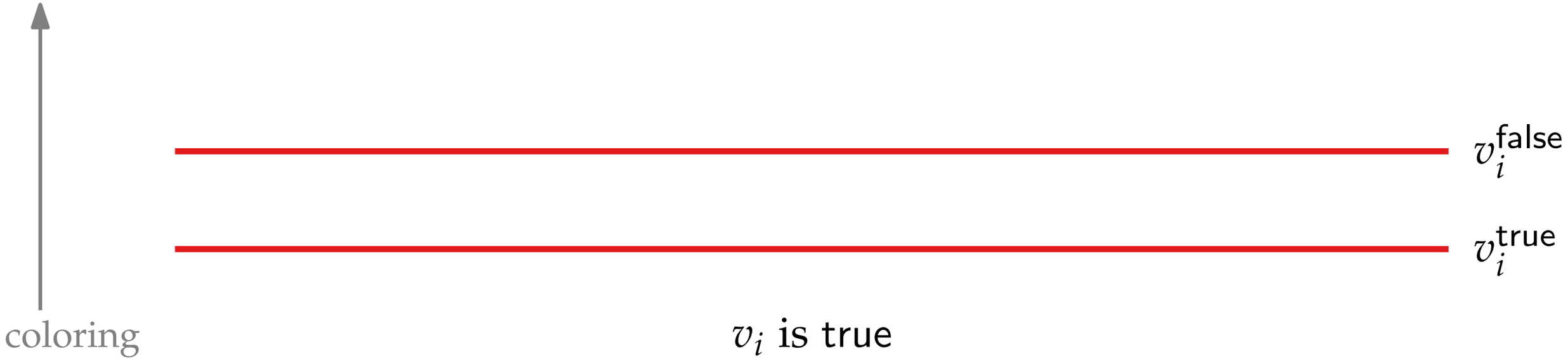
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

variable gadget for each variable v_i:



Coloring Mixed Interval Graphs

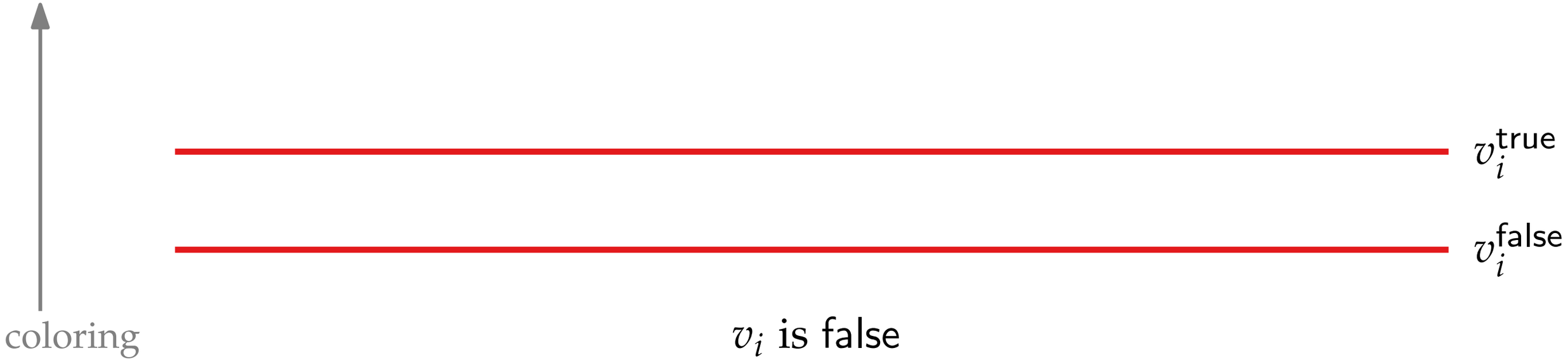
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

variable gadget for each variable v_i:



Coloring Mixed Interval Graphs

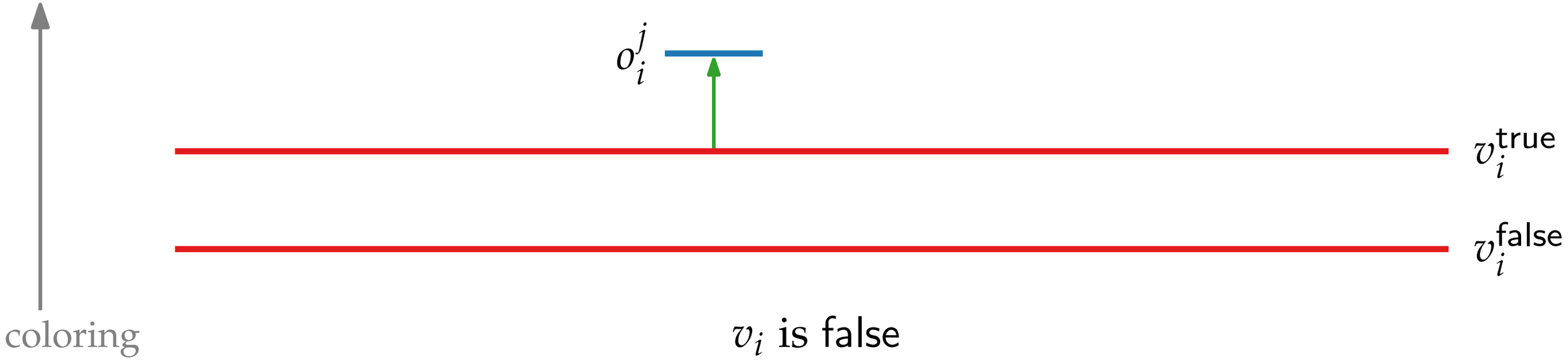
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

clause c_j containing literal v_i:



Coloring Mixed Interval Graphs

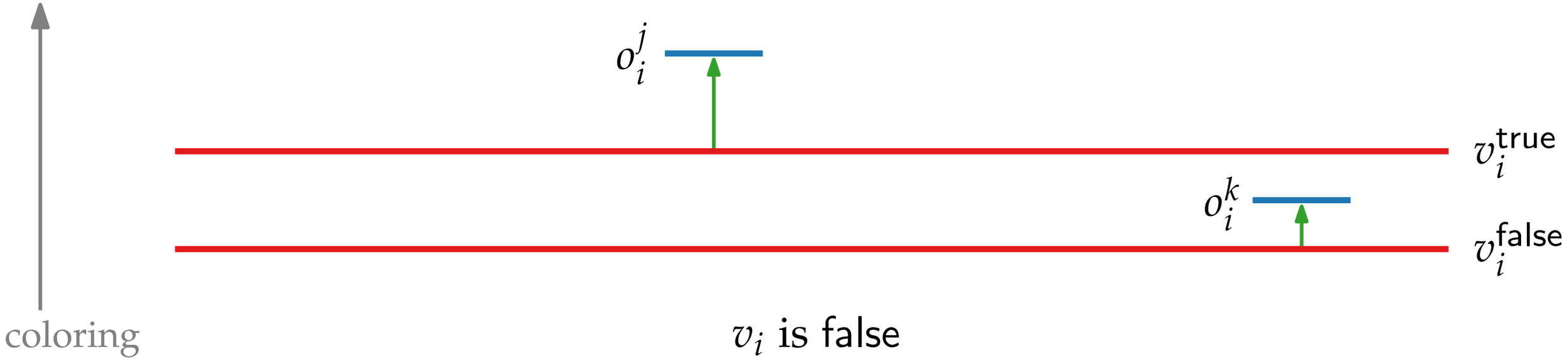
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

clause c_j containing literal v_i: clause c_k containing literal $\neg v_i$:



Coloring Mixed Interval Graphs

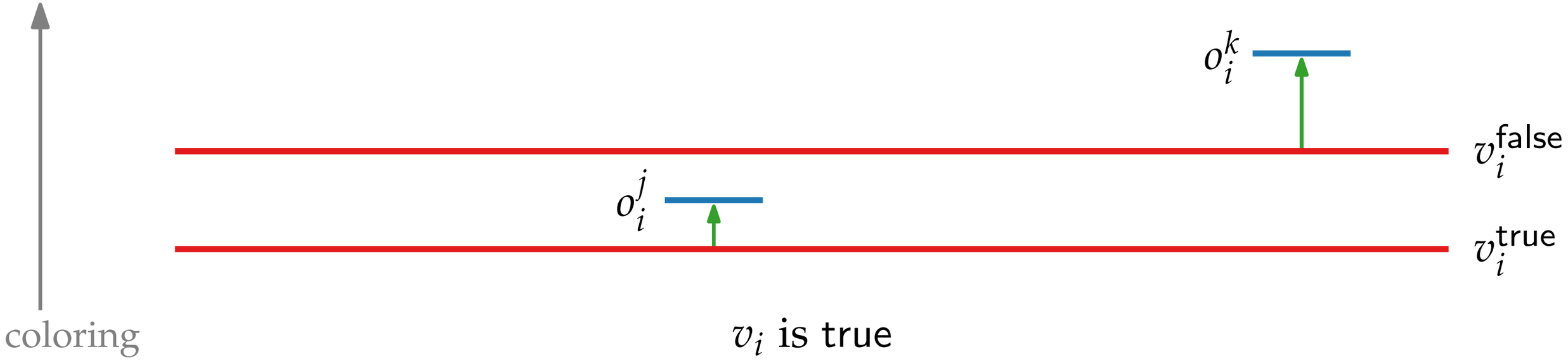
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

clause c_j containing literal v_i: clause c_k containing literal $\neg v_i$:



Coloring Mixed Interval Graphs

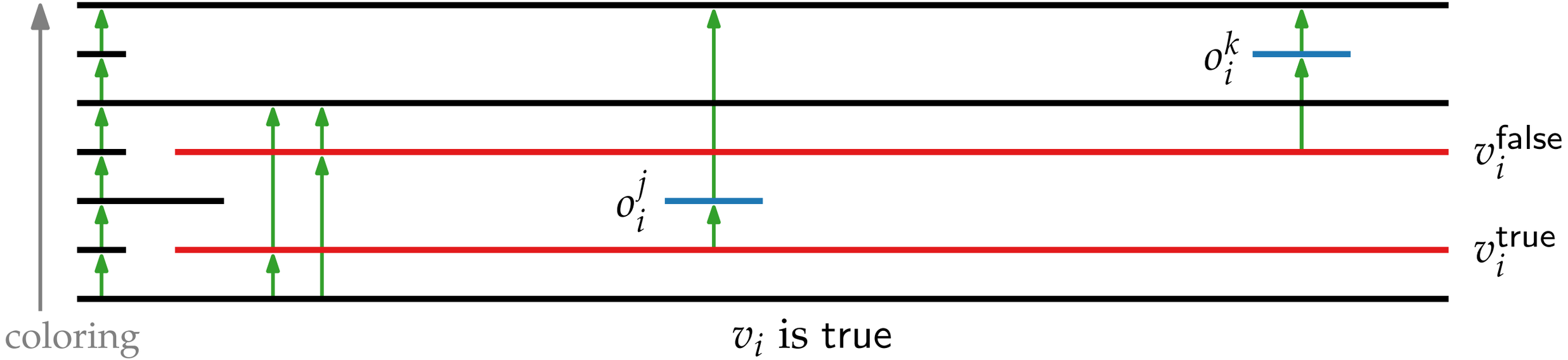
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

fix positions by adding "frame" intervals



Coloring Mixed Interval Graphs

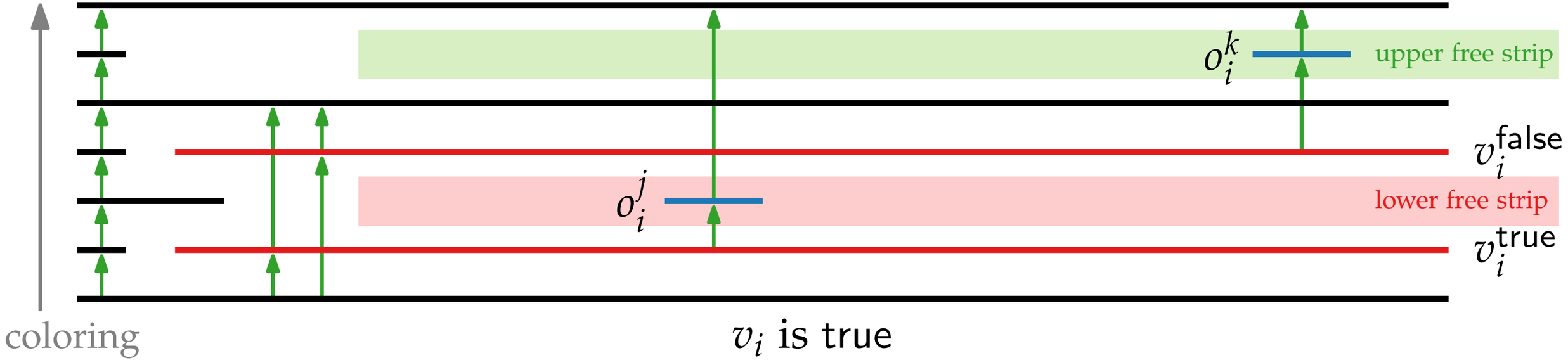
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

We model an instance Φ of 3-SAT as a mixed interval graph G_Φ.

fix positions by adding "frame" intervals



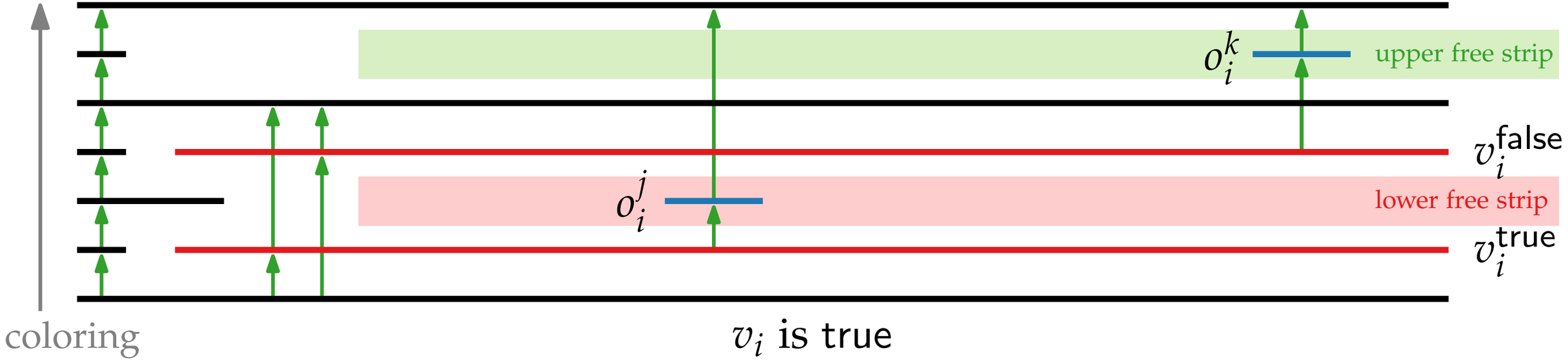
Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:



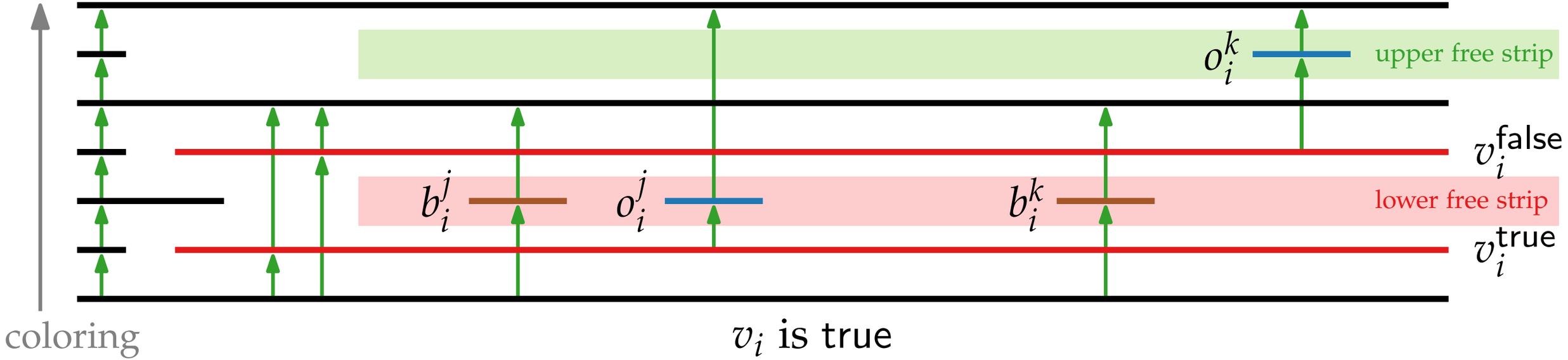
Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:



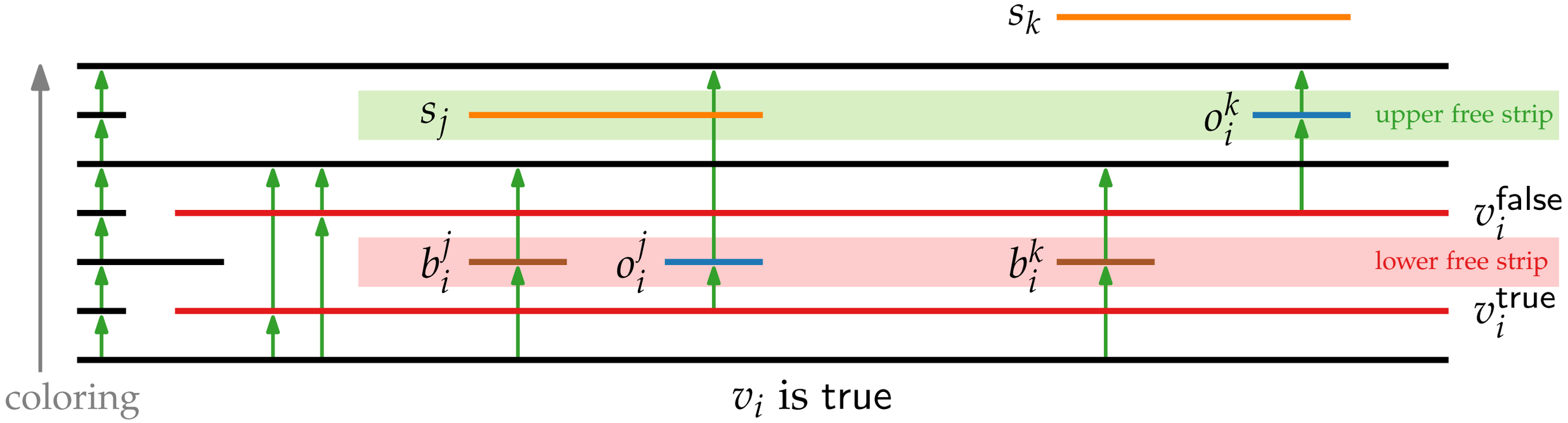
Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:



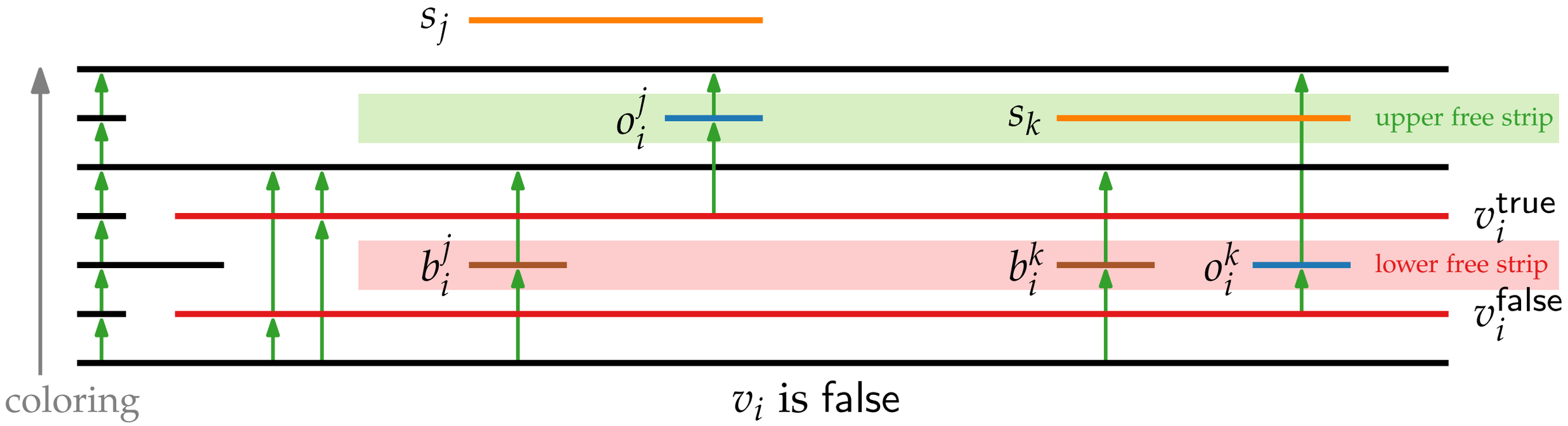
Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:



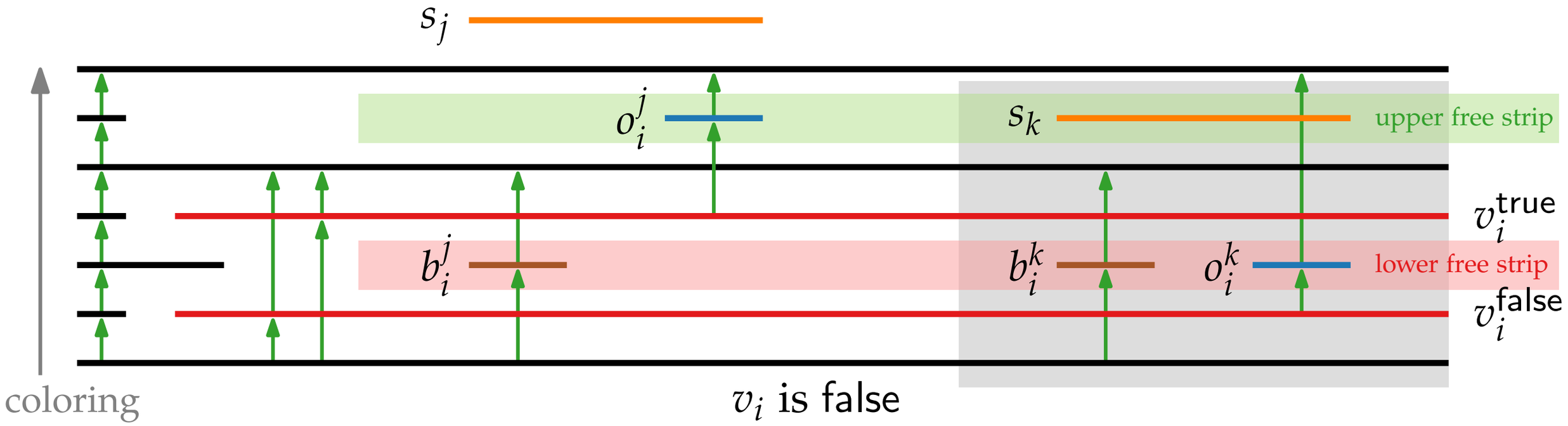
Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:



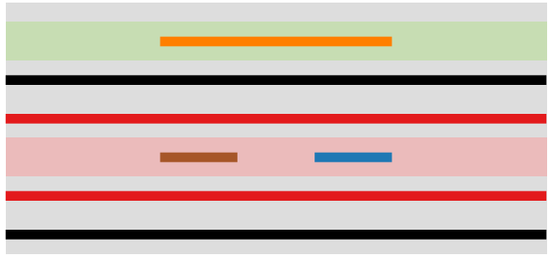
Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:



Coloring Mixed Interval Graphs

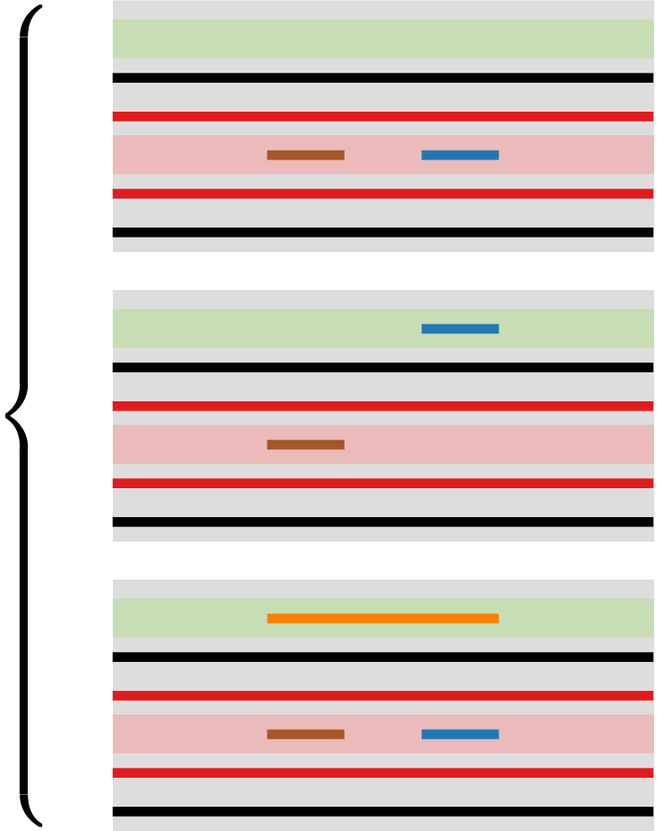
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:

$6n$ colors
(n := # variables)



Coloring Mixed Interval Graphs

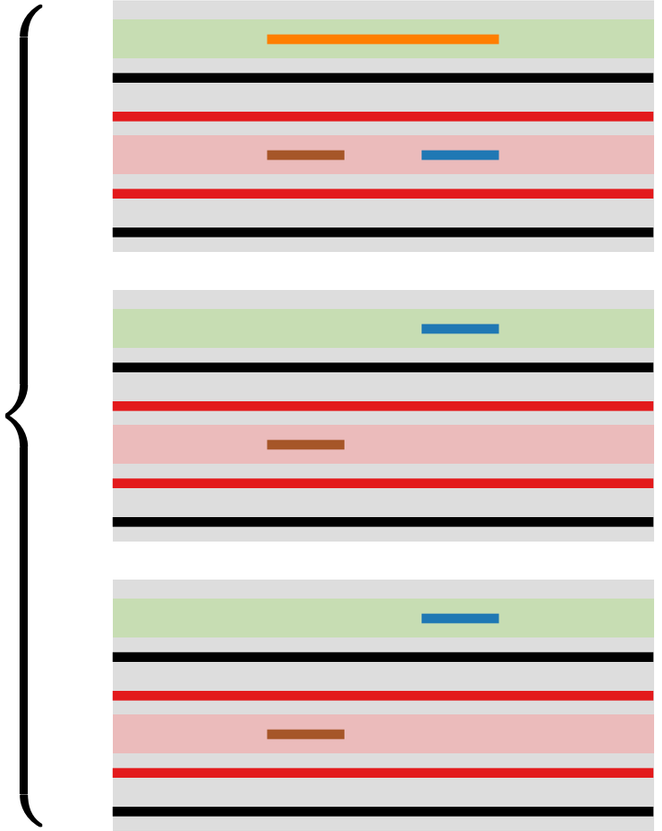
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:

$6n$ colors
(n := # variables)



Coloring Mixed Interval Graphs

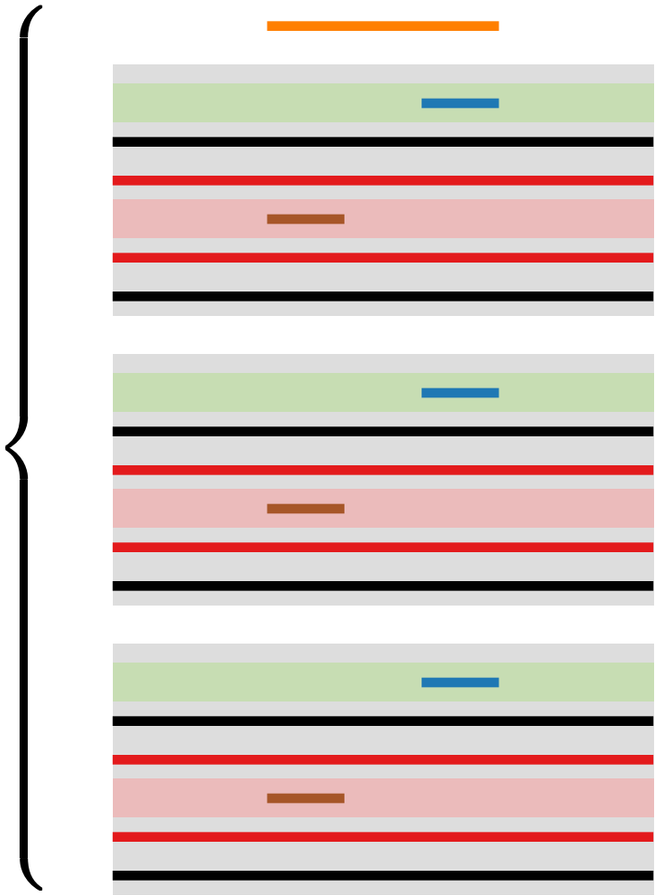
Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

Clause gadget:

$6n + 1$ colors
(n := # variables)



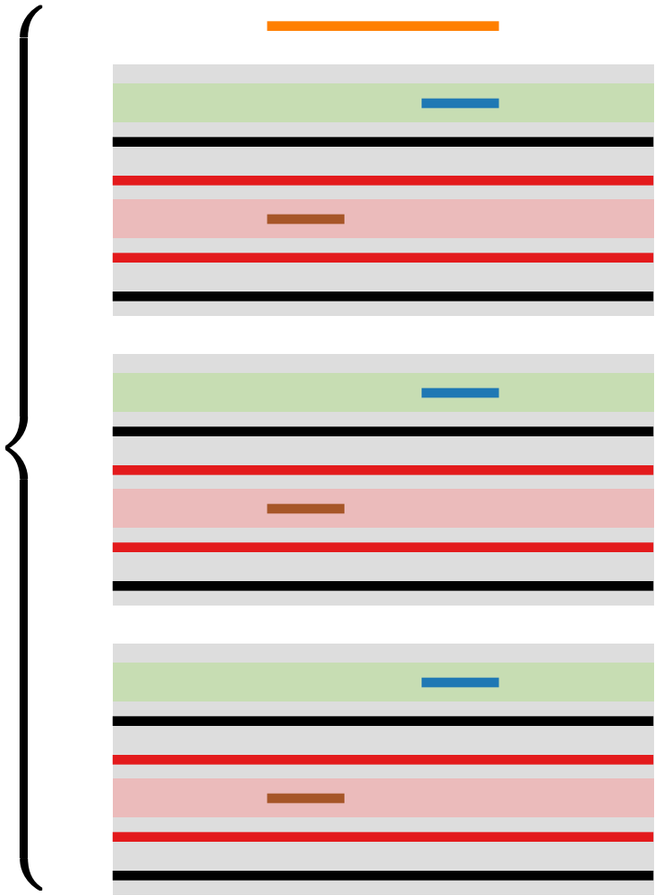
Coloring Mixed Interval Graphs

Theorem 2:

Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

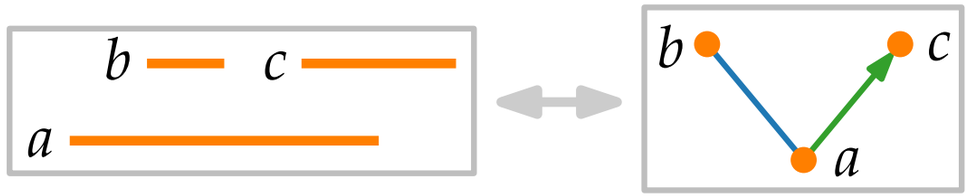
Clause gadget:



$6n + 1$ colors
(n := # variables)

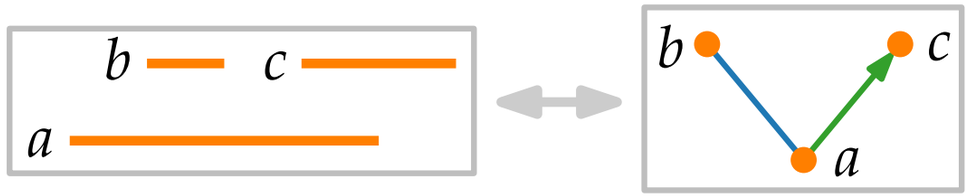
Φ is satisfiable
$\Leftrightarrow$ G_Φ admits a coloring with $6n$ colors. $\square$

Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.

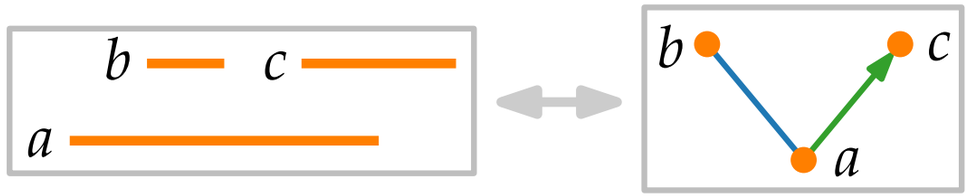
Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

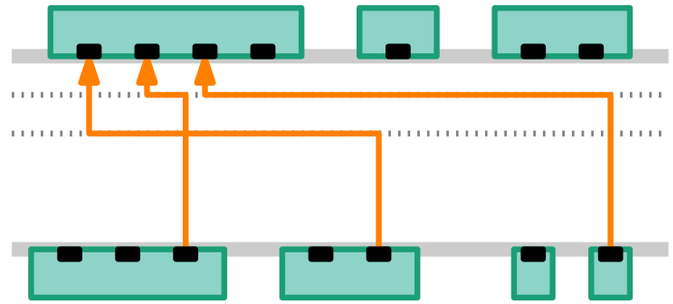
Conclusion and Open Problems



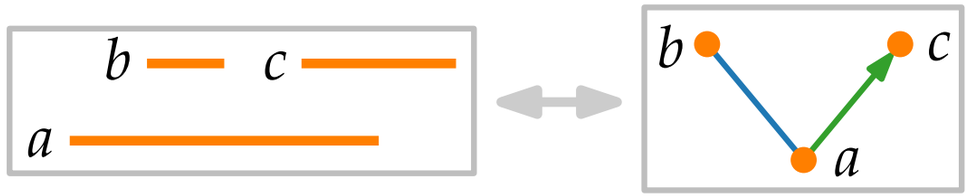
- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n := \#$ vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

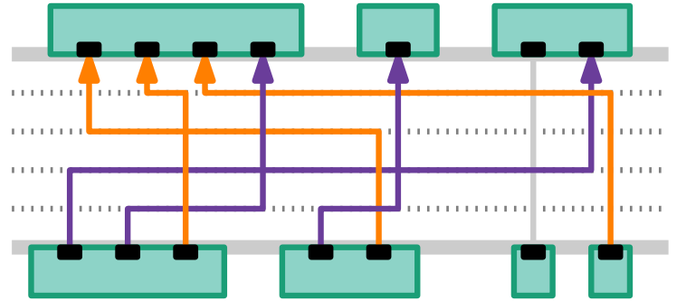


Conclusion and Open Problems

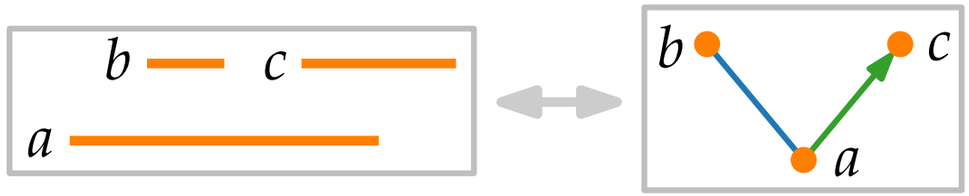


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time. $n := \#$ vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

$\Rightarrow$ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)

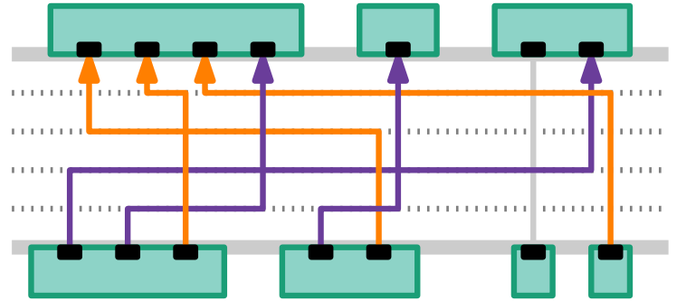


Conclusion and Open Problems



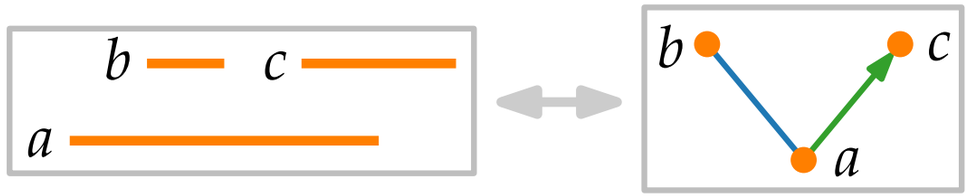
- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time. $n := \#$ vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

$\Rightarrow$ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

Conclusion and Open Problems

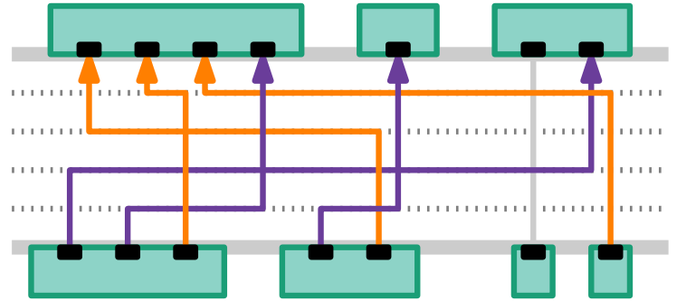


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

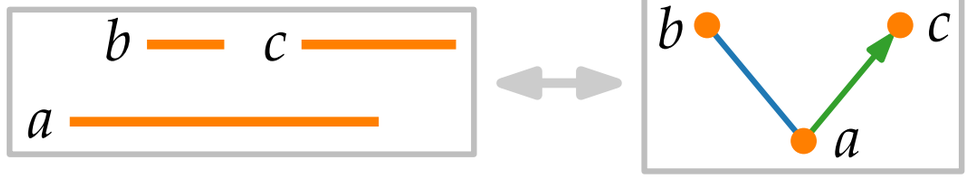
$\Rightarrow$ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.
- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems

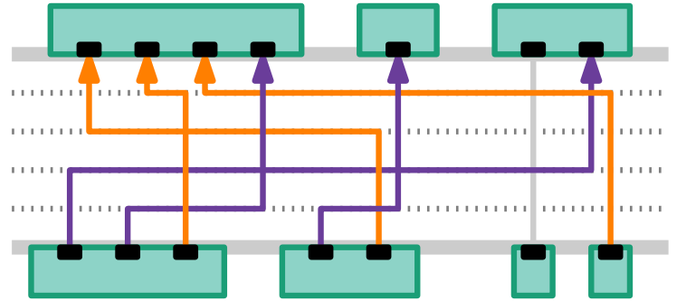


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

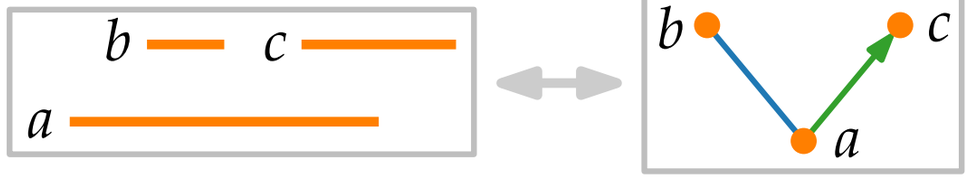
$\Rightarrow$ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.
- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems

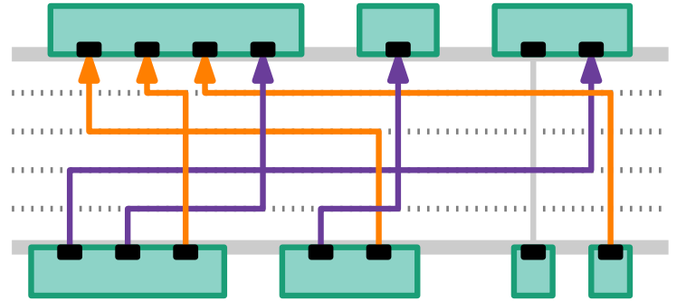


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)

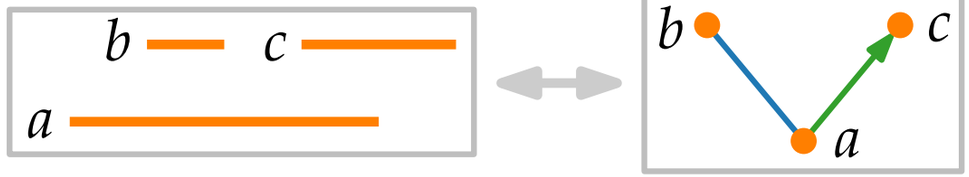


can we do better?

- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.
- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems

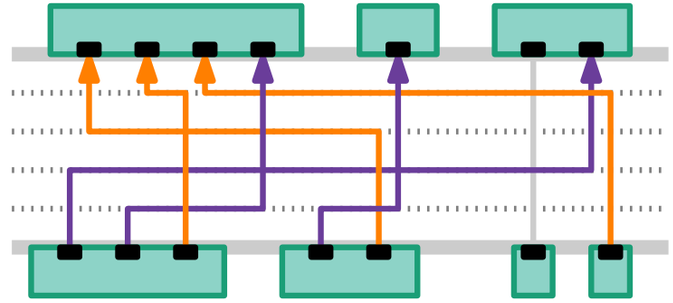


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

$\Rightarrow$ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



can we do better?

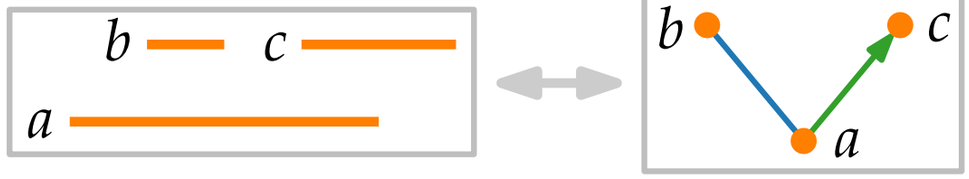
- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

bidirectional?

- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems



- We have introduced the natural concept of directional interval graphs.

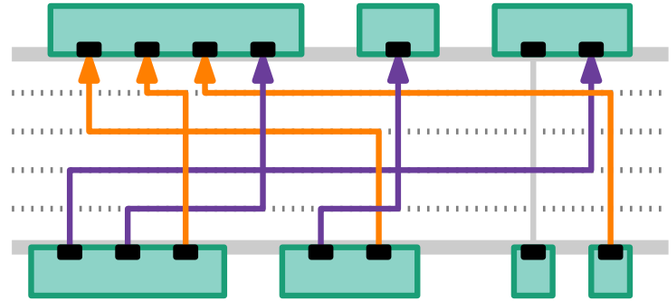
???

- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n := \#$ vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

$\Rightarrow$ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



can we do better?

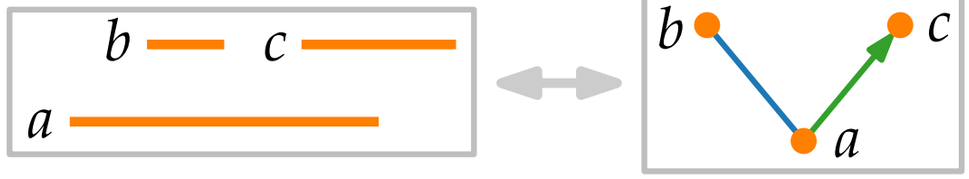
- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

bidirectional?

- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems



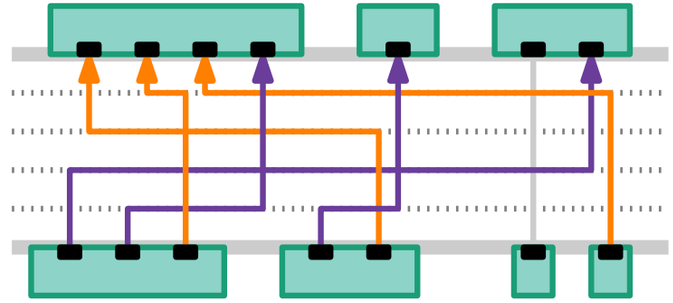
- We have introduced the natural concept of directional interval graphs.

??? Reviewer: Consider containment interval graphs!

- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n := \#$ vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)



⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)

can we do better?

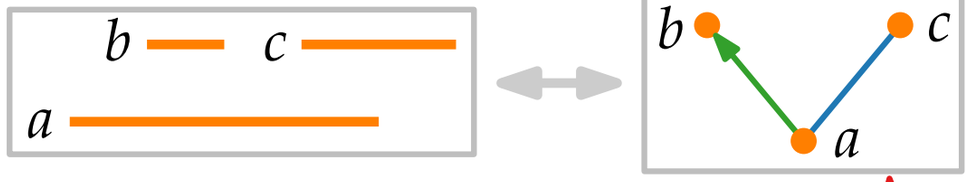
- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

bidirectional?

- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Conclusion and Open Problems



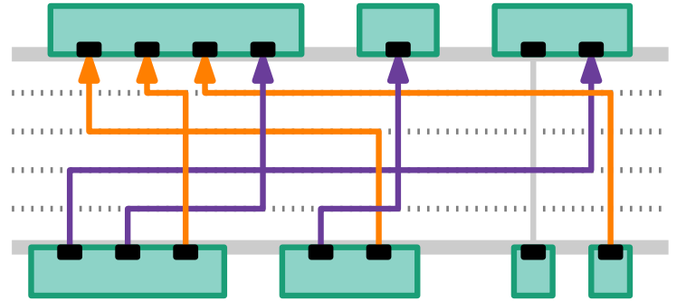
- We have introduced the natural concept of directional interval graphs.

??? Reviewer: Consider containment interval graphs!

- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)



⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)

can we do better?

- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

bidirectional?

- For the more general case of mixed interval graphs, coloring is NP-hard.

(Remark: NP-hardness requires both directed and undirected edges.)

Coloring and Recognizing Mixed Interval Graphs

ISAAC 2023, Kyoto

Grzegorz Gutowski — Uniwersytet Jagielloński Kraków

Konstanty Szaniawski — Uniwersytet Jagielloński Kraków

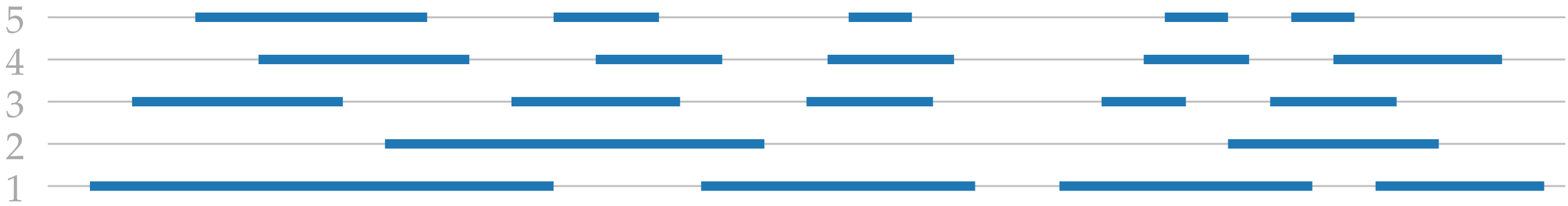
Felix Klesen — Julius-Maximilians-Universität Würzburg

Paweł Rzążewski — Warsaw University of Technology

Alexander Wolff — Julius-Maximilians-Universität Würzburg

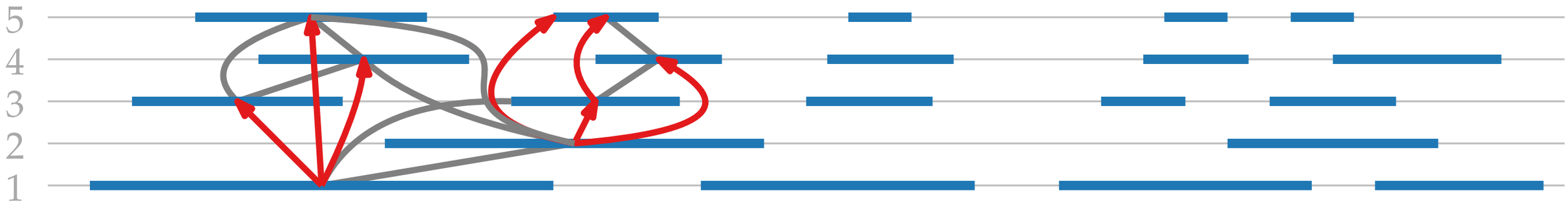
Johannes Zink — Warsaw University of Technology

Some Observation about Interval Containment Graphs



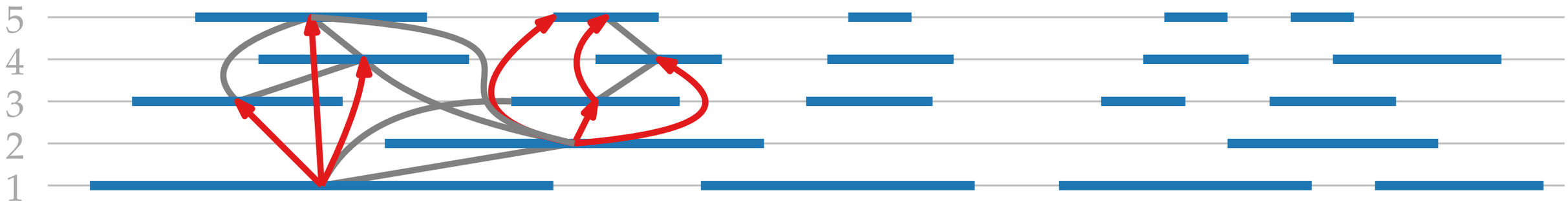
Let $\mathcal{I}$ be a set of intervals.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

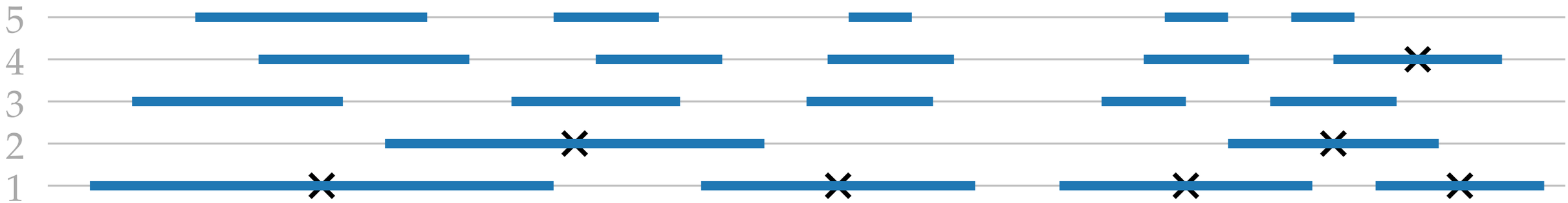
Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

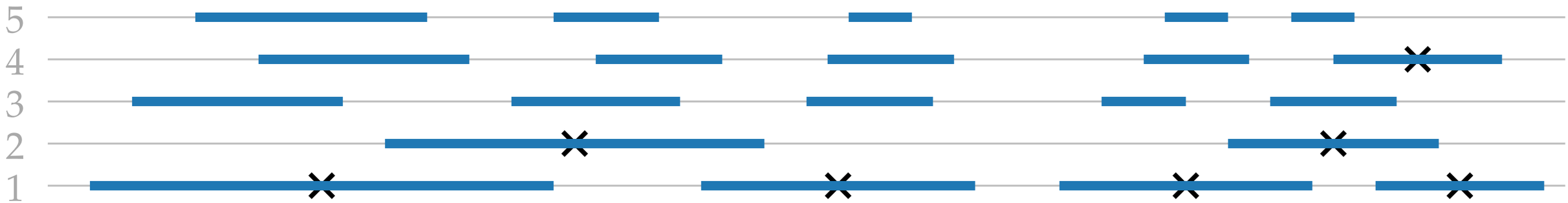
Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Some Observation about Interval Containment Graphs

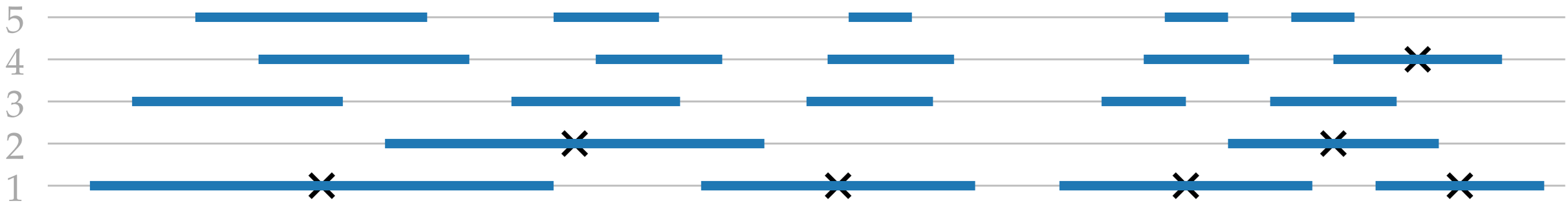


Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph

Some Observation about Interval Containment Graphs

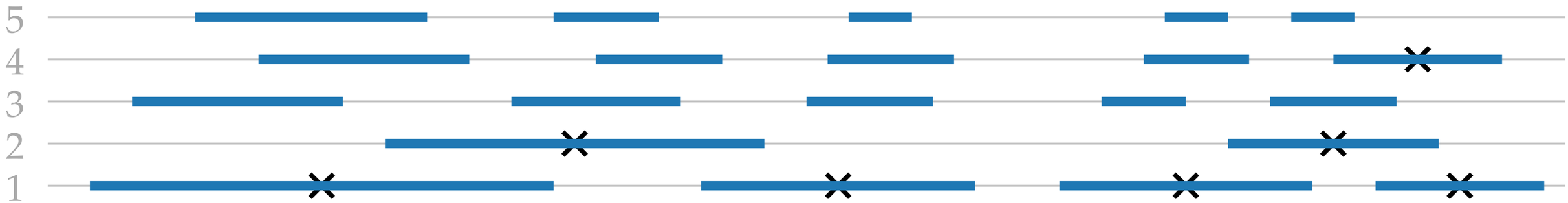


Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Some Observation about Interval Containment Graphs



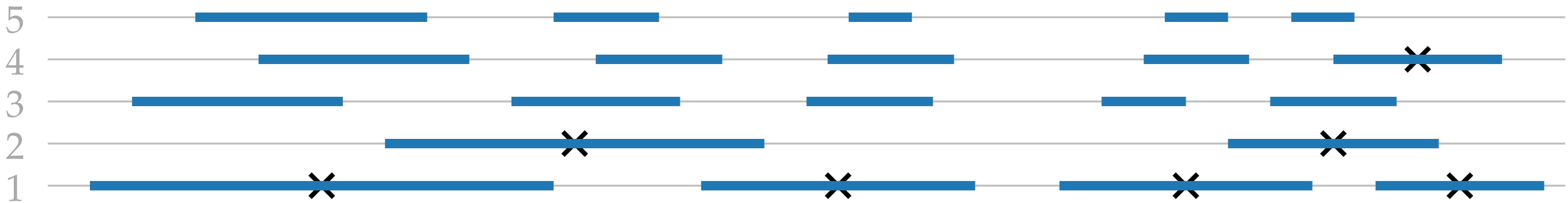
Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) =$

Some Observation about Interval Containment Graphs



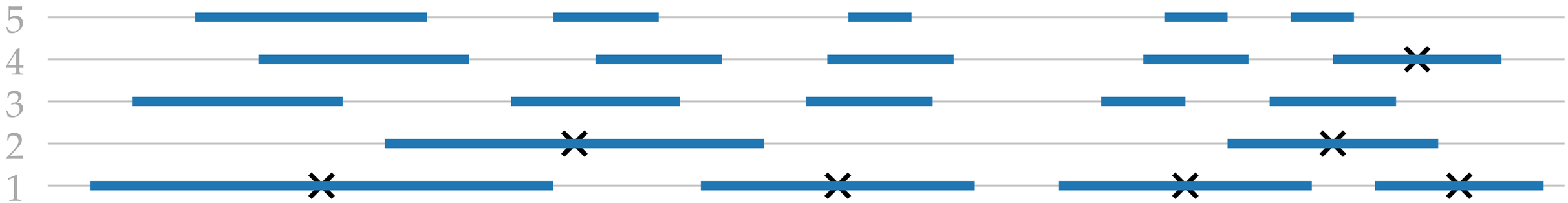
Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

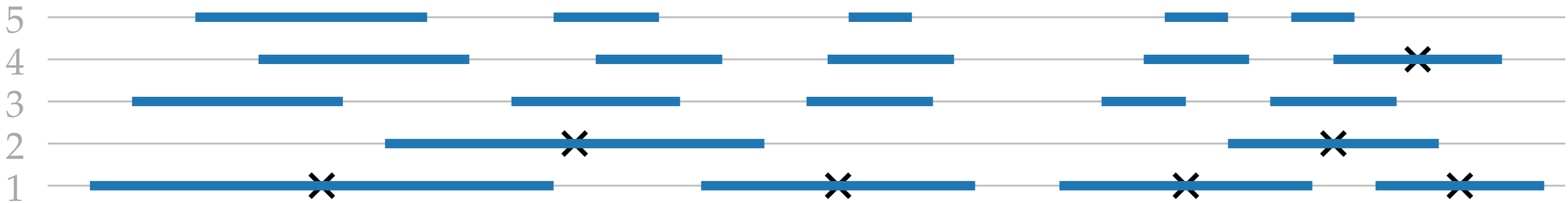
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

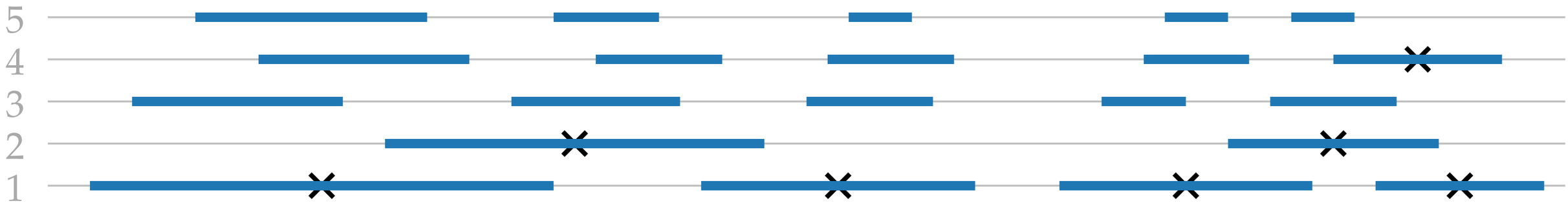
Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.

Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

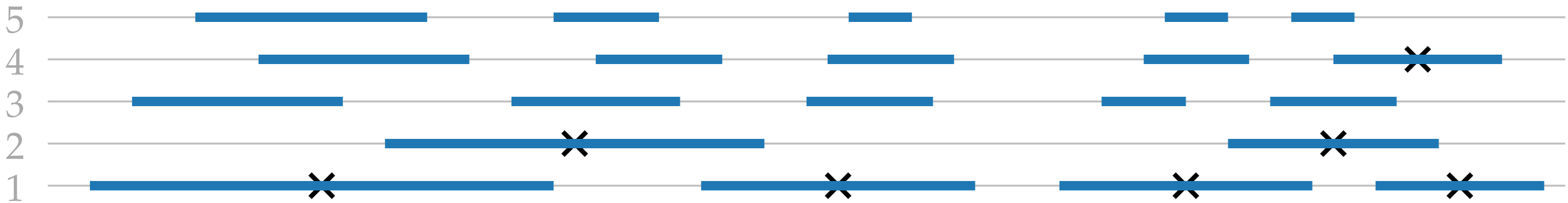
Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

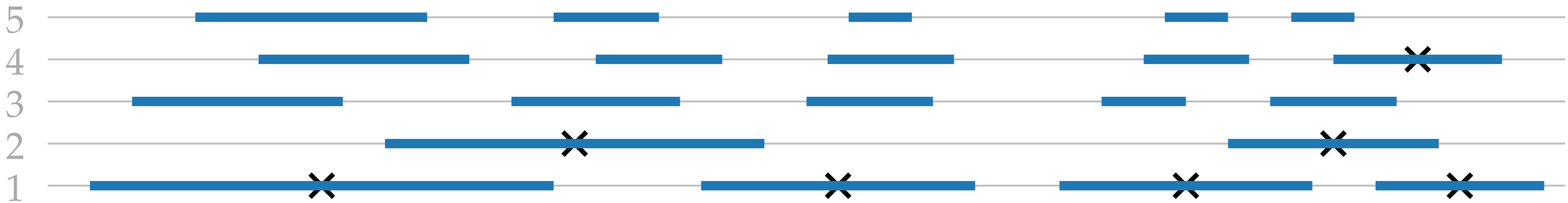
Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

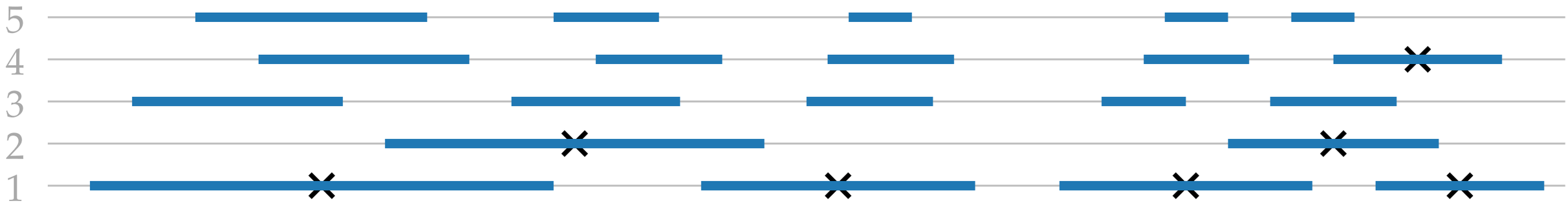
Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$ contains no induced

Some Observation about Interval Containment Graphs



Let $\mathcal{I}$ be a set of intervals. Let $G = \mathcal{C}[\mathcal{I}]$ be the containment graph induced by $\mathcal{I}$.
Let $M(\mathcal{I})$ be the set of inclusion-wise maximum elements in $\mathcal{I}$.

Then $\mathcal{C}[M(\mathcal{I})]$ is a *proper* interval graph – no interval contains another interval.

Also note that $\bigcup M(\mathcal{I}) = \bigcup \mathcal{I}$.

Let R be an inclusion-wise minimal subset of $M(\mathcal{I})$ such that $\bigcup R = \bigcup \mathcal{I}$.

Claim. $\mathcal{C}[R]$ is an undirected linear forest.

Proof. $\mathcal{C}[R]$ is proper $\Rightarrow$ contains no induced $K_{1,3}$ and no induced C_ℓ for $\ell \geq 4$.
It remains to show that $\mathcal{C}[R]$ contains no triangle.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
$\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \omega(\mathcal{C}[\mathcal{I}]) - 1$ colors.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
$\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1, 2\}$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq$

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
$\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.
Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals,
$\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.
Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G. ↯

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1,2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G. ↯

Induction $\Rightarrow G'$ admits a proper coloring f_2 using at most $2 \cdot \omega(G') - 1$ colors.

A 2-Approximation Algorithm for Coloring

Theorem. For any set $\mathcal{I}$ of intervals, $\mathcal{C}[\mathcal{I}]$ admits a proper coloring with at most $2 \cdot \overbrace{\omega(\mathcal{C}[\mathcal{I}])}^{\omega} - 1$ colors.

Since $\mathcal{C}[R]$ is a linear forest, it admits a proper coloring $f_1 : R \to \{1, 2\}$.

If $R = \mathcal{I}$, we are done (using only ω many colors), so we assume $\mathcal{I} \setminus R \neq \emptyset$.

Slightly abusing notation, let $G' := G - R$.

Claim. $\omega(G') \leq \omega - 1$.

Proof. Suppose that there is a clique S in G' of size ω.

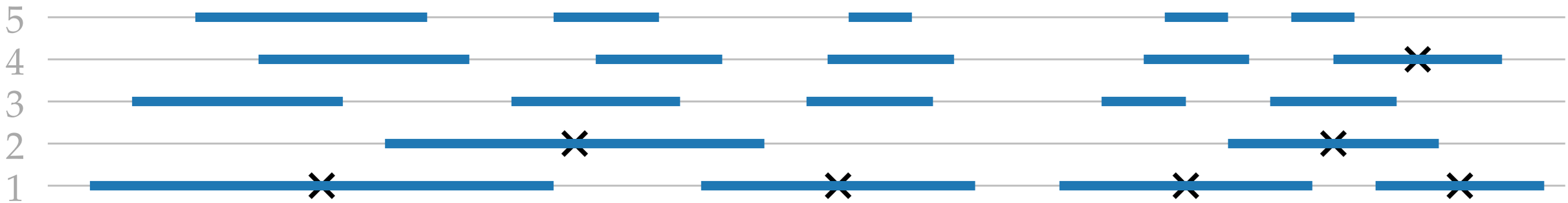
Helly property of intervals $\Rightarrow \bigcap S \neq \emptyset$. Let $p \in \bigcap S$.

Pick an $r \in R$ that contains p. $\Rightarrow S \cup \{r\}$ is a clique of size $\omega + 1$ in G. ↯

Induction $\Rightarrow G'$ admits a proper coloring f_2 using at most $2 \cdot \omega(G') - 1$ colors.

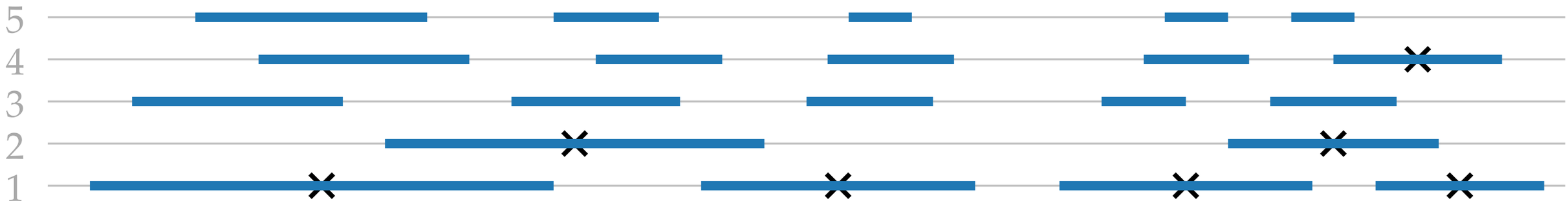
With f_1 and f_2, we construct a proper coloring f of G using colors $\{1, \ldots, 2\omega - 1\}$.

An Inductive Coloring



Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

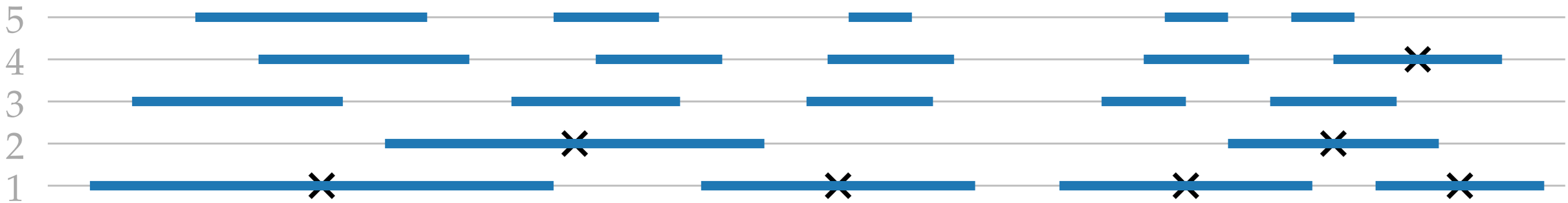
An Inductive Coloring



Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

An Inductive Coloring

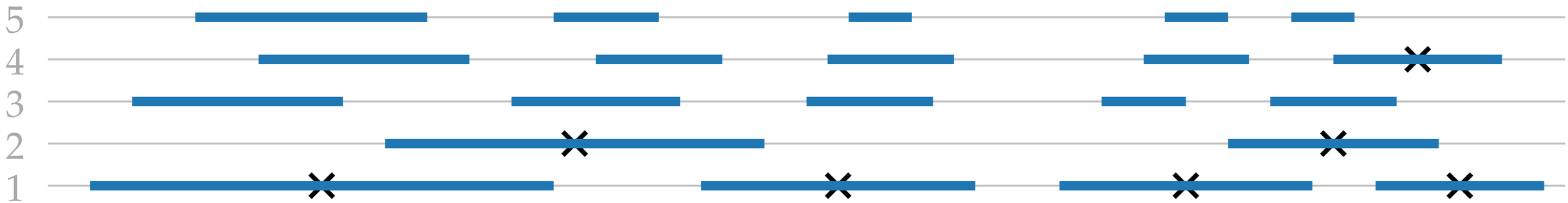


Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$.
2. If $x \subseteq y$, then $f(x) > f(y)$.

An Inductive Coloring

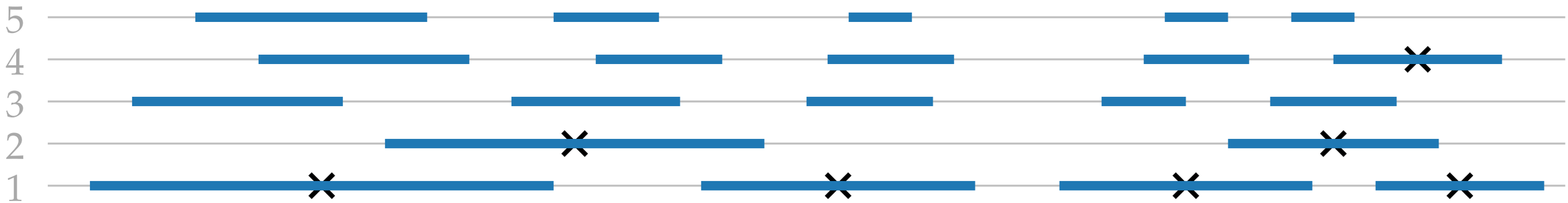


Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$.

An Inductive Coloring

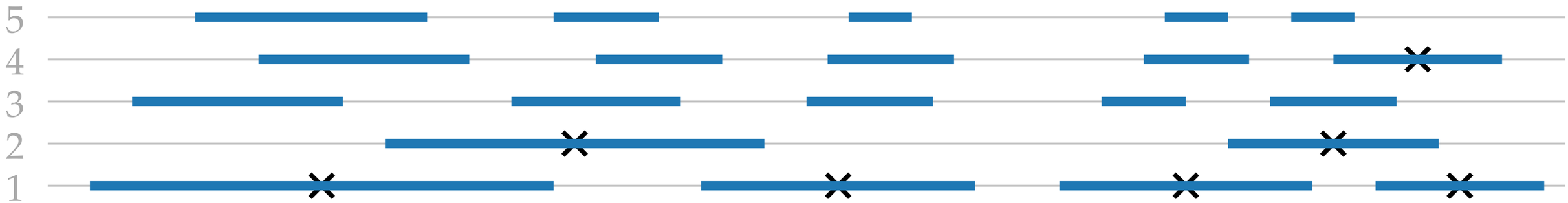


Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R$

An Inductive Coloring

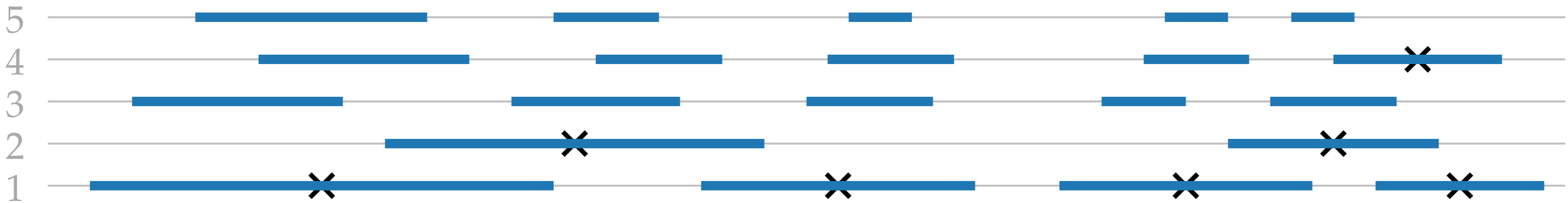


Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

An Inductive Coloring



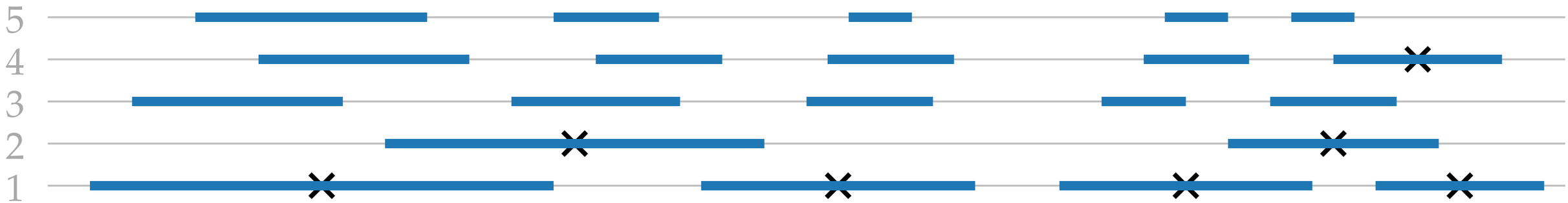
Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x)$

An Inductive Coloring



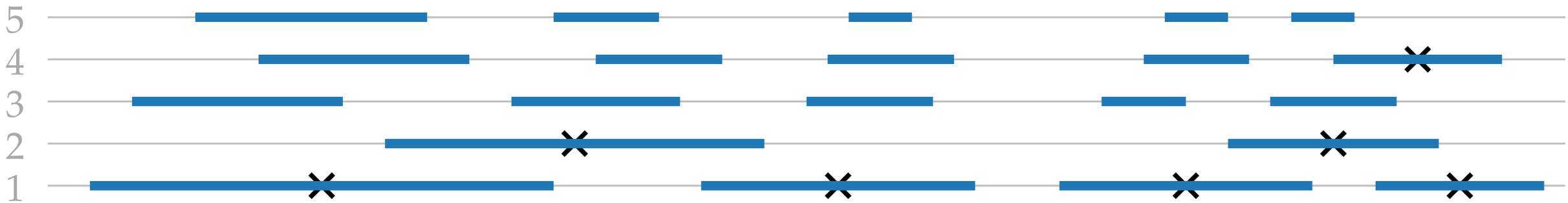
Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \Rightarrow y \neq R$

An Inductive Coloring



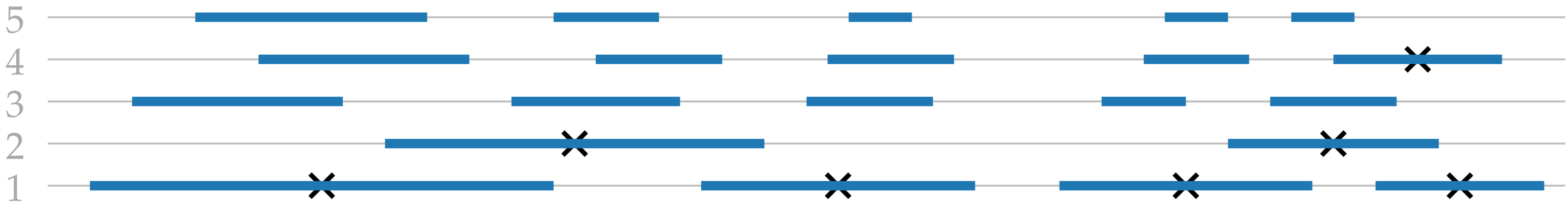
Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \Rightarrow y \neq R$, but $f_2(x) > f_2(y)$.

An Inductive Coloring



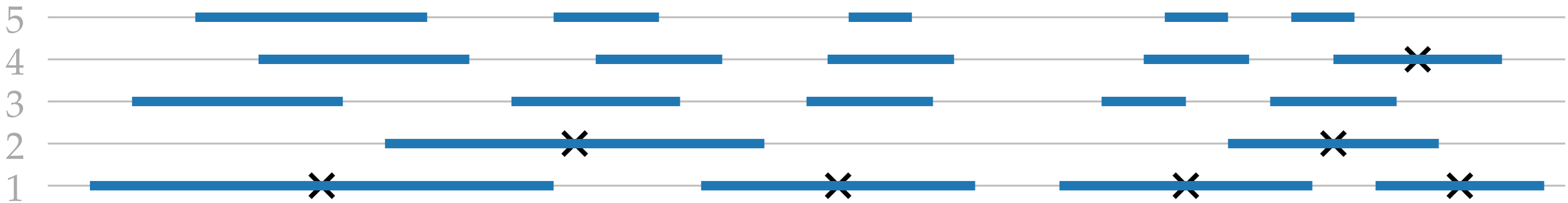
Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \Rightarrow y \neq R$, but $f_2(x) > f_2(y)$. ↯

An Inductive Coloring



Let $f(x) = \begin{cases} f_1(x) & \text{if } x \in R, \\ f_2(x) + 2 & \text{else.} \end{cases}$

This defines a proper coloring of G:

1. If $x \cap y \neq \emptyset$, then $f(x) \neq f(y)$. Check: $x, y \in R$; $x, y \neq R$; $x \in R$ and $y \neq R$.
2. If $x \subseteq y$, then $f(x) > f(y)$. Observe that $x \neq R \Rightarrow f(x) \geq 3$

   Suppose $f(y) > f(x) \Rightarrow y \neq R$, but $f_2(x) > f_2(y)$. ↯

Corollary. There is a 2-approximation for coloring interval containment graphs properly. Given n intervals, the algorithm runs in $O(n \log n)$ time.

A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

A Lower Bound Example

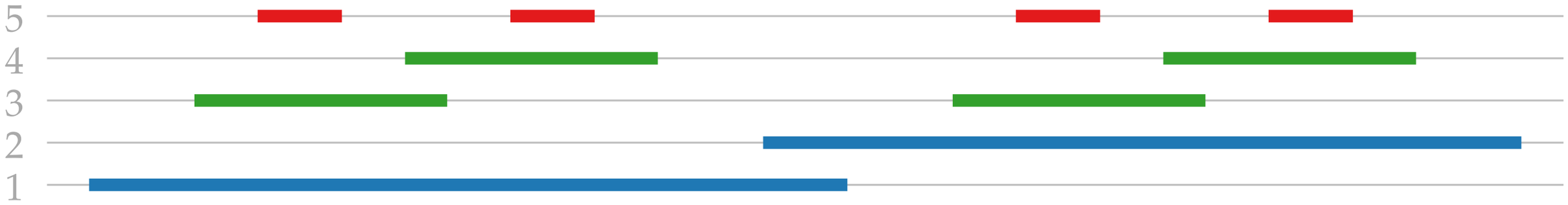
Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

This yields $\lim_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.

A Lower Bound Example

Proposition. There is an infinite family $(\mathcal{I}_n)_{n\geq 1}$ of sets of intervals with $|\mathcal{I}_n| = 3 \cdot 2^{n-1} - 2$, $\chi(\mathcal{C}[\mathcal{I}_n]) = 2n - 1$, and $\omega(\mathcal{C}[\mathcal{I}_n]) = n$.

This yields $\lim_{n\to\infty} \chi(\mathcal{I}_n)/\omega(\mathcal{I}_n) = 2$.



Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Proof. By reduction from (exact) 3-SAT, where each clause has exactly 3 literals.

Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

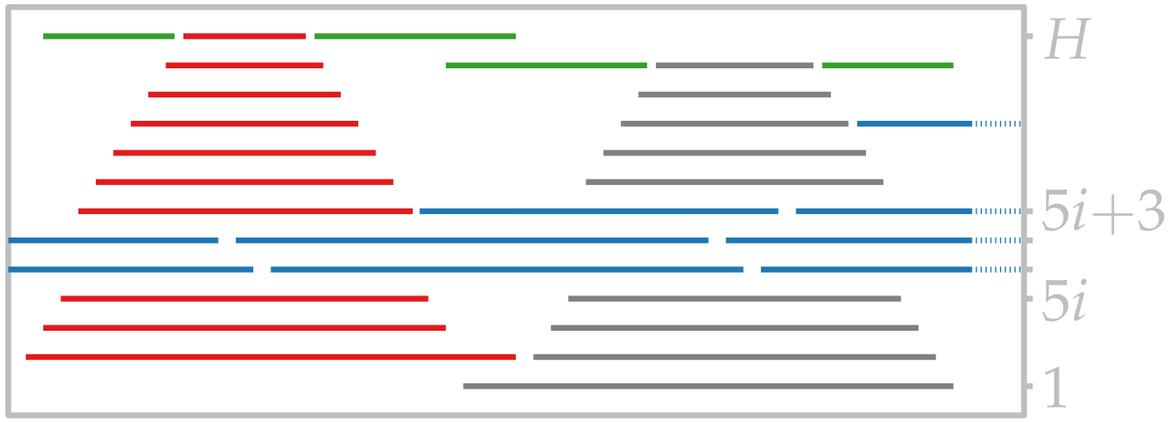
Proof. By reduction from (exact) 3-SAT, where each clause has exactly 3 literals.

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-SAT with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$.

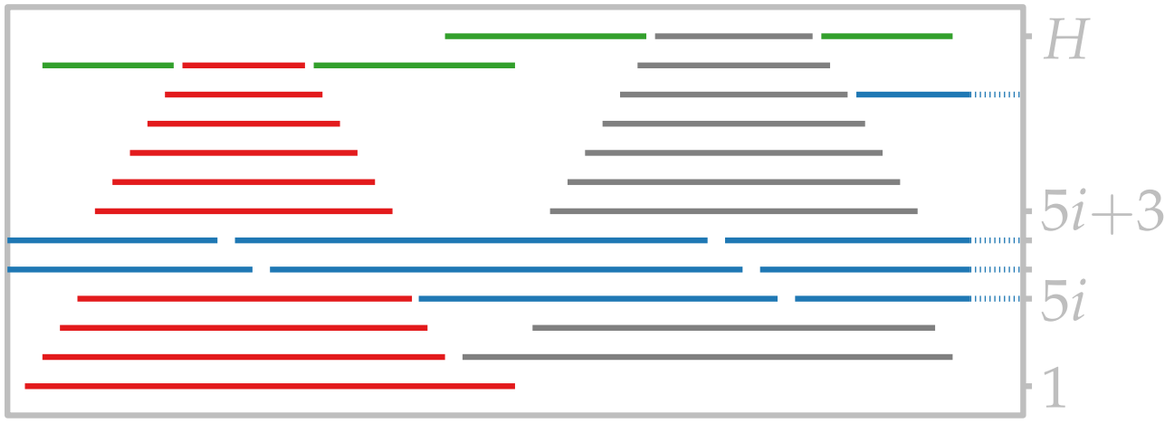
Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Proof. By reduction from (exact) 3-SAT, where each clause has exactly 3 literals.



x true



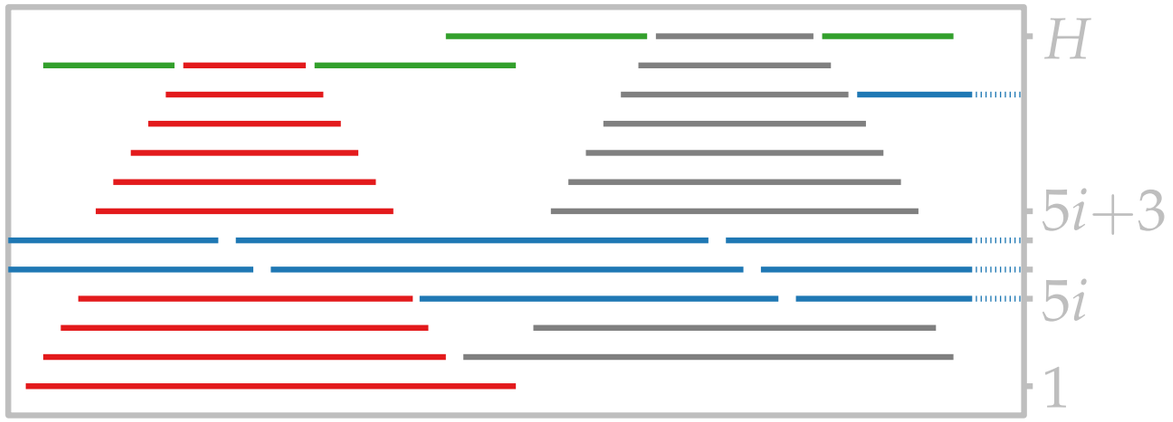
x false

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-SAT with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$.

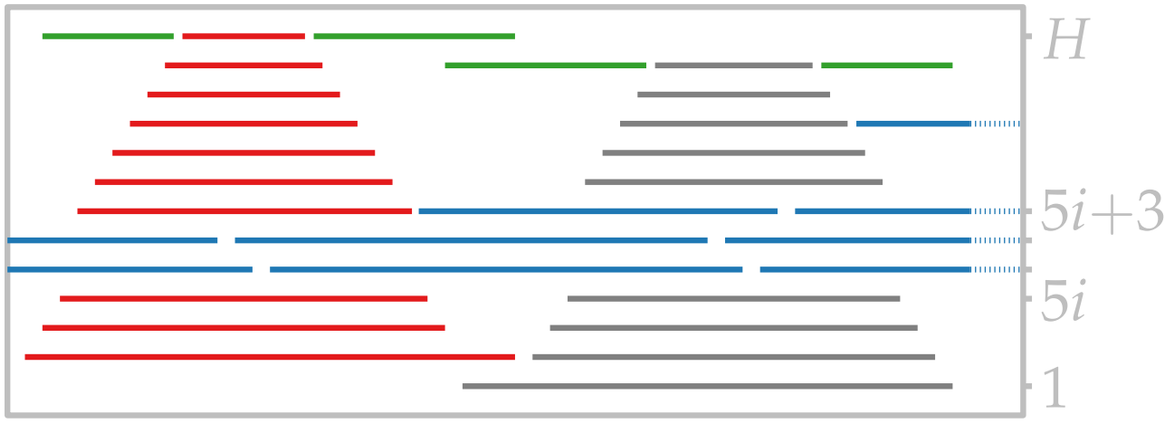
Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Proof. By reduction from (exact) 3-SAT, where each clause has exactly 3 literals.



x false



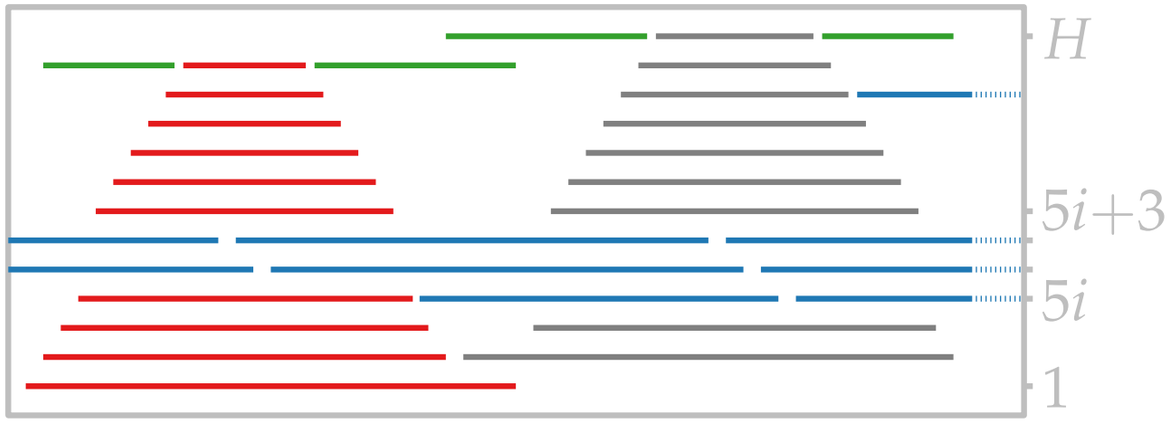
x true

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-SAT with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$.

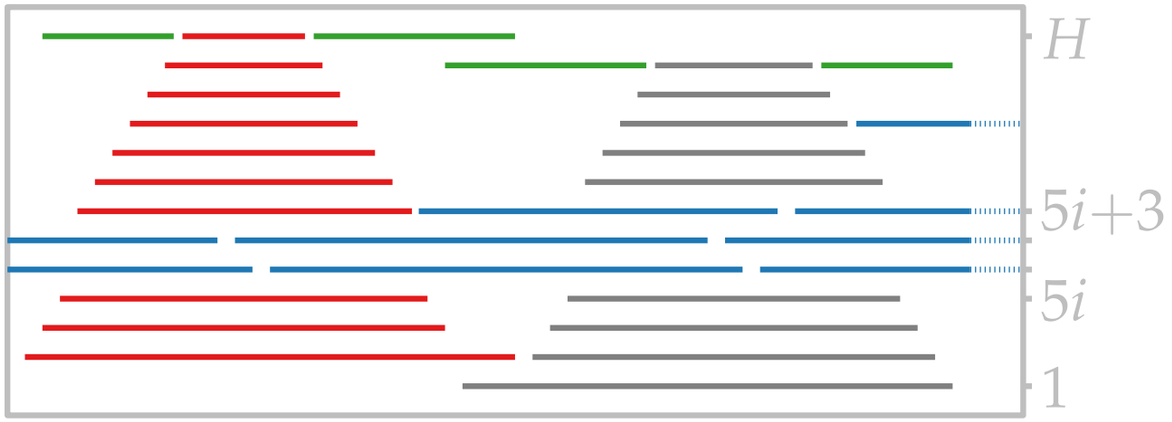
Computational Complexity

Theorem. Given a set $\mathcal{I}$ of intervals and a positive integer k, it is NP-hard to decide whether $\chi(\mathcal{C}[\mathcal{I}]) \leq k$.

Proof. By reduction from (exact) 3-SAT, where each clause has exactly 3 literals.



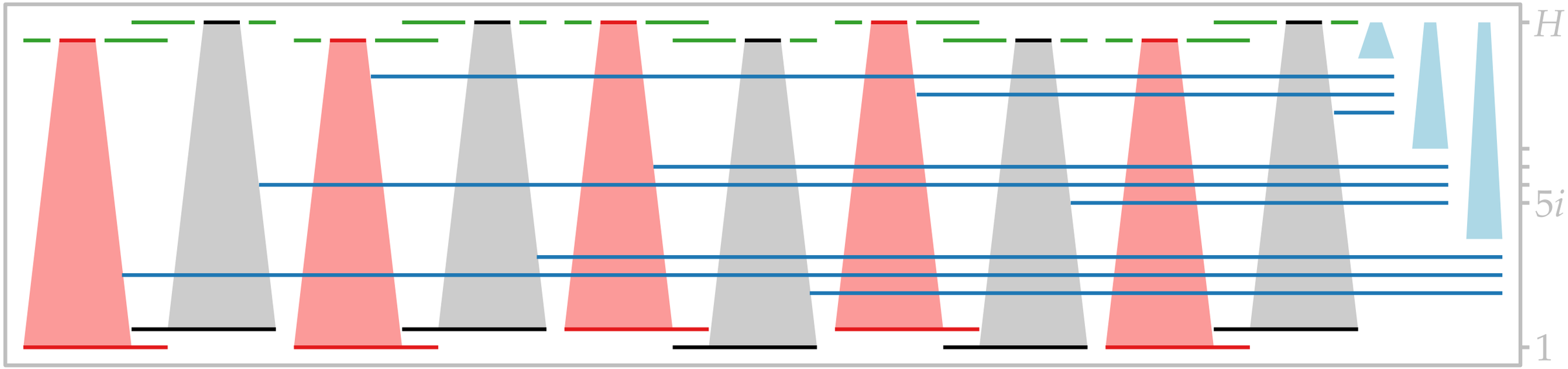
x false



x true

Let $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ be an instance of 3-SAT with variables $\{x_1, x_2, \ldots, x_n\}$, and let $H = 5m + 1$. We construct a set $\mathcal{I}_\varphi$ of intervals.

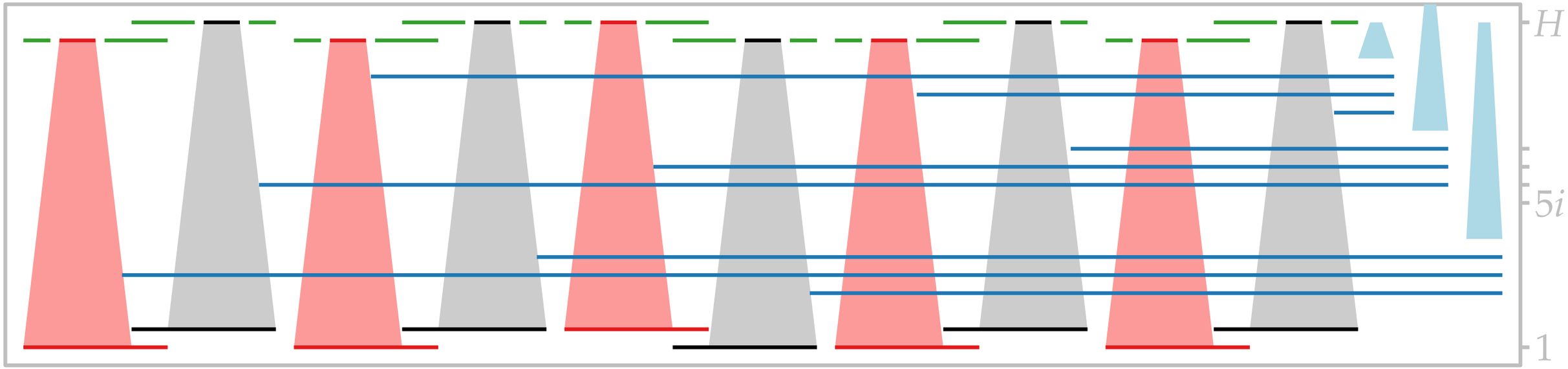
Clause Gadget



x_1 false x_2 false x_3 true x_4 true x_5 false

Example for $(\neg x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

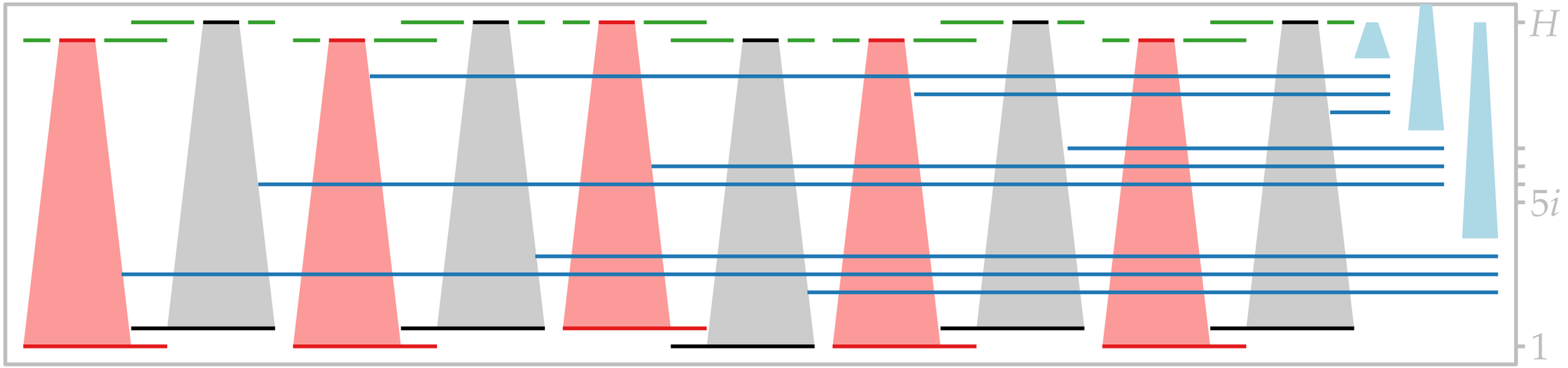
Clause Gadget



x_1 false x_2 false x_3 true x_4 false x_5 false

Example for $(\neg x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

Clause Gadget



x_1 false x_2 false x_3 true x_4 false x_5 false

Example for $(\neg x_2 \vee \neg x_4 \vee x_5) \wedge (x_1 \vee \neg x_3 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3)$.

The graph $\mathcal{C}[\mathcal{I}_\varphi]$ admits a proper coloring with H colors $\Leftrightarrow$ φ is satisfiable. $\square$

Summary

Mixed interval graph class	Coloring complexity	lower bound	upper bound	approximation	Recognition
containment	NP-hard	$2\omega-1$	$2\omega-1$	2	$O(nm)$
directional	$O(n \log n)$			1	$O(n^2)$
bidirectional	NP-hard			2	open
general	NP-hard	$(\lambda+2)\omega/2$	$(\lambda+1)\omega$	$\min\{\omega, \lambda+1\}$	$O(n+m)$ [LB79]

Summary

Mixed interval graph class	complexity	Coloring lower bound	Coloring upper bound	Coloring approximation	Recognition
containment	NP-hard	$2\omega-1$	$2\omega-1$	2	$O(nm)$
directional	$O(n \log n)$			1	$O(n^2)$
bidirectional	NP-hard			2	open
general	NP-hard	$(\lambda+2)\omega/2$	$(\lambda+1)\omega$	$\min\{\omega, \lambda+1\}$	$O(n+m)$ [LB79]