Coloring Mixed and Directional Interval Graphs

GD 2022, Tokyo

Grzegorz Gutowski
Florian Mittelstädt
Ignaz Rutter
Joachim Spoerhase
Alexander Wolff
Johannes Zink

Uniwersytet Jagielloński Kraków
Julius-Maximilians-UNIVERSITÄT WÜRZBURG
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:

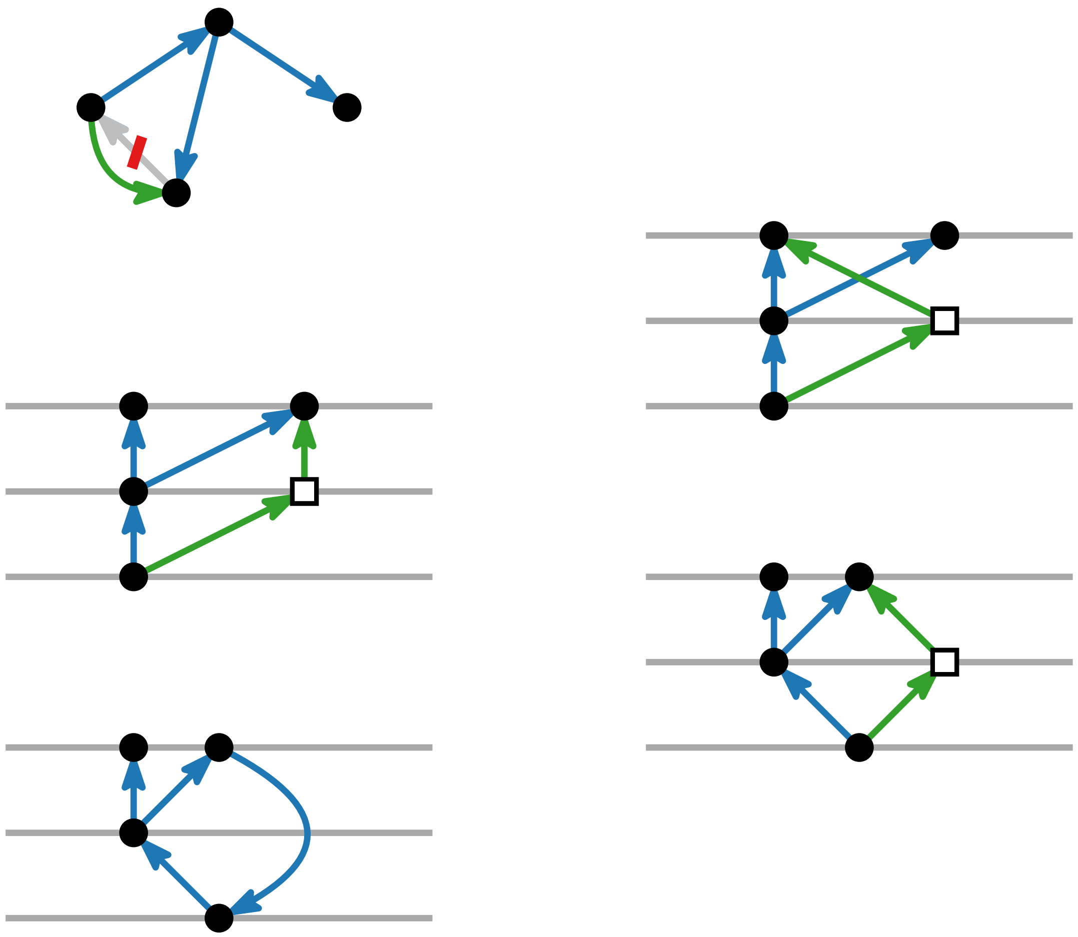
Motivation

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



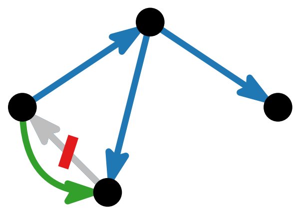
Motivation

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

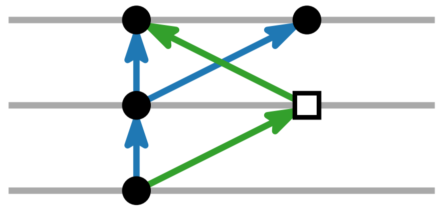
Input: directed graph G **Output:** layered drawing of G

Consists of five phases:

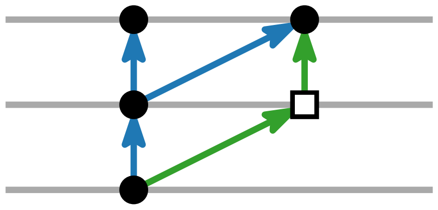
1. cycle elimination



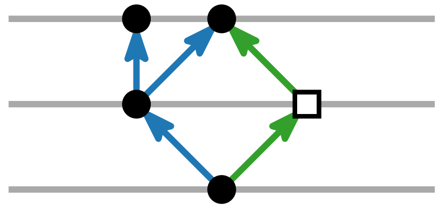
2. layer assignment



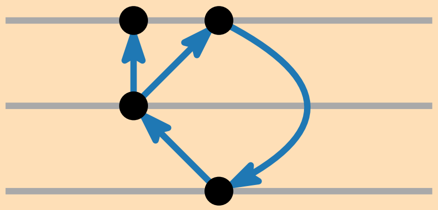
3. crossing minimization



4. node placement



5. edge routing



we want orthogonal edges!

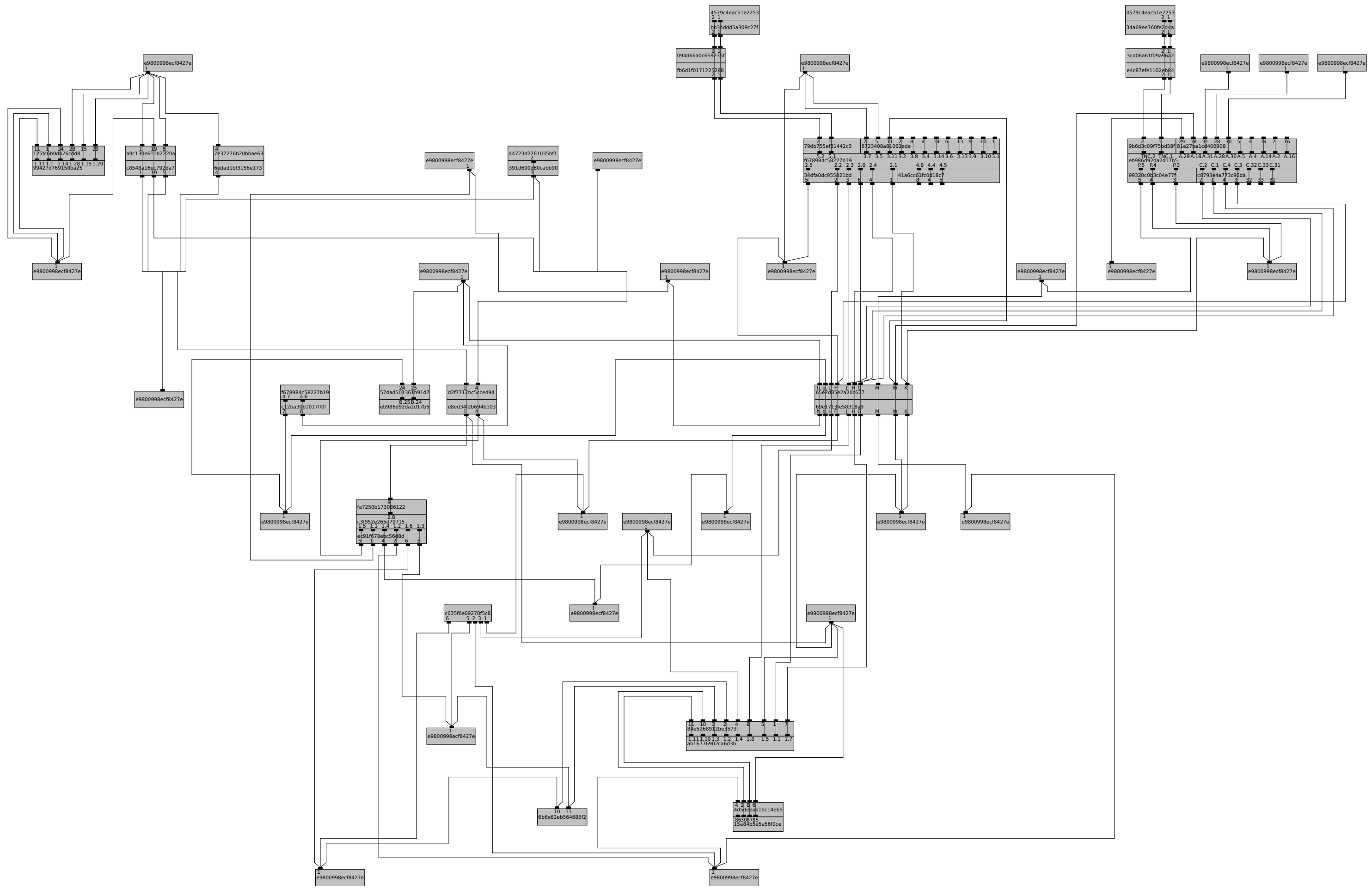
Motivation

Framework for layered g

Input: directed graph G

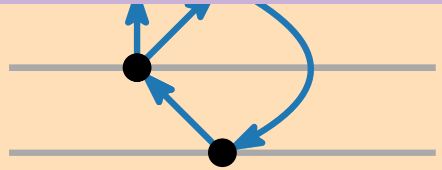
Consists of five phases:

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

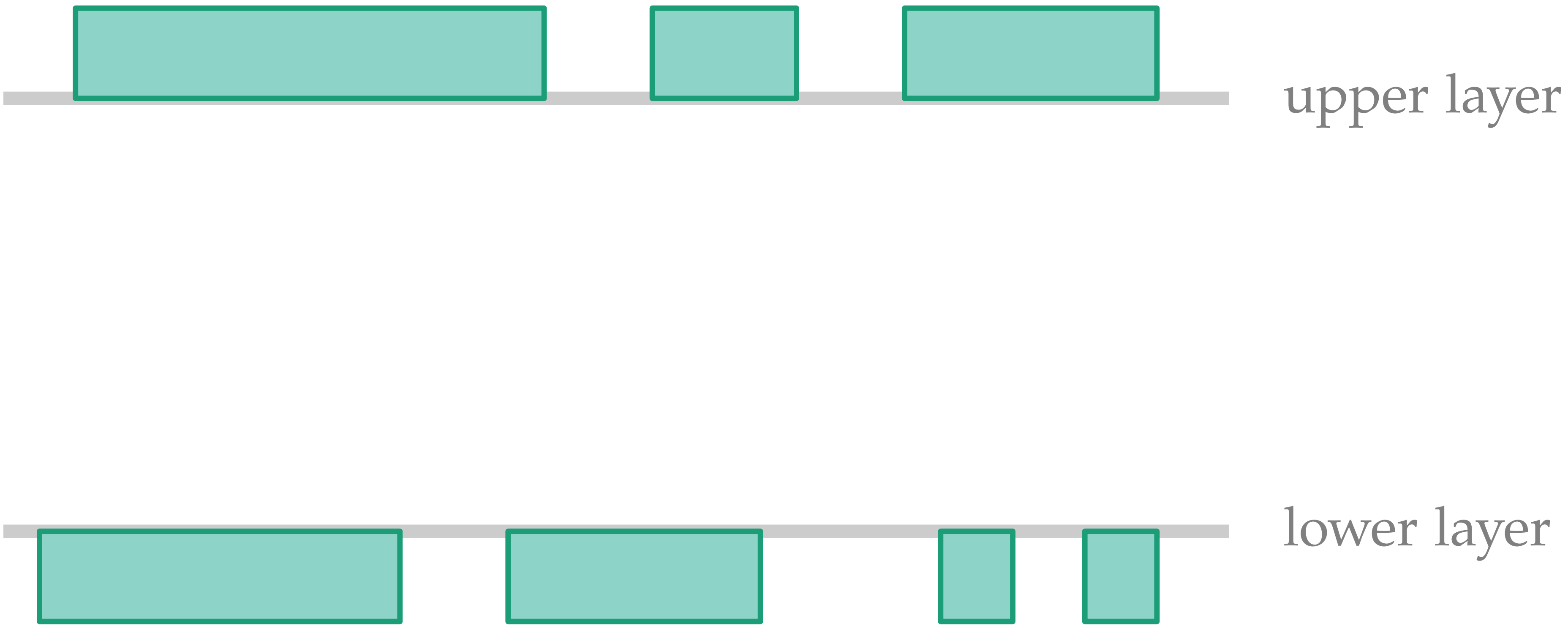
- it suffices to consider each pair of consecutive layers individually

Motivation – Layered Orthogonal Edge Routing

- it suffices to consider each pair of consecutive layers individually

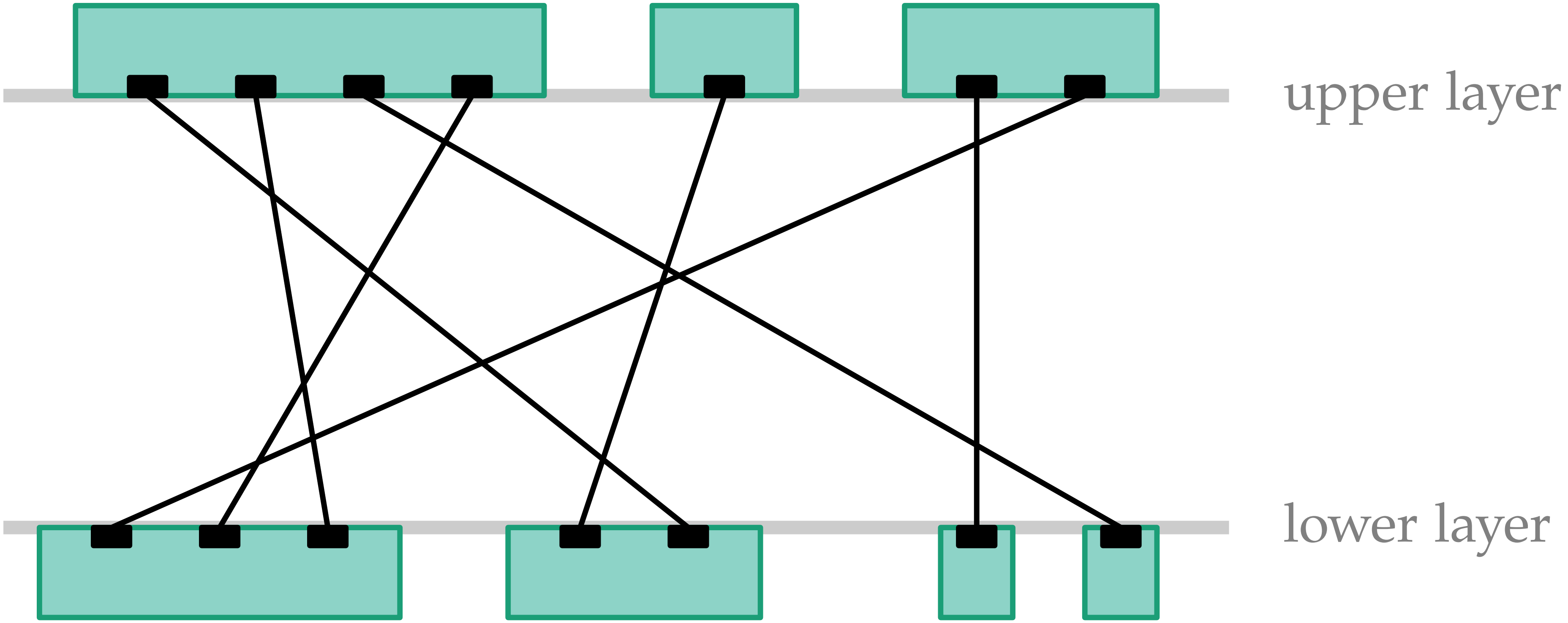
Motivation – Layered Orthogonal Edge Routing

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



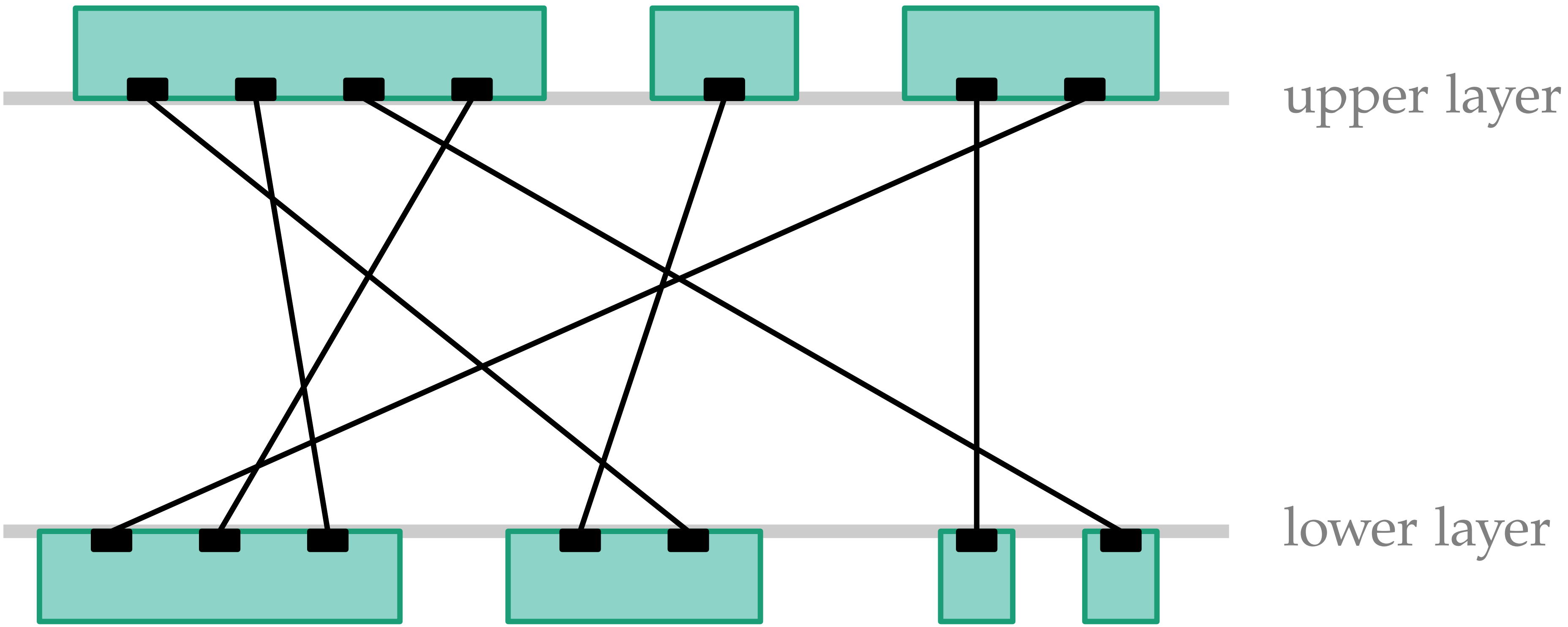
Motivation – Layered Orthogonal Edge Routing

- 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)



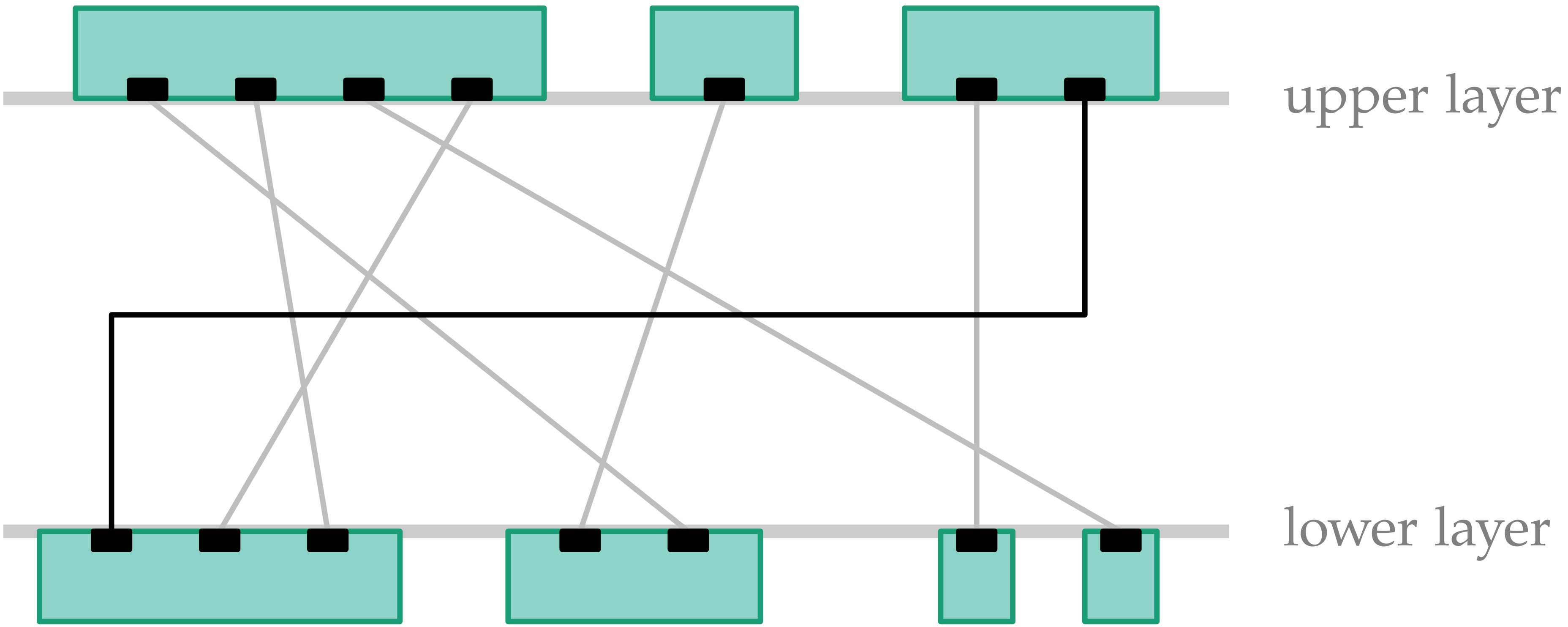
Motivation – Layered Orthogonal Edge Routing

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



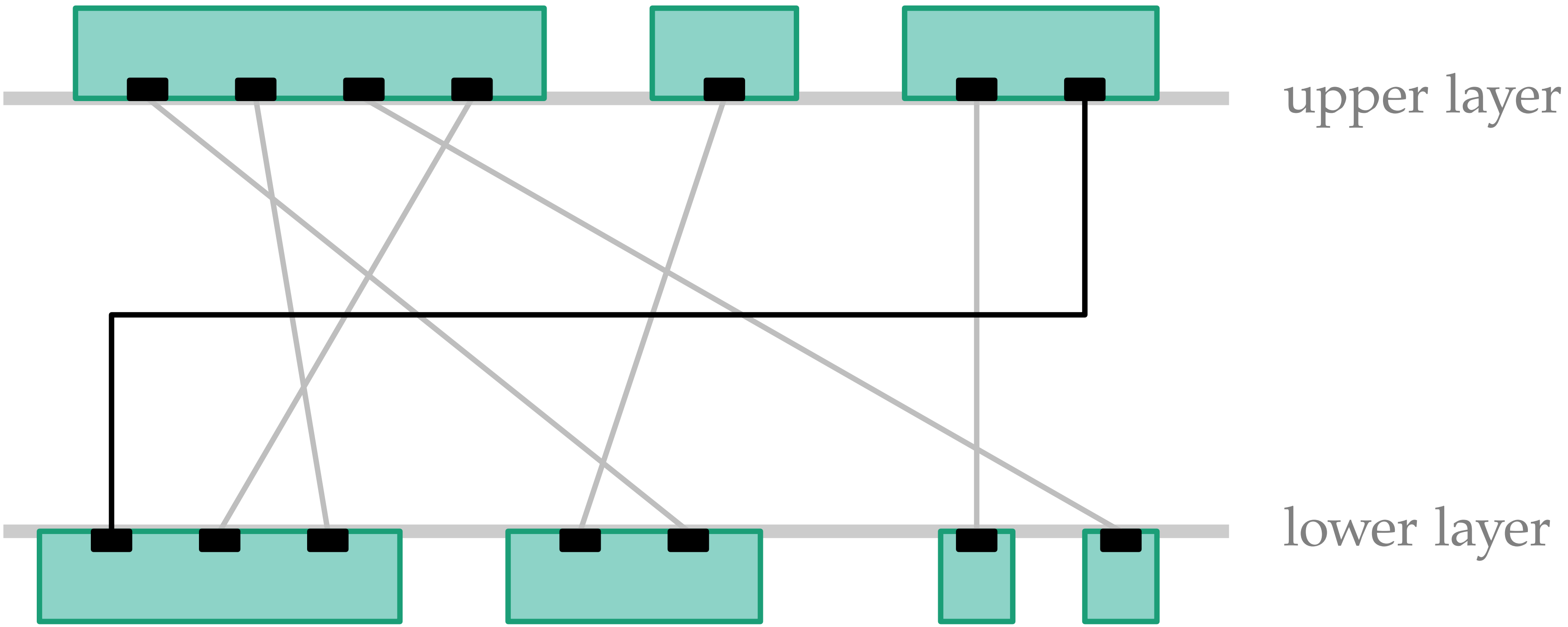
Motivation – Layered Orthogonal Edge Routing

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



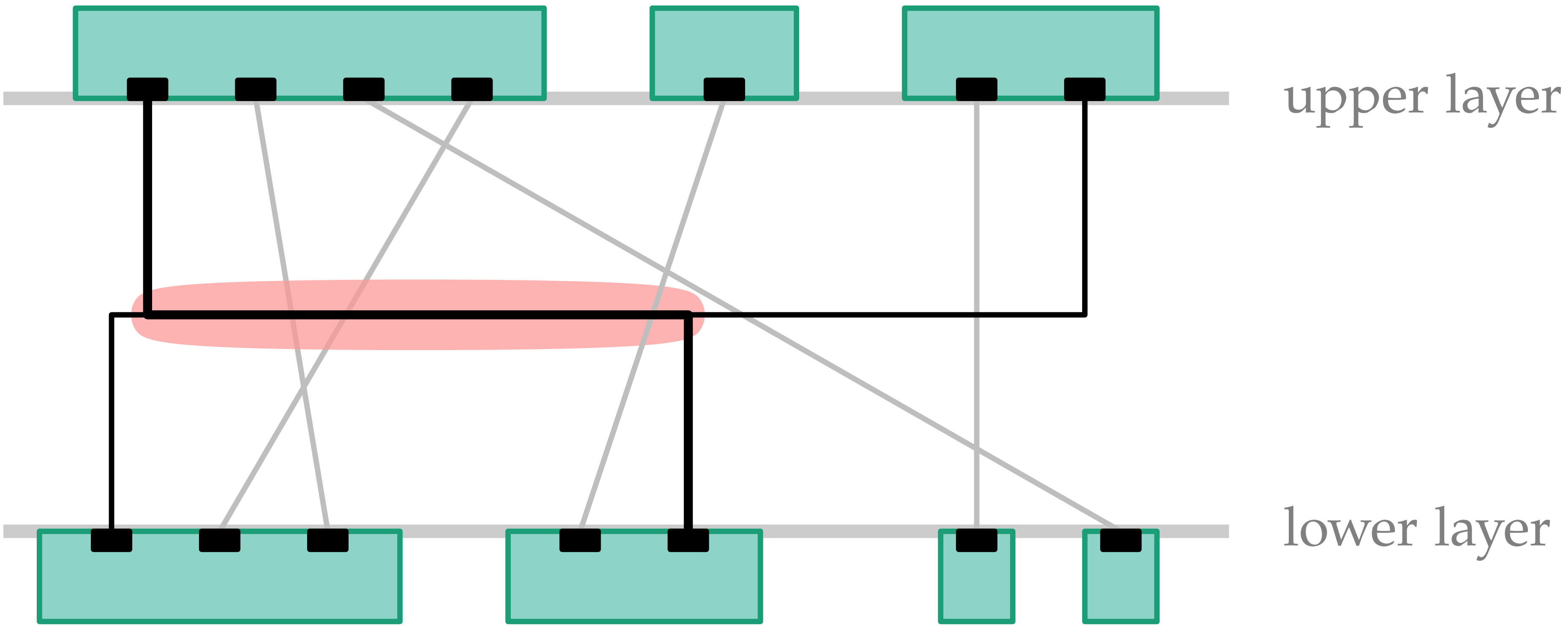
Motivation – Layered Orthogonal Edge Routing

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



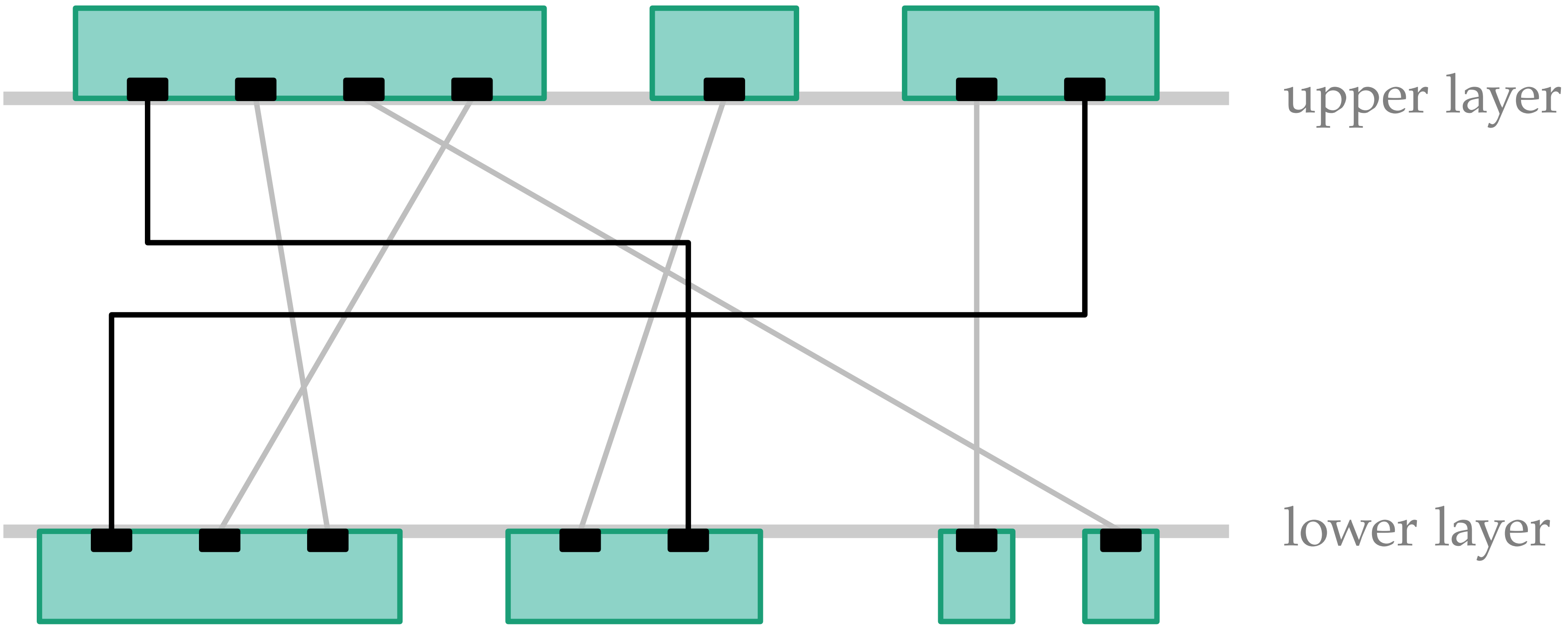
Motivation – Layered Orthogonal Edge Routing

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



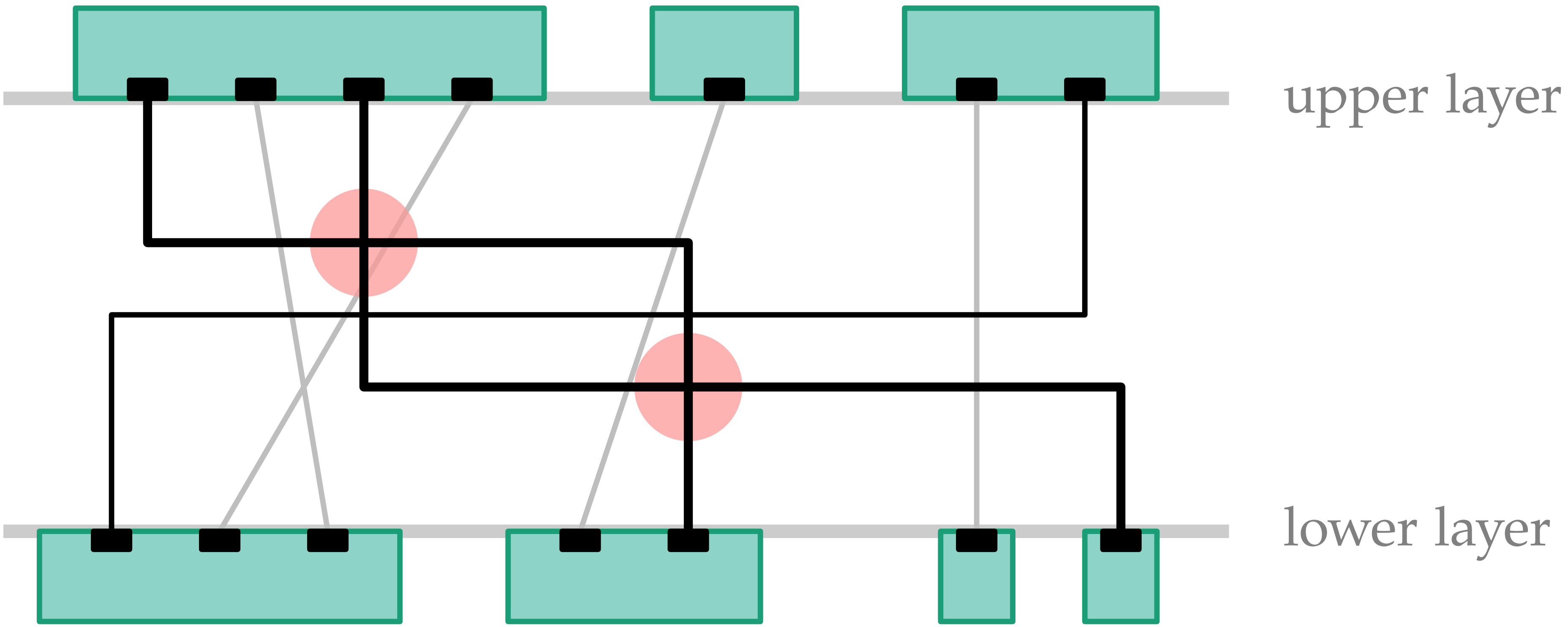
Motivation – Layered Orthogonal Edge Routing

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



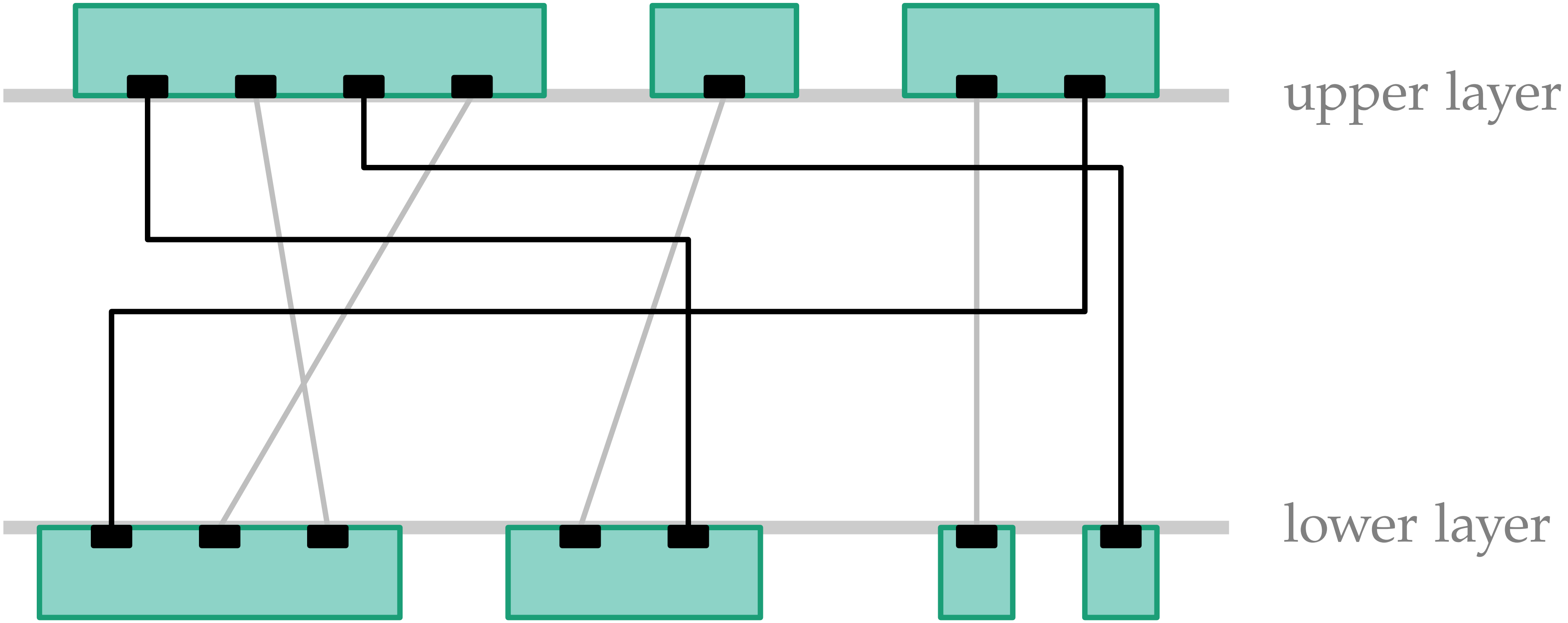
Motivation – Layered Orthogonal Edge Routing

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



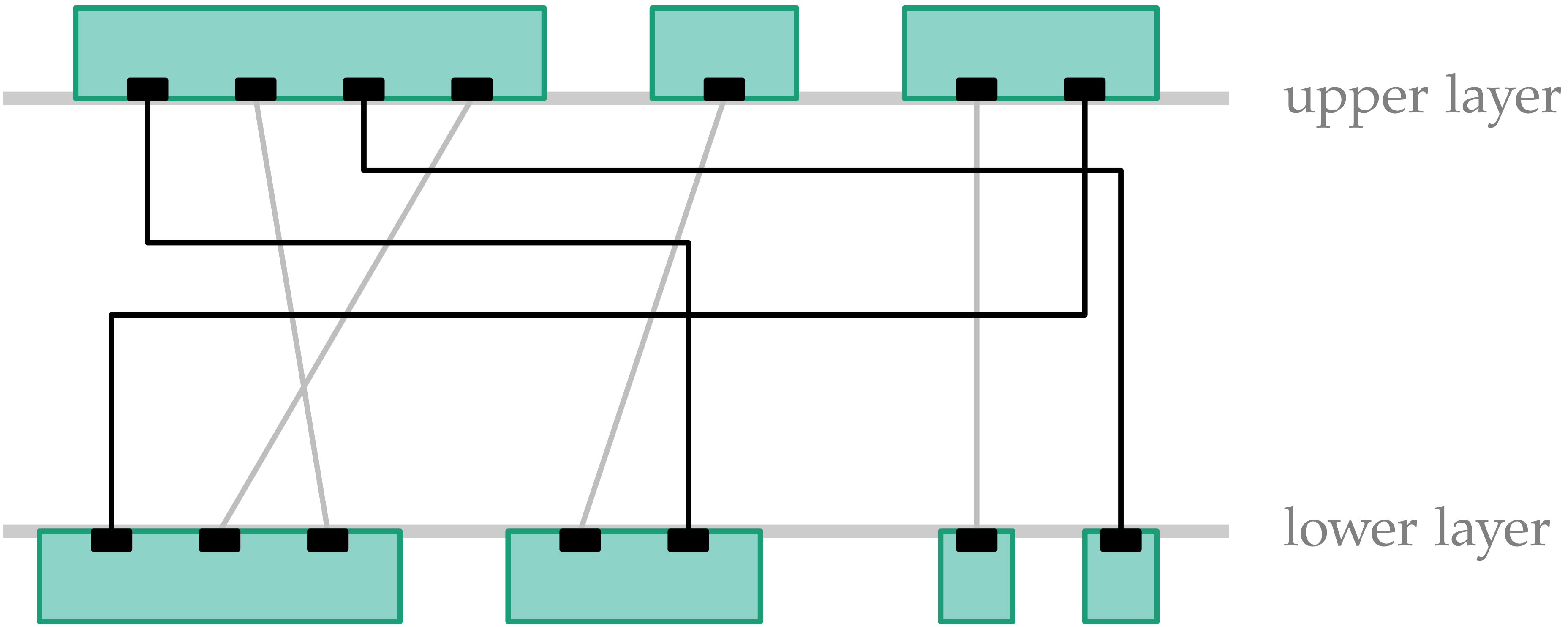
Motivation – Layered Orthogonal Edge Routing

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



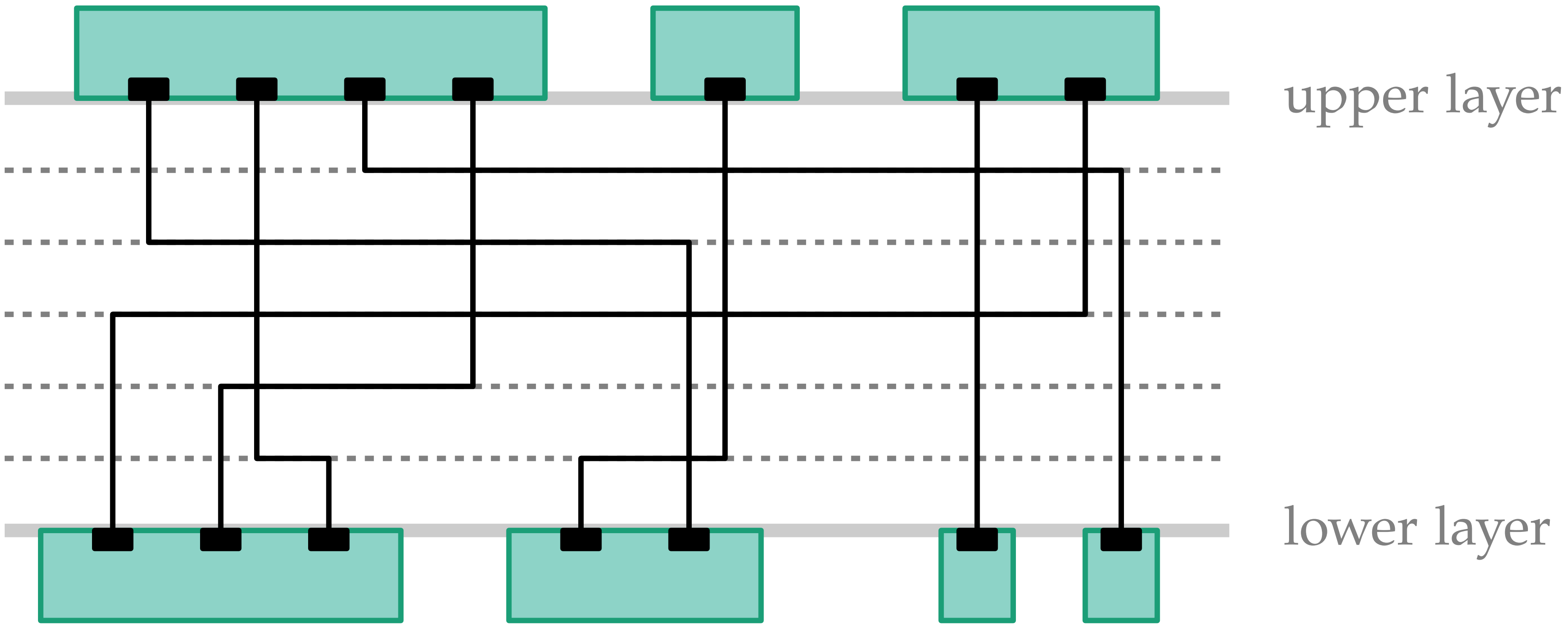
Motivation – Layered Orthogonal Edge Routing

- 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



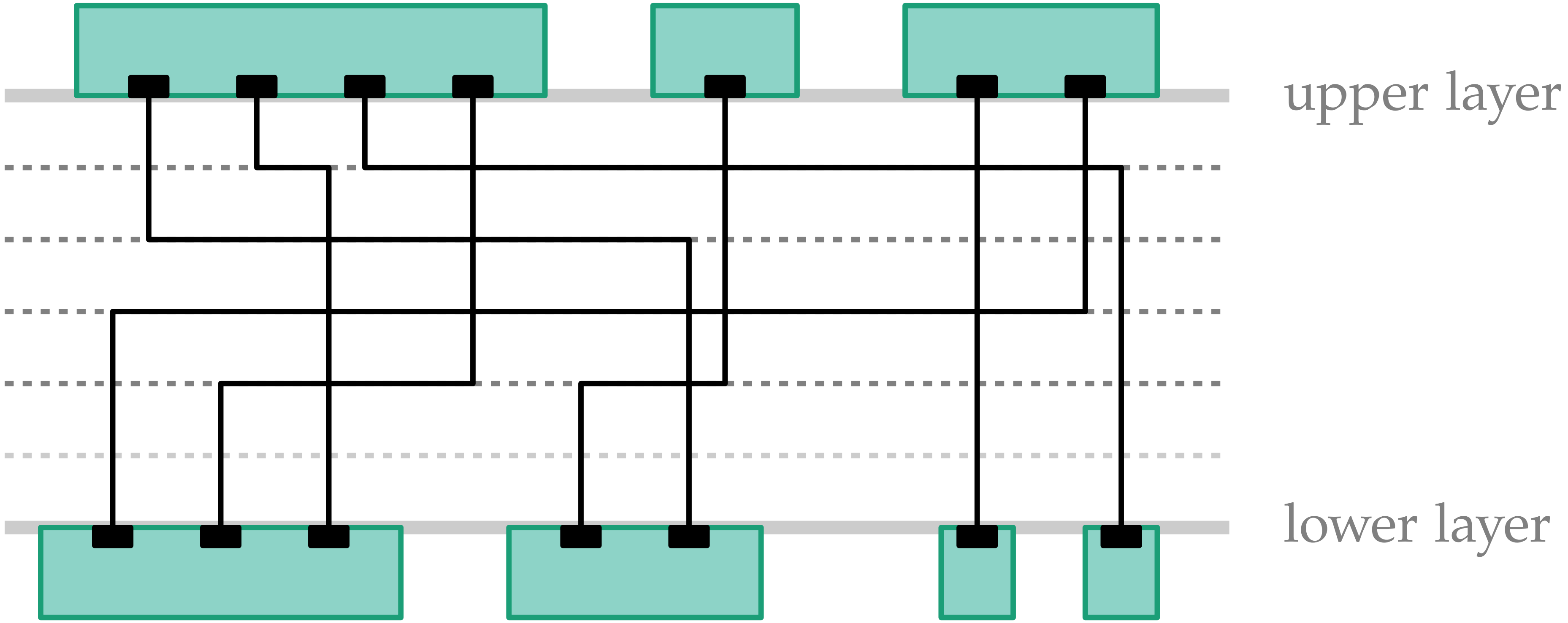
Motivation – Layered Orthogonal Edge Routing

- 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



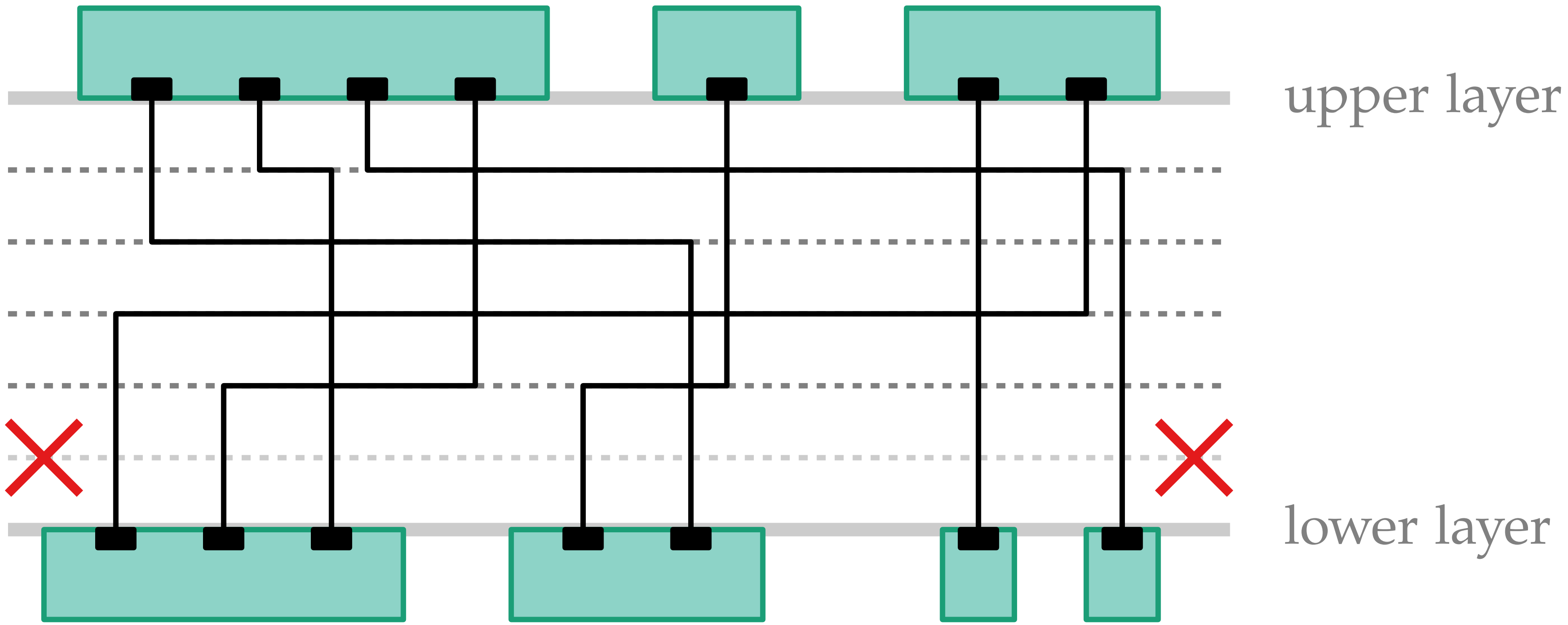
Motivation – Layered Orthogonal Edge Routing

- 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



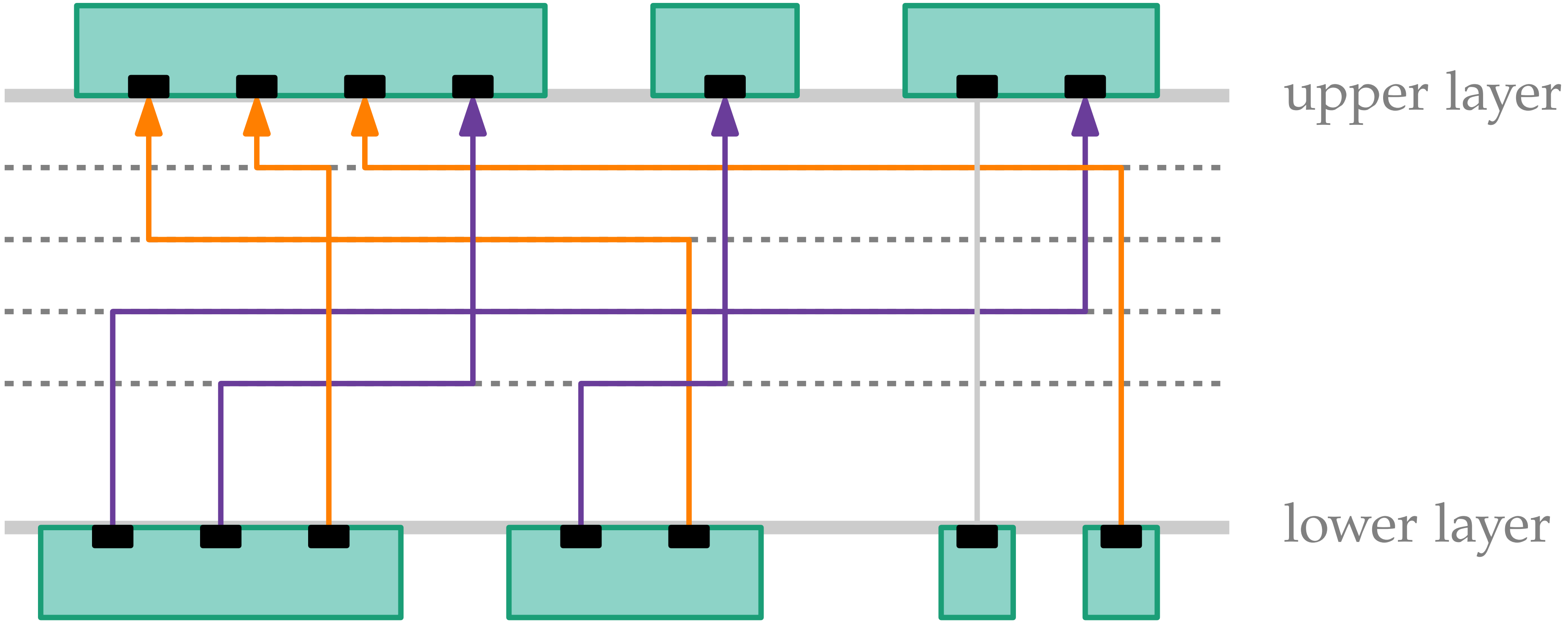
Motivation – Layered Orthogonal Edge Routing

- 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



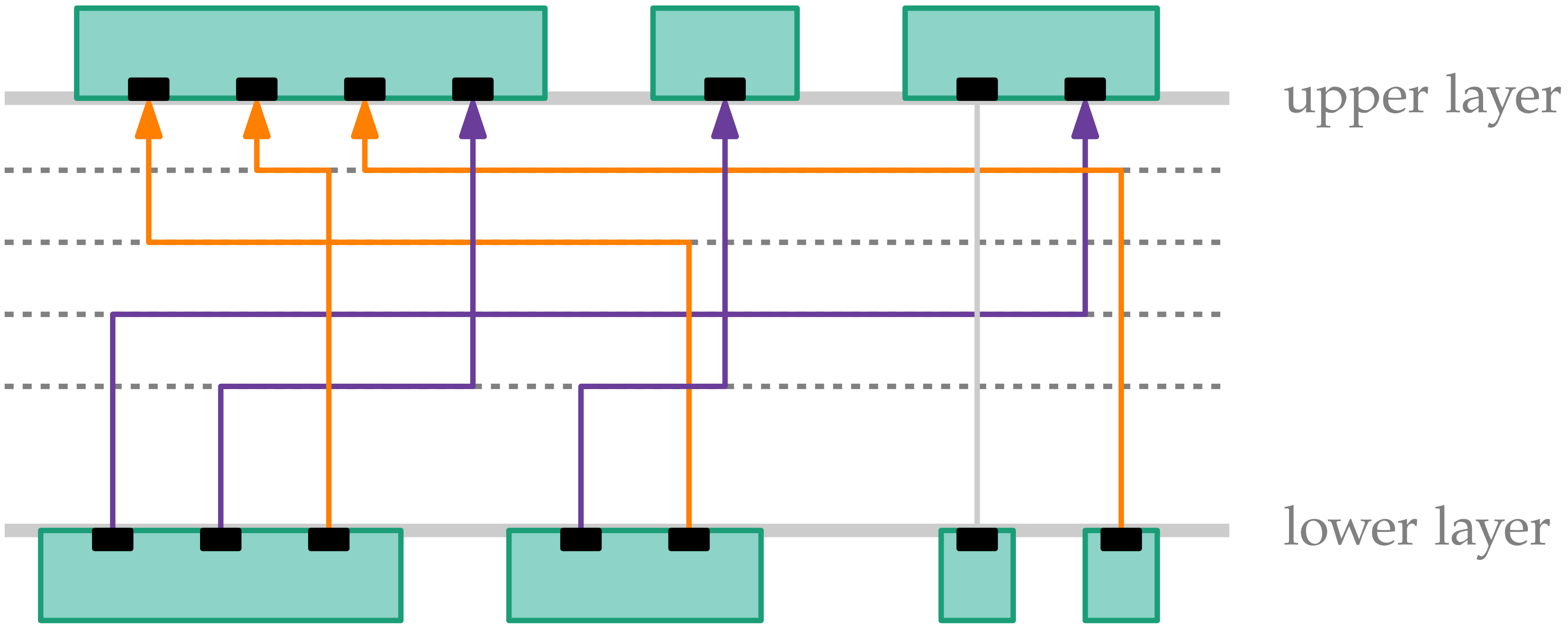
Motivation – Layered Orthogonal Edge Routing

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



Motivation – Layered Orthogonal Edge Routing

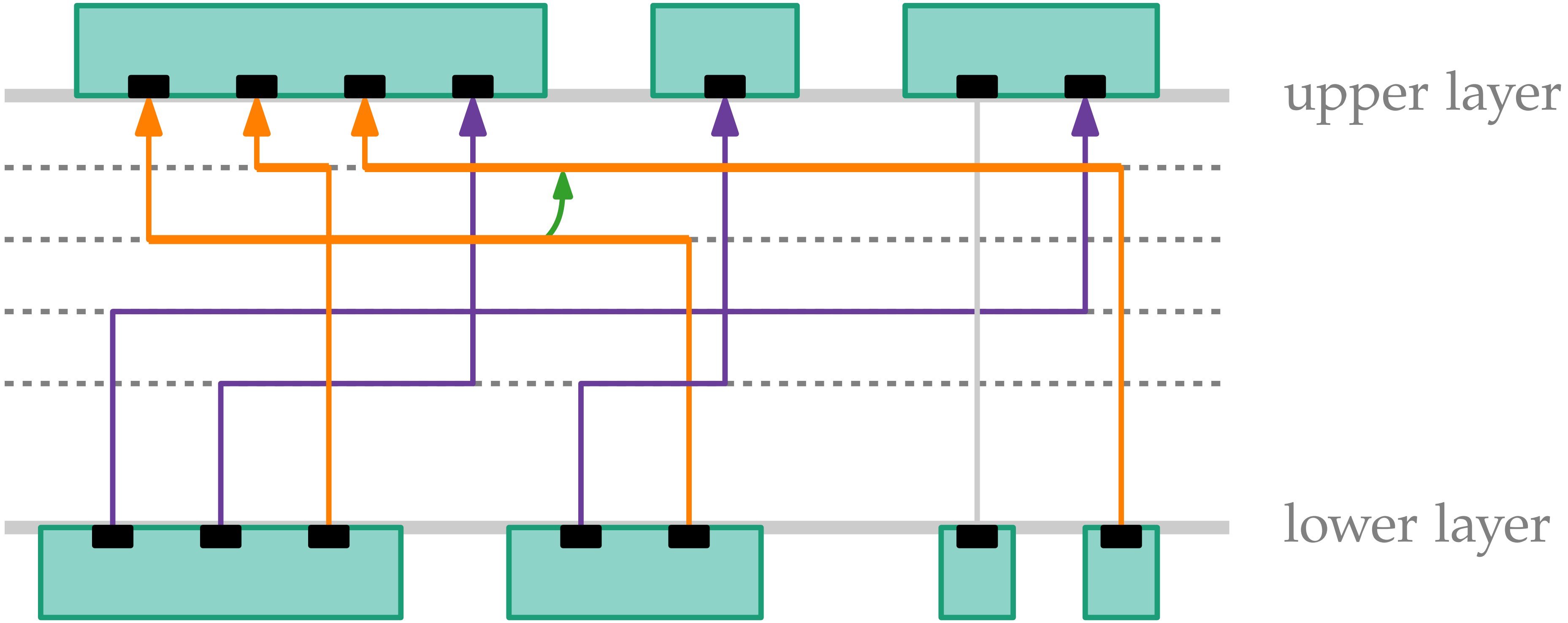
- distinguish between *left-going* and *right-going* edges
- only edges going in the same direction and overlapping partially in x-dimension can cross twice



Motivation – Layered Orthogonal Edge Routing

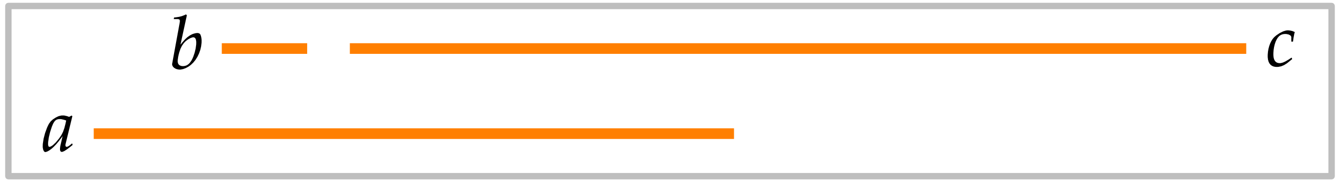
- distinguish between *left-going* and *right-going* edges
- only edges going in the same direction and overlapping partially in x-dimension can cross twice

  ⇒ induce a vertical order for the horizontal middle segments



Definition – Directional Interval Graphs

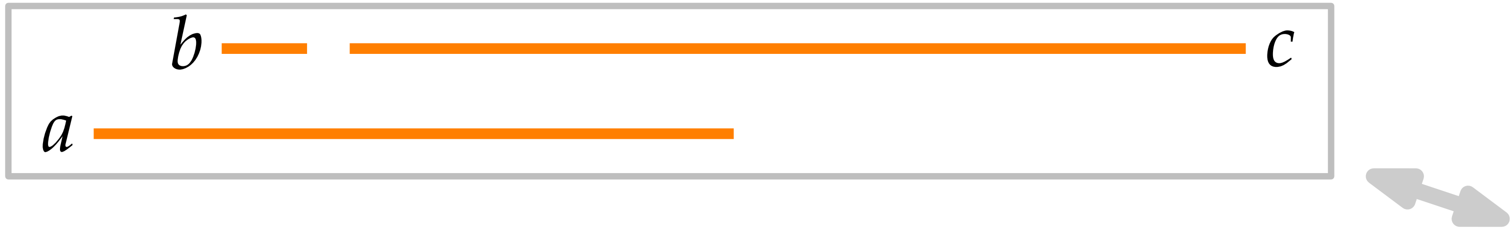
Interval representation: set of intervals



Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

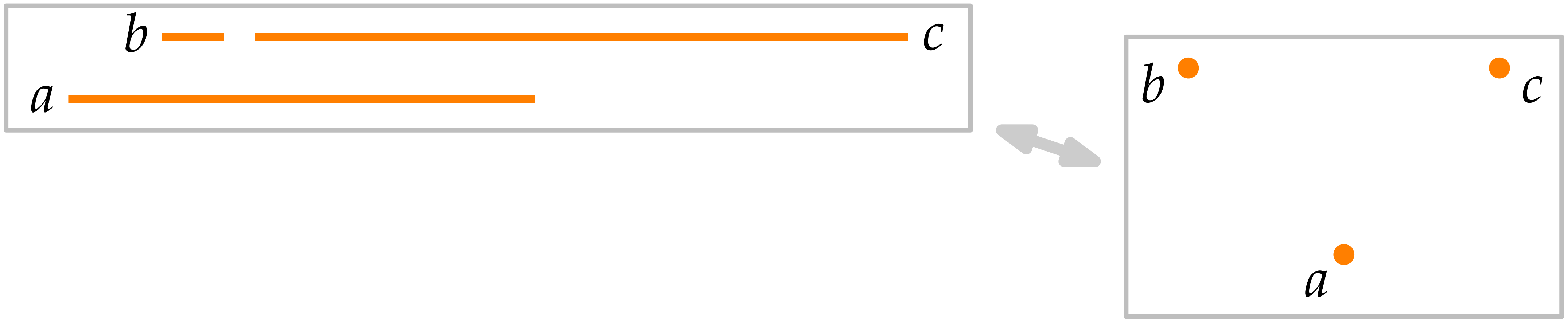


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

- vertex for each interval

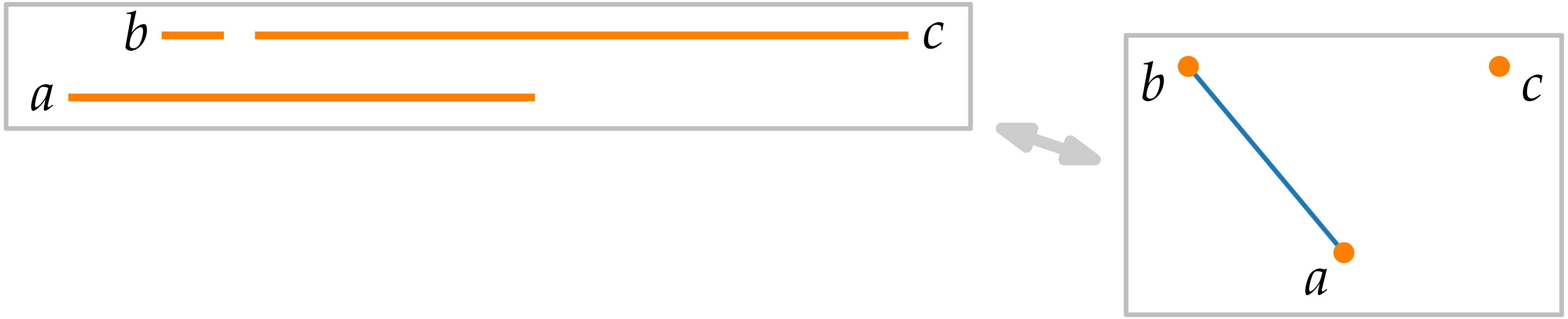


Definition – Directional Interval Graphs

Interval representation: set of intervals

Directional interval graph:

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

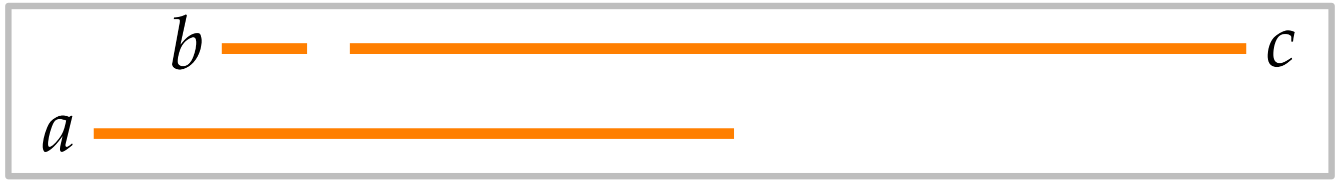


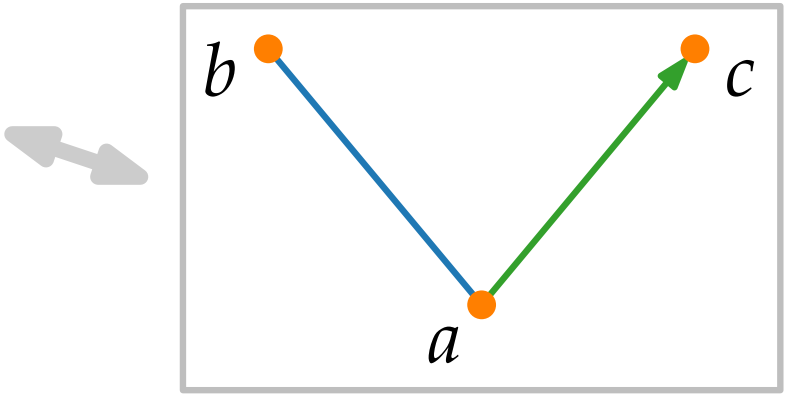
Definition – Directional Interval Graphs

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



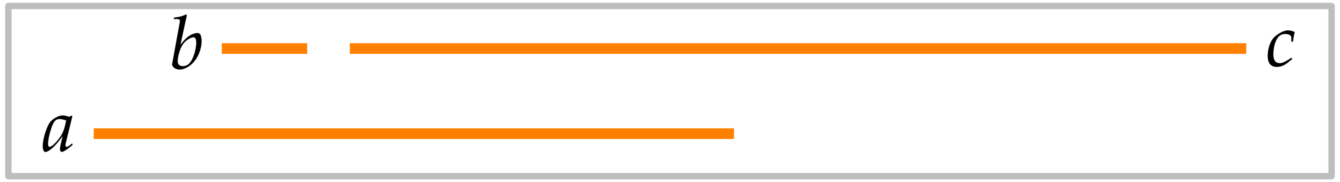


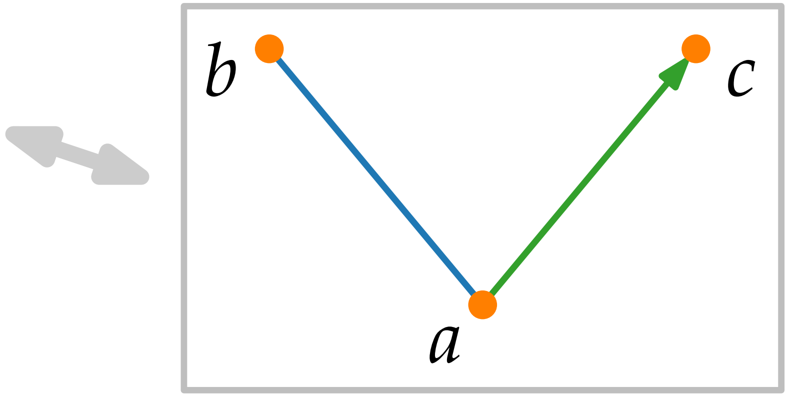
Definition – Directional Interval Graphs

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





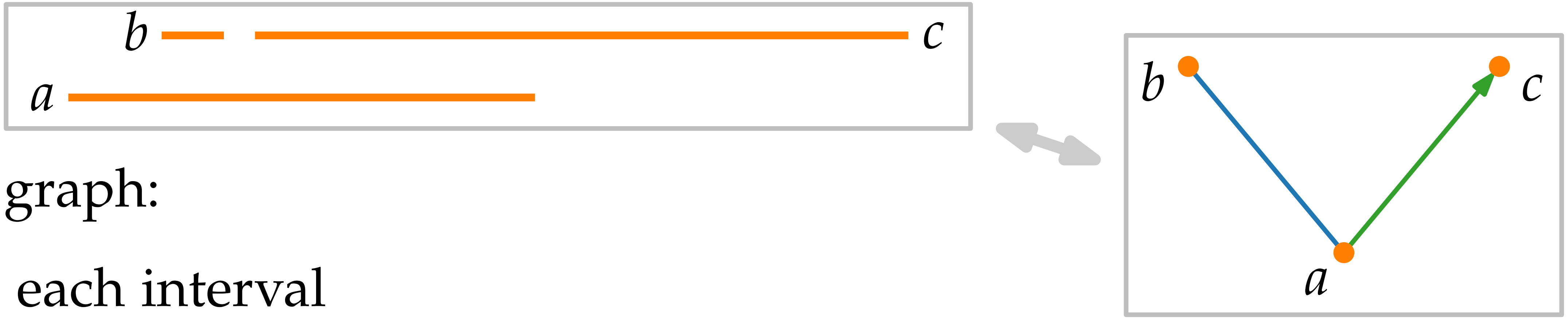
Mixed interval graph:

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:

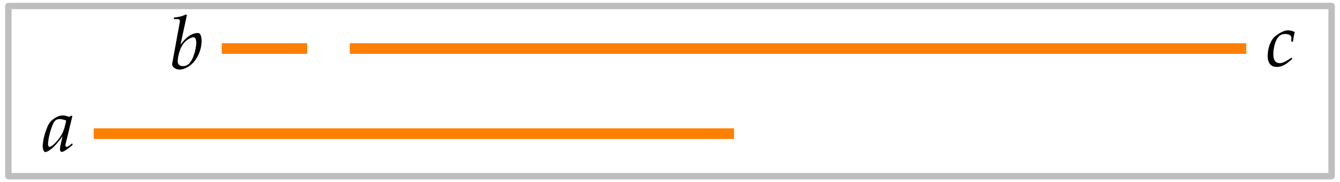
- vertex for each interval

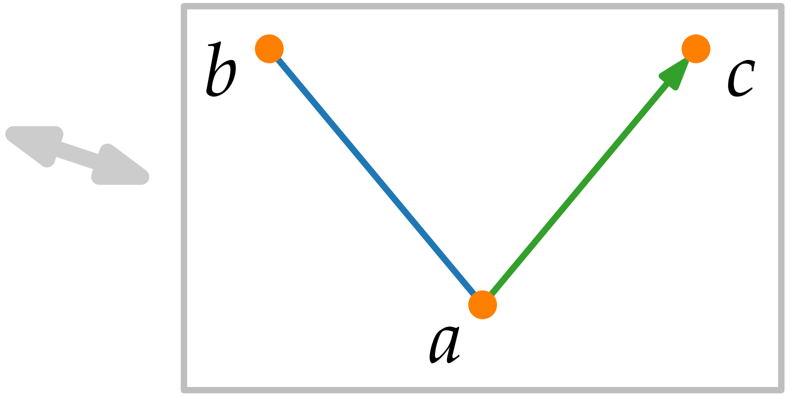
Definition – Directional Interval Graphs

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]

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

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

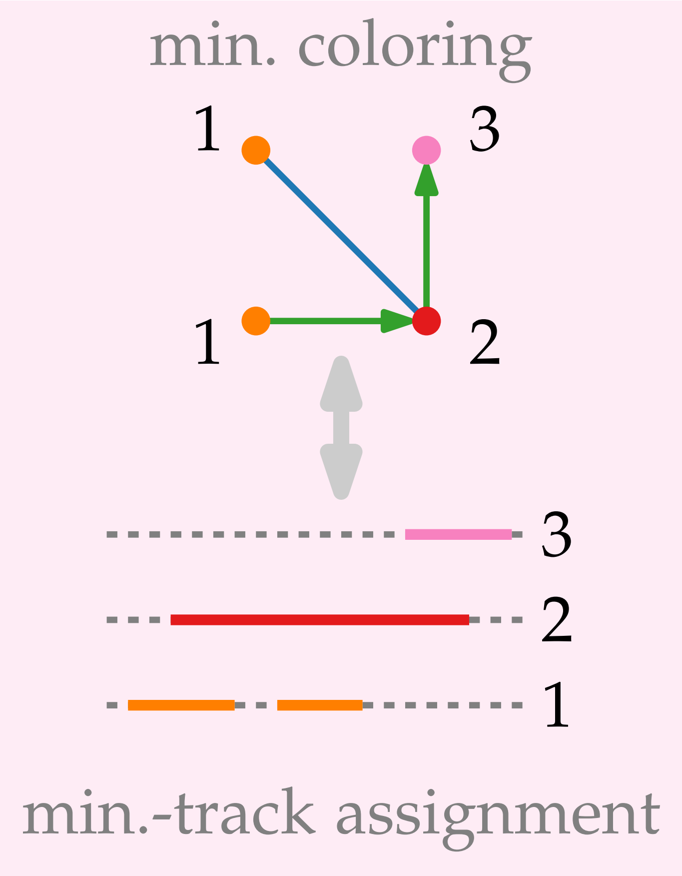
- coloring in linear time by a greedy algorithm

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

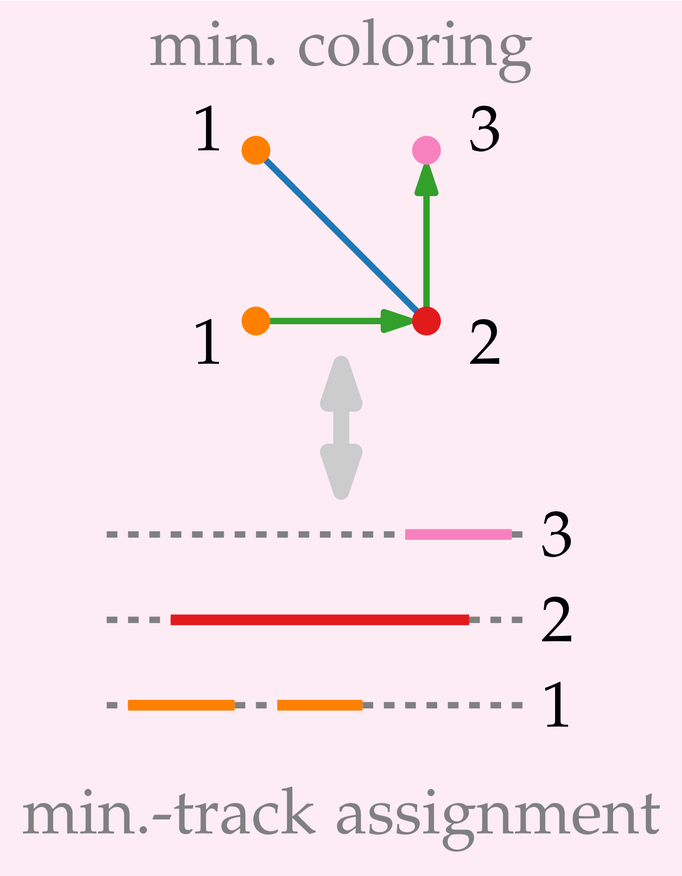
Directional interval graphs:

- 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:

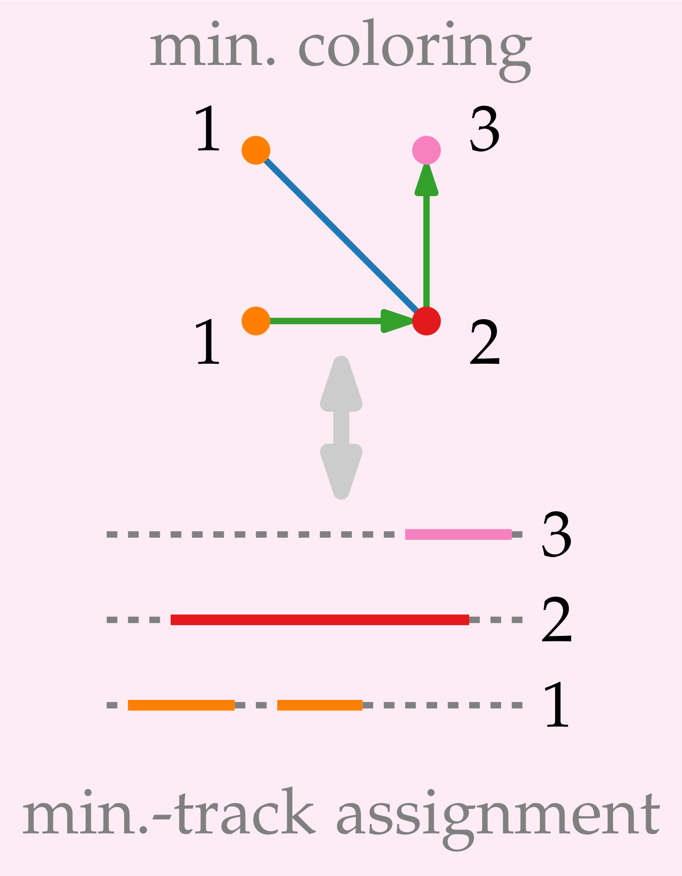
- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

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**

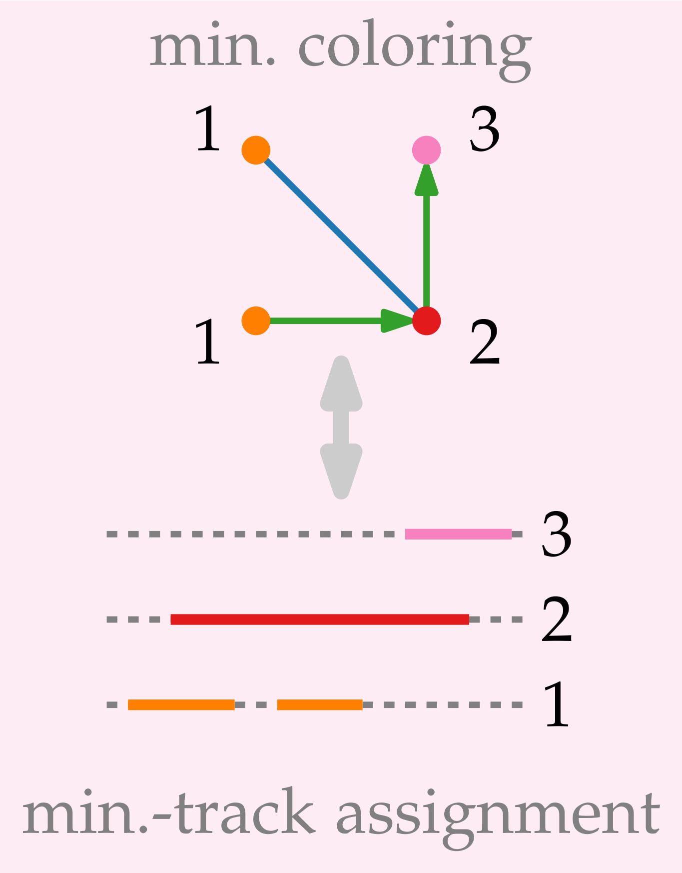
- recognition in $O(n^2)$ time
- coloring in $O(n \log n)$ time by a greedy algorithm

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

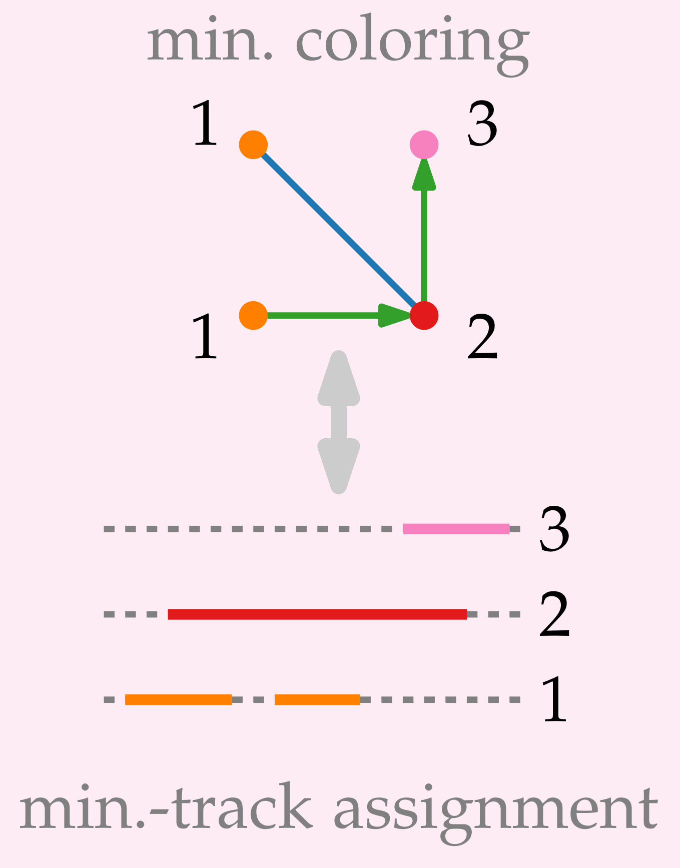
Mixed interval graphs:

- coloring is NP-complete

agenda for this talk

Directed graphs (only directed edges):

- coloring in linear time using topological sorting



$n :=$ # 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:`

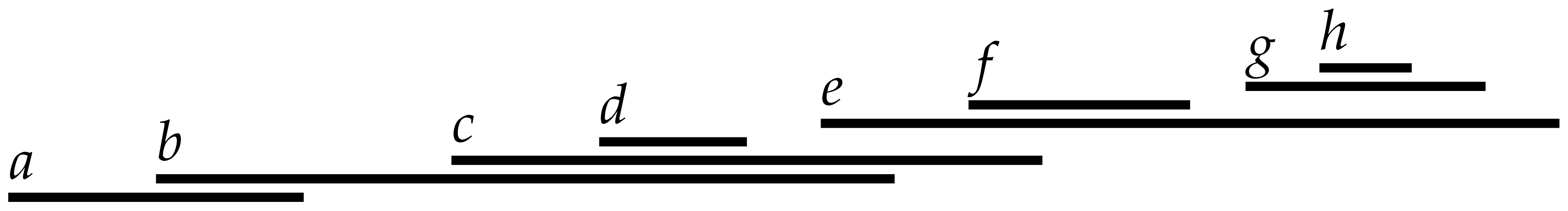
1. sort all intervals by left endpoint
2. for each interval, assign the smallest available color respecting incident edges

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

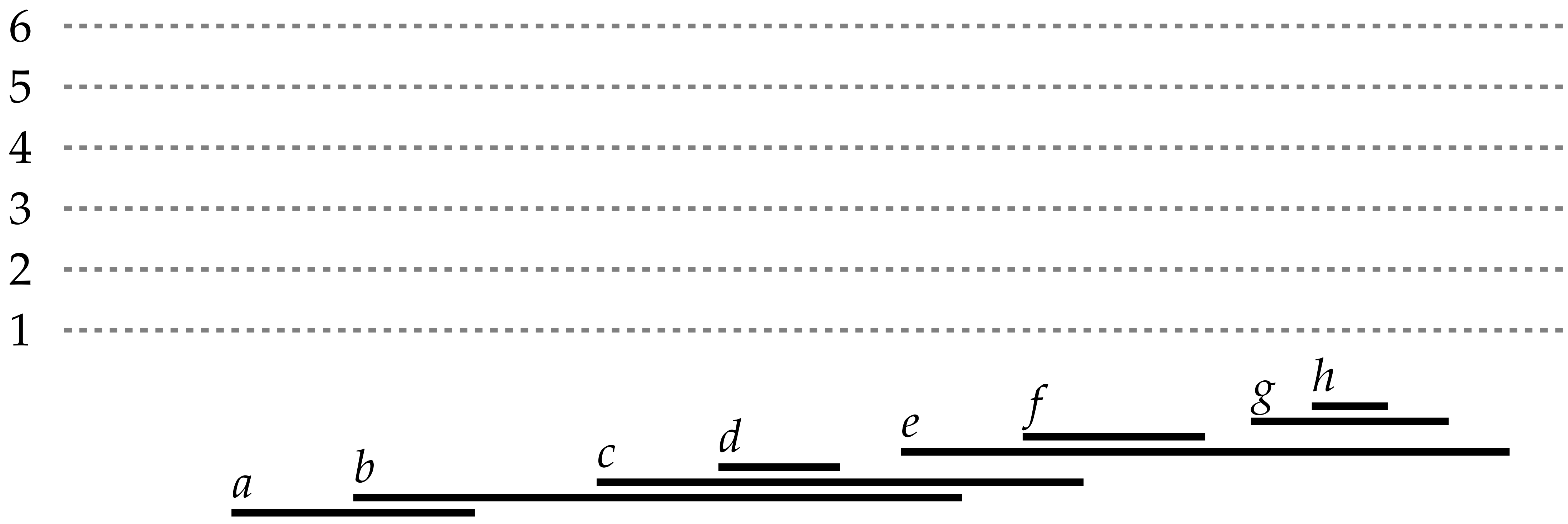


Coloring Directional Interval Graphs

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

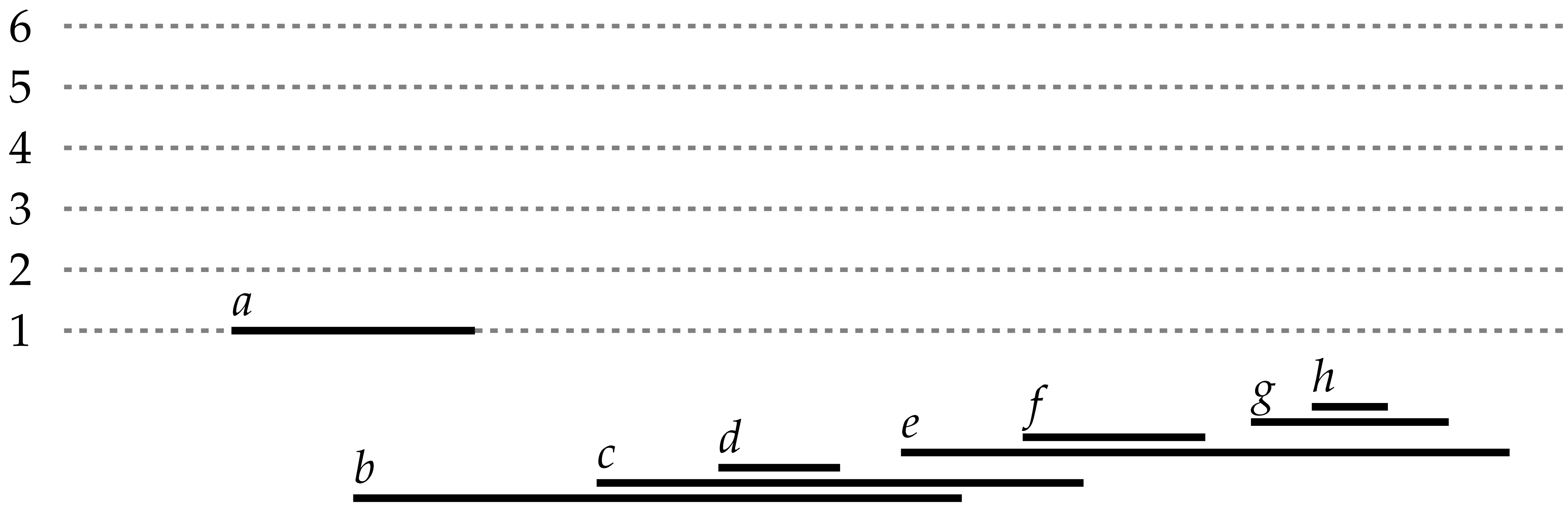


Coloring Directional Interval Graphs

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

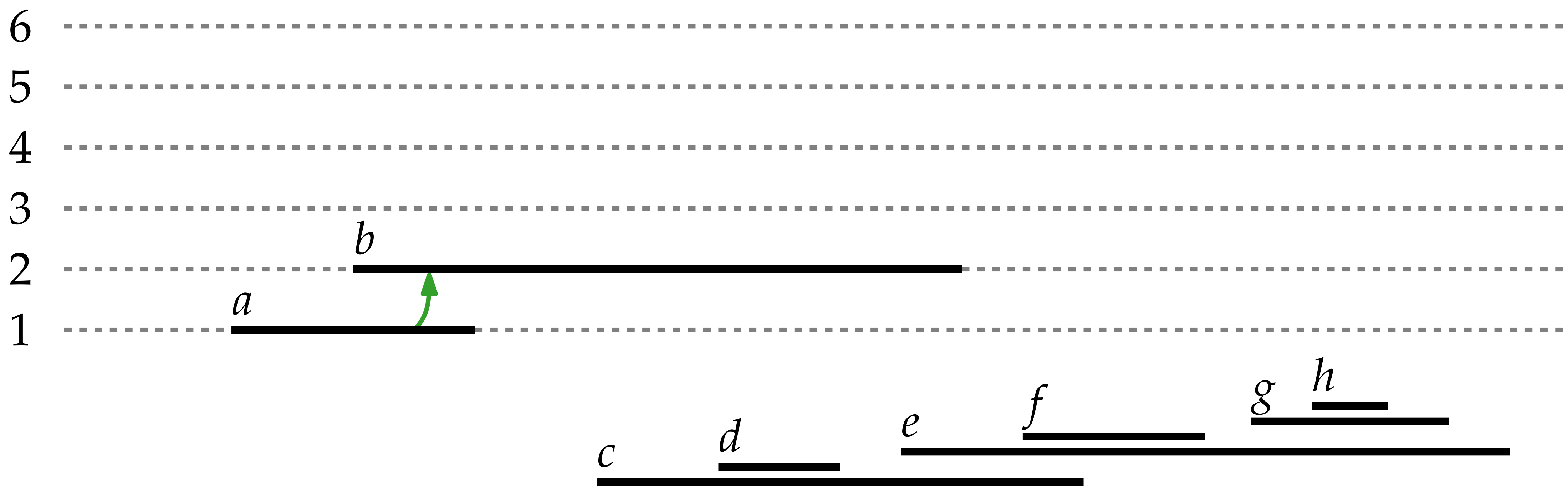


Coloring Directional Interval Graphs

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

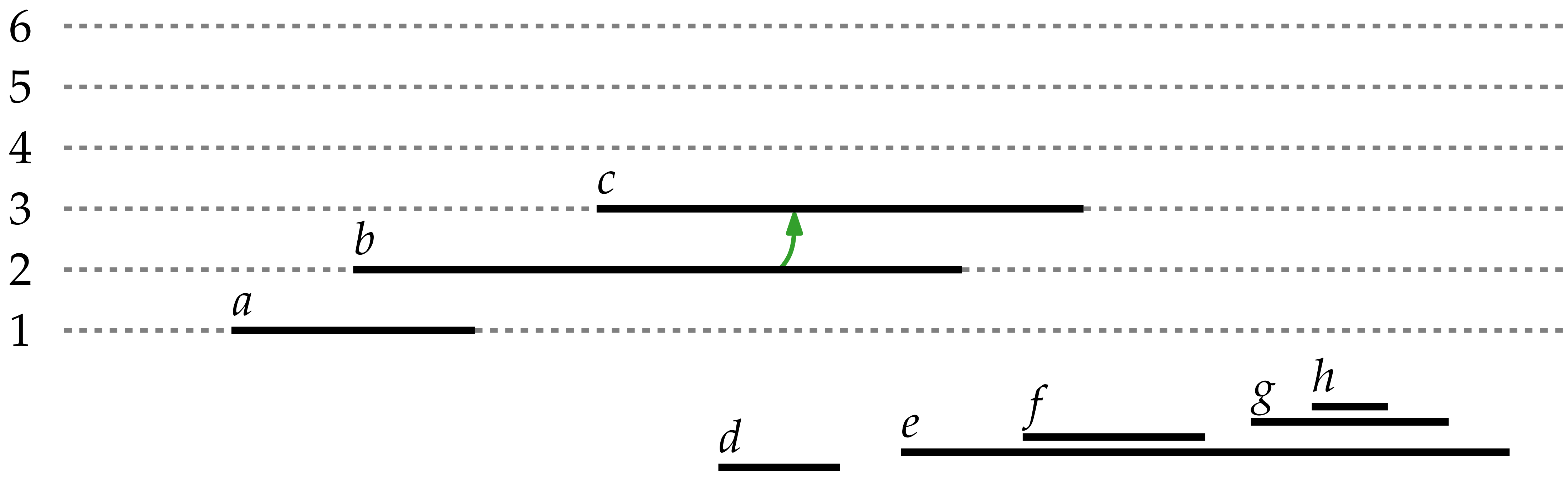


Coloring Directional Interval Graphs

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

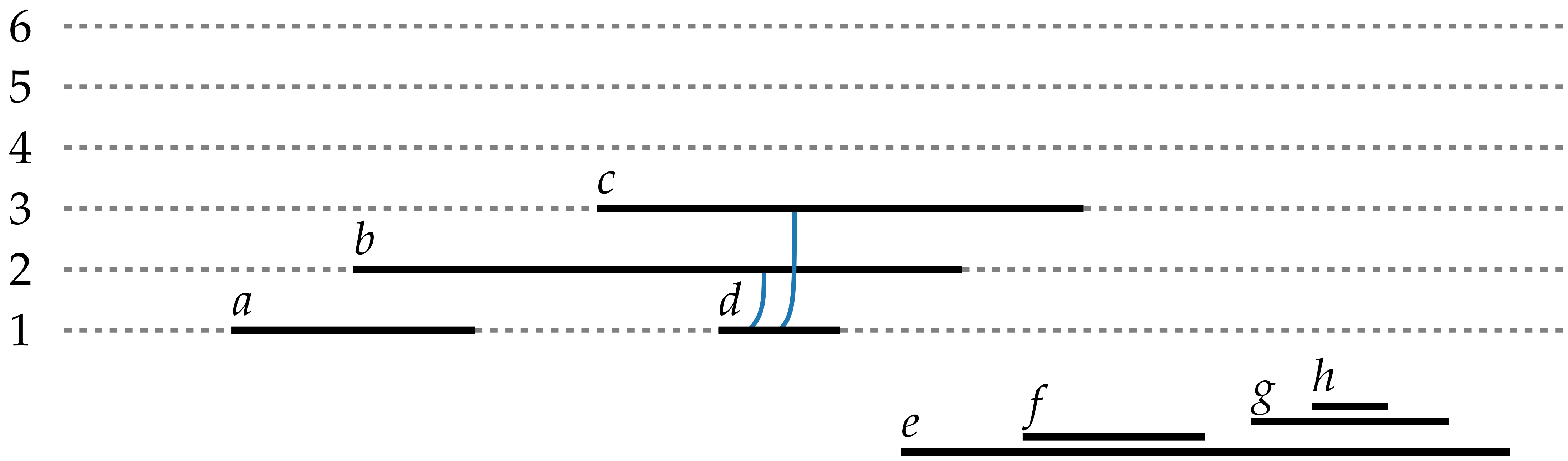


Coloring Directional Interval Graphs

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

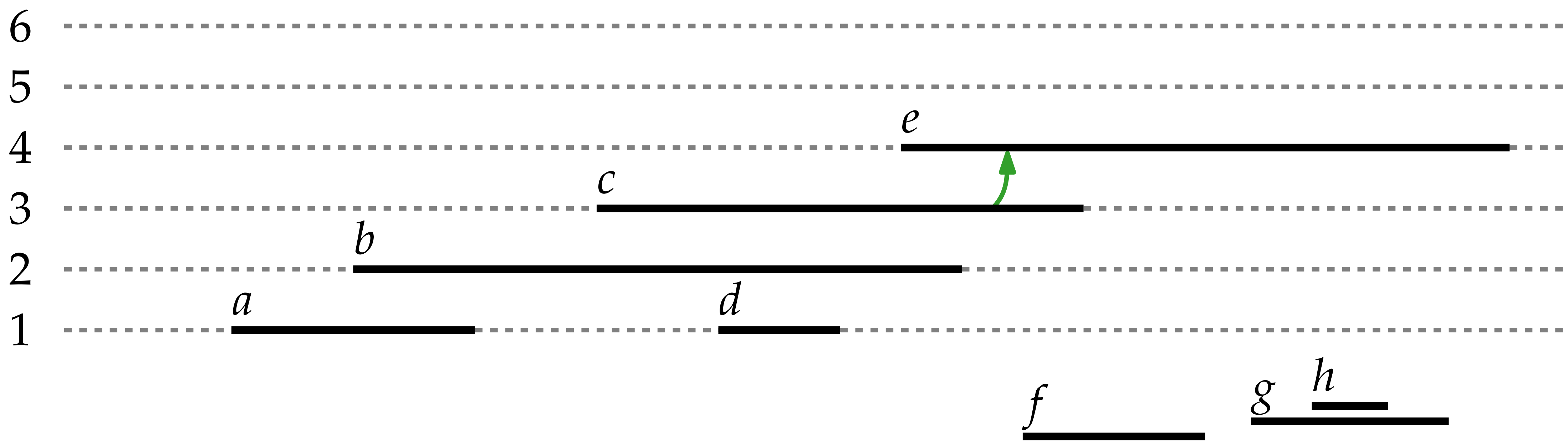


Coloring Directional Interval Graphs

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

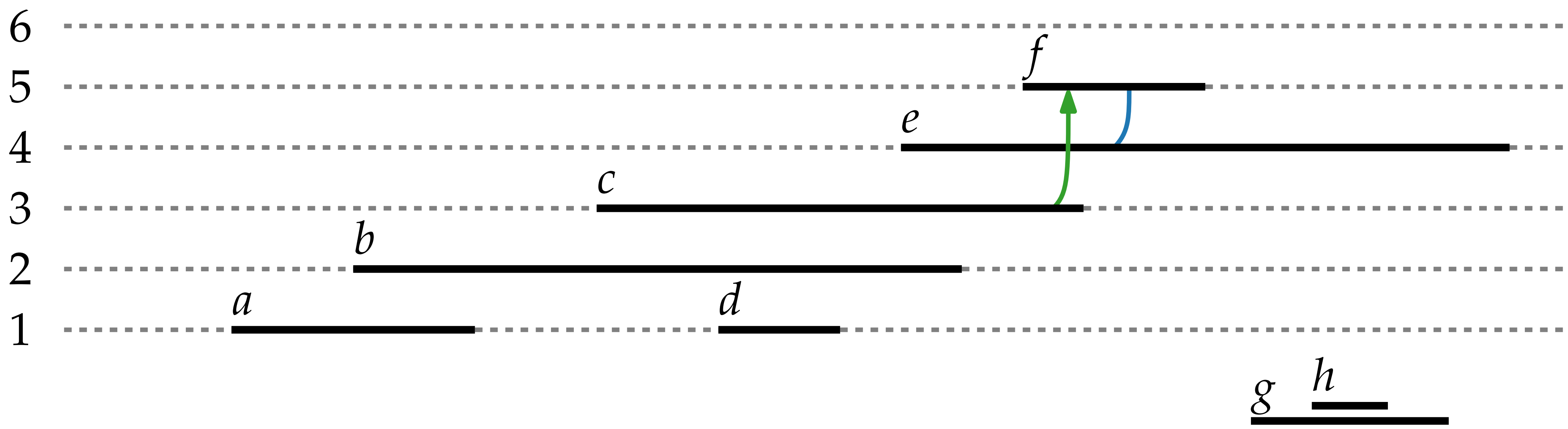


Coloring Directional Interval Graphs

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

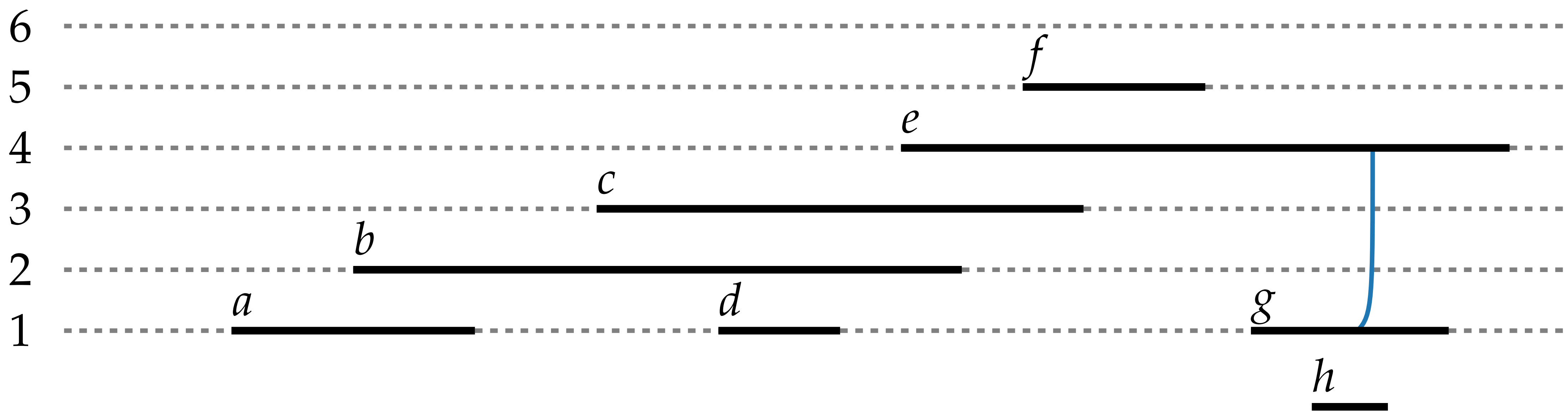


Coloring Directional Interval Graphs

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

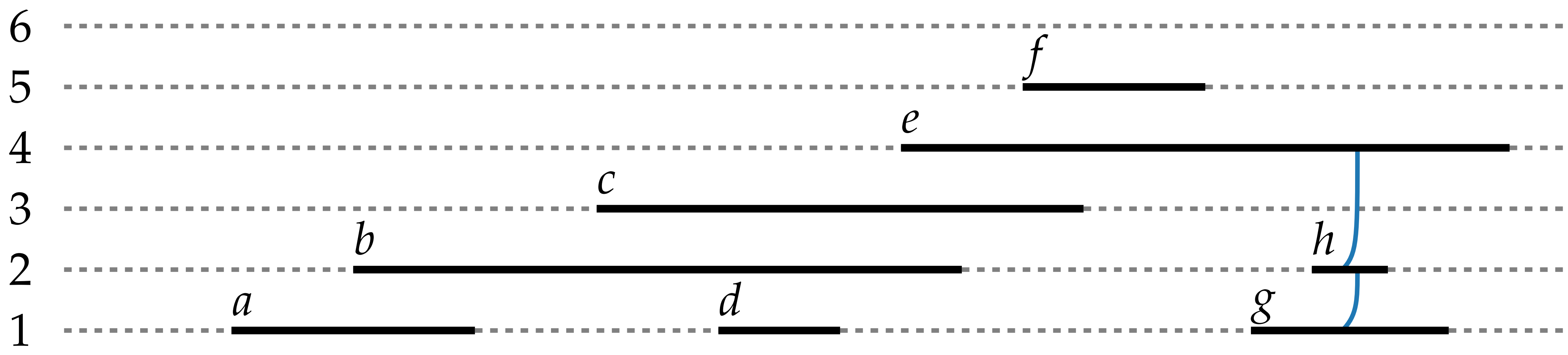


Coloring Directional Interval Graphs

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

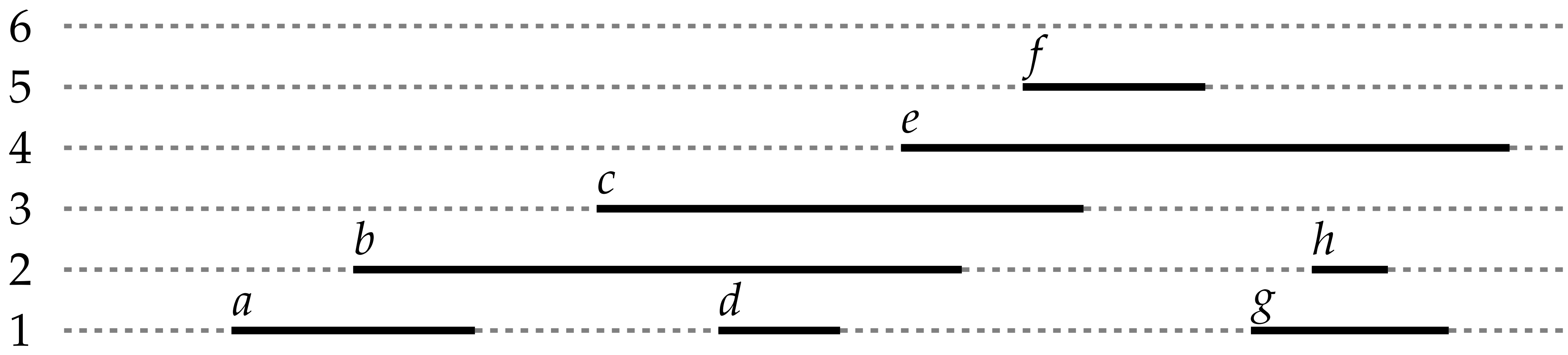


Coloring Directional Interval Graphs

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:

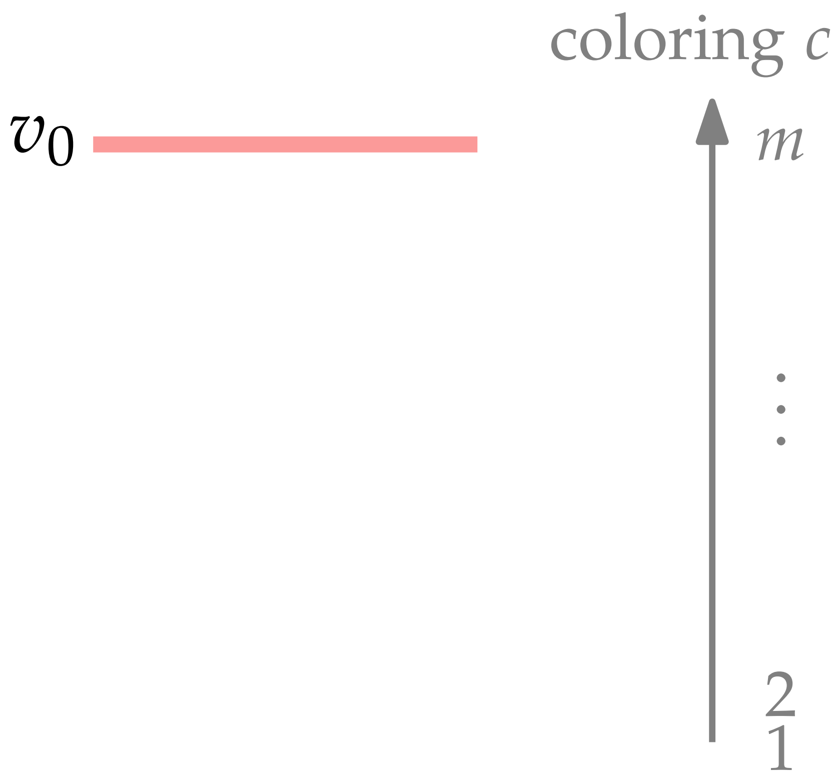
- Let v_0 be an interval of maximum color, i.e., $c(v_0) = m$.

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., $c(v_0) = m$.

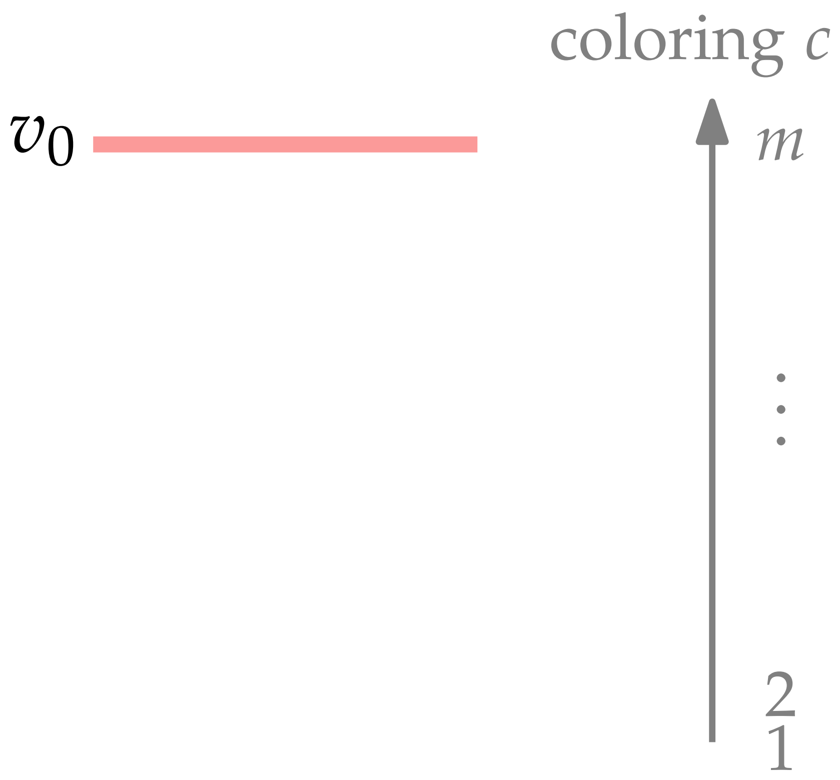


Coloring Directional Interval Graphs

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.

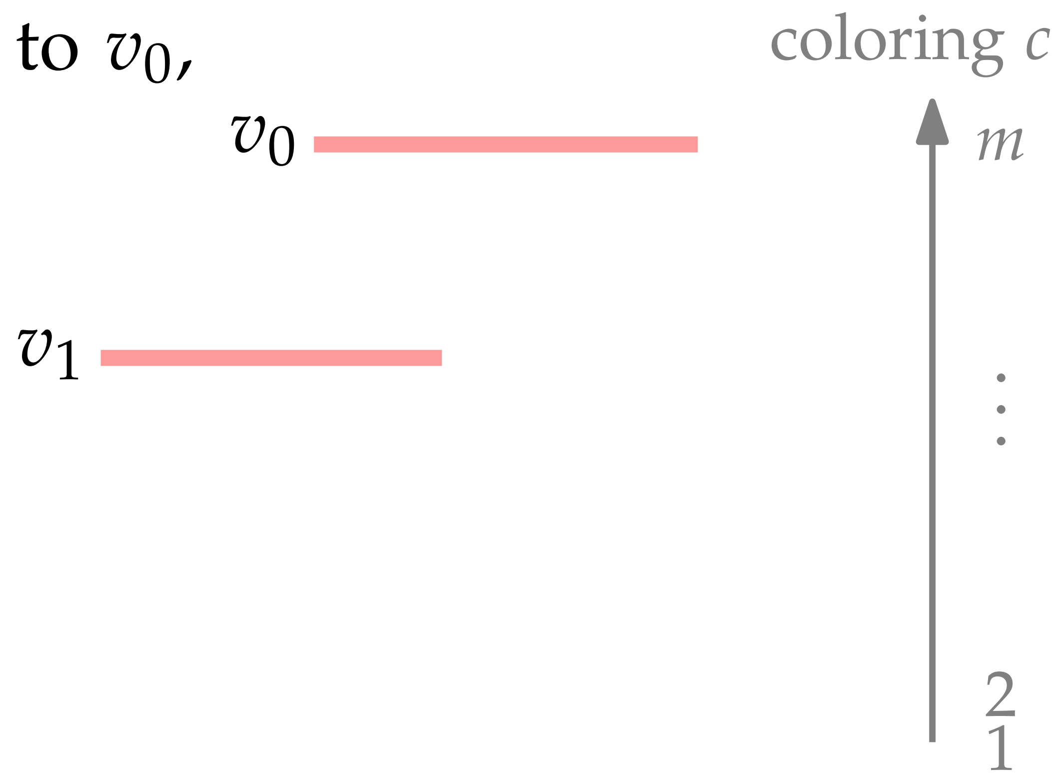


Coloring Directional Interval Graphs

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.

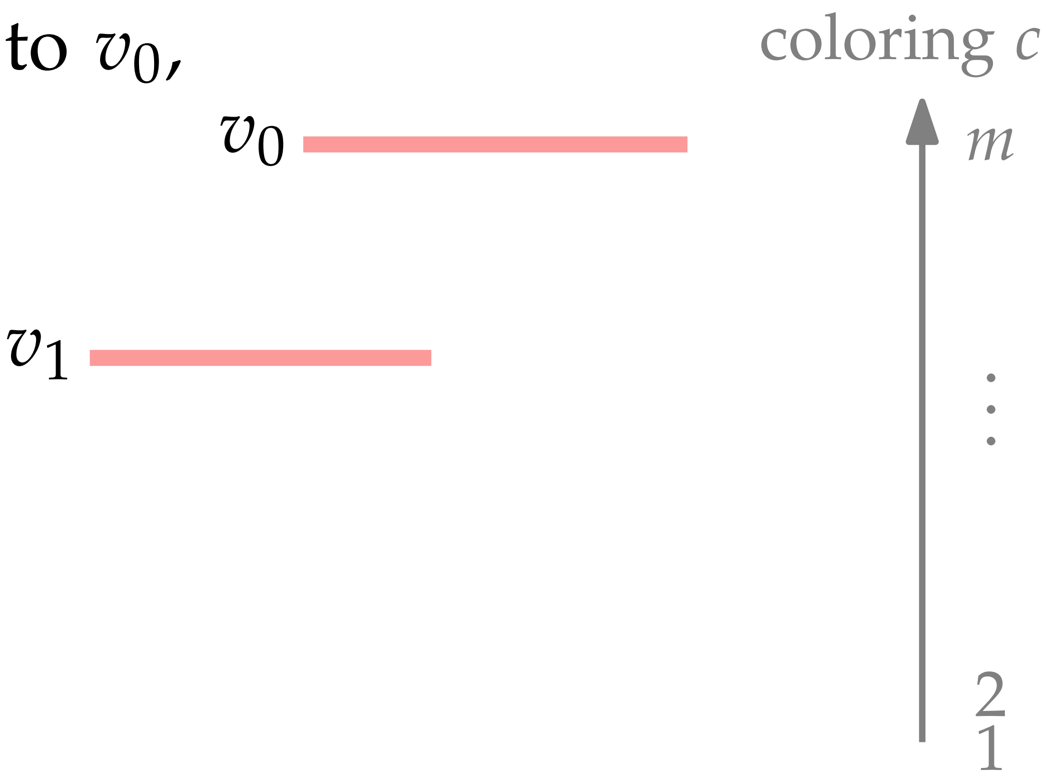


Coloring Directional Interval Graphs

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.

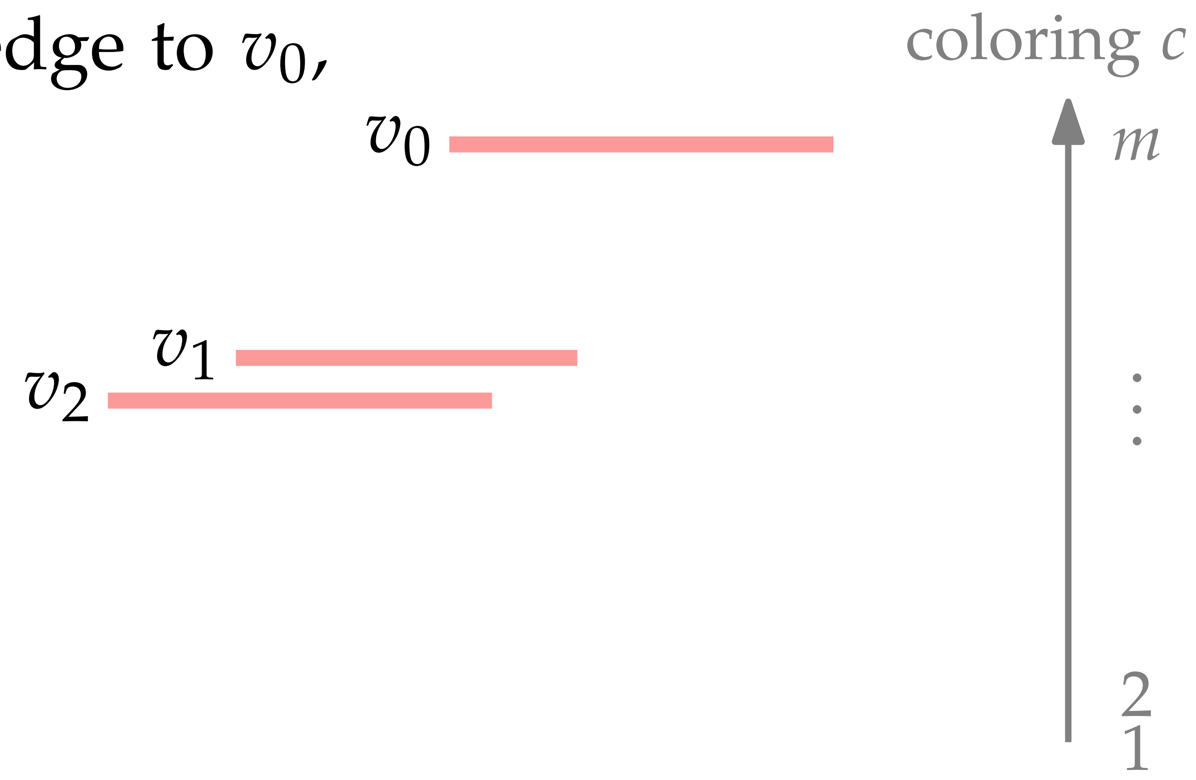


Coloring Directional Interval Graphs

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.

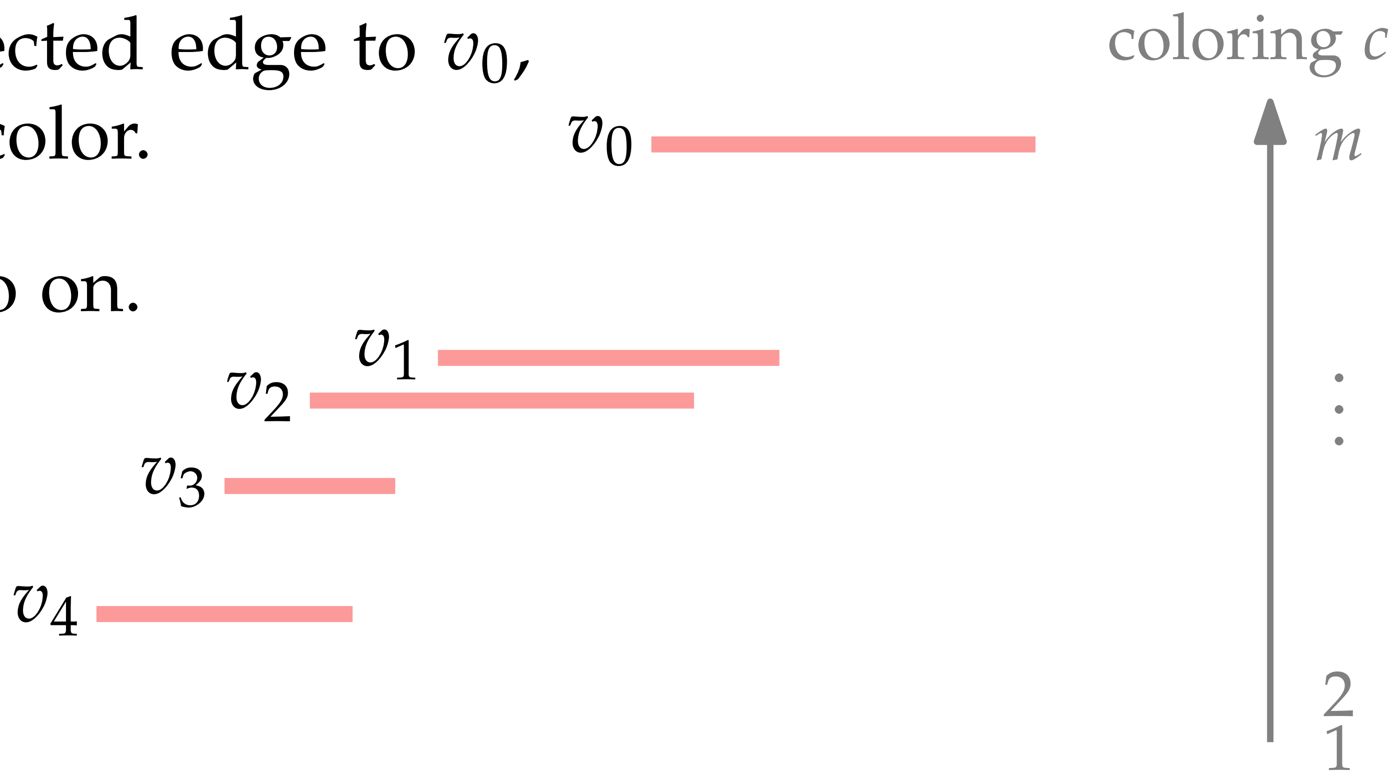


Coloring Directional Interval Graphs

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.

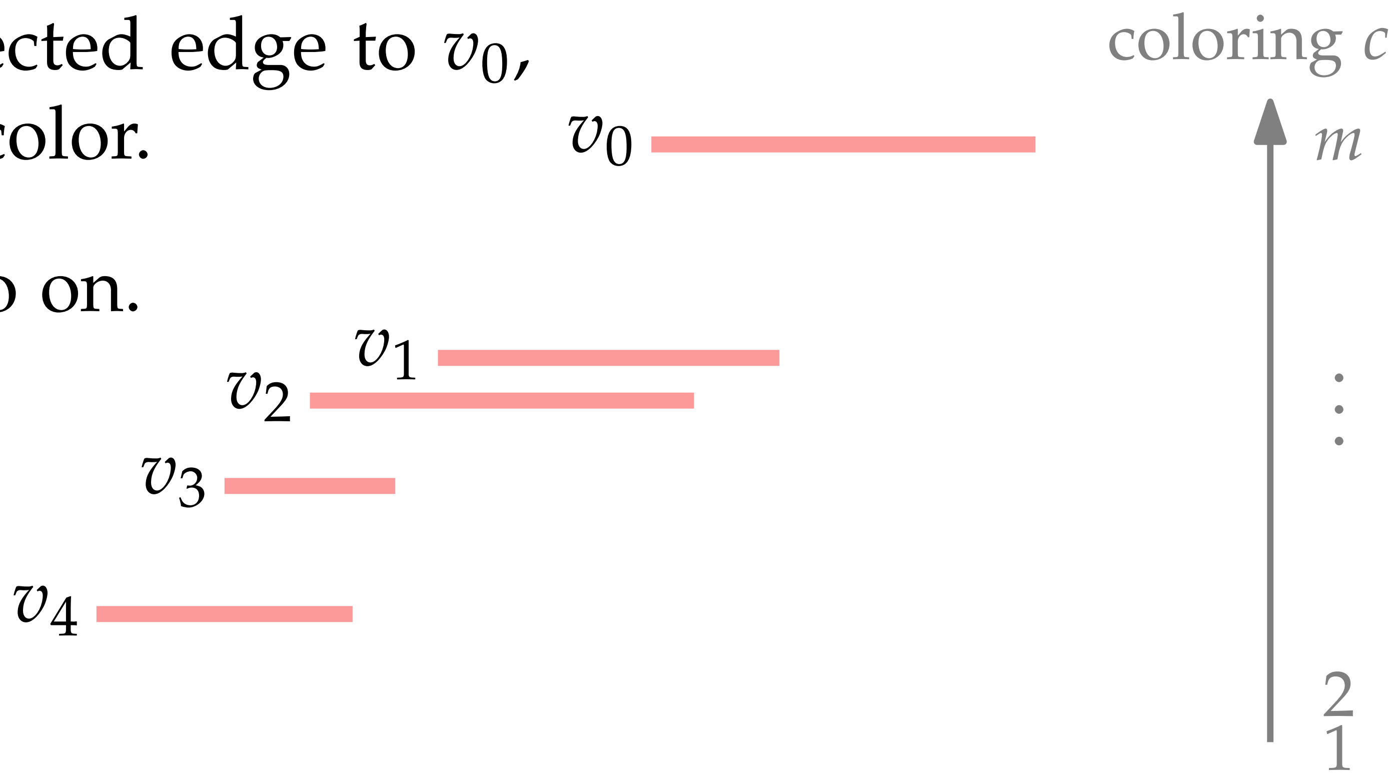


Coloring Directional Interval Graphs

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

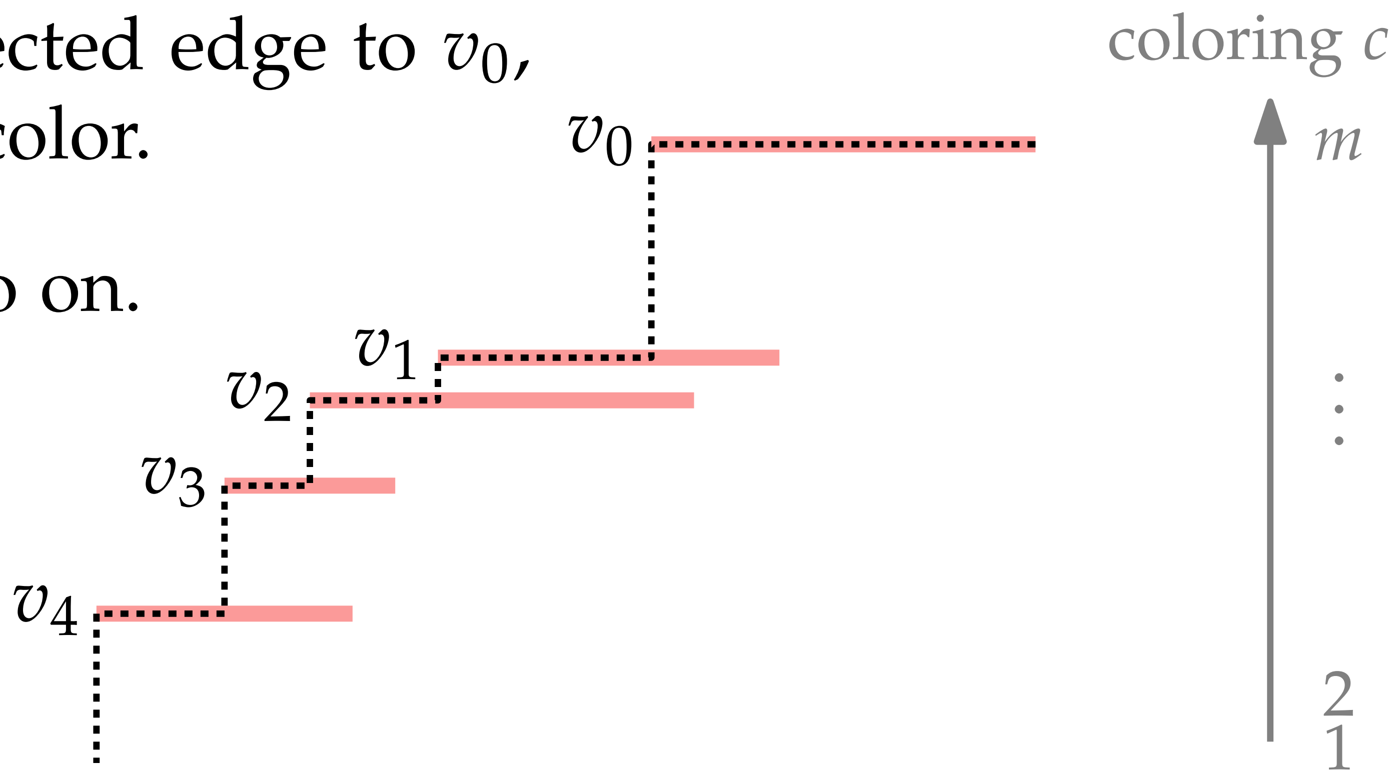


Coloring Directional Interval Graphs

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

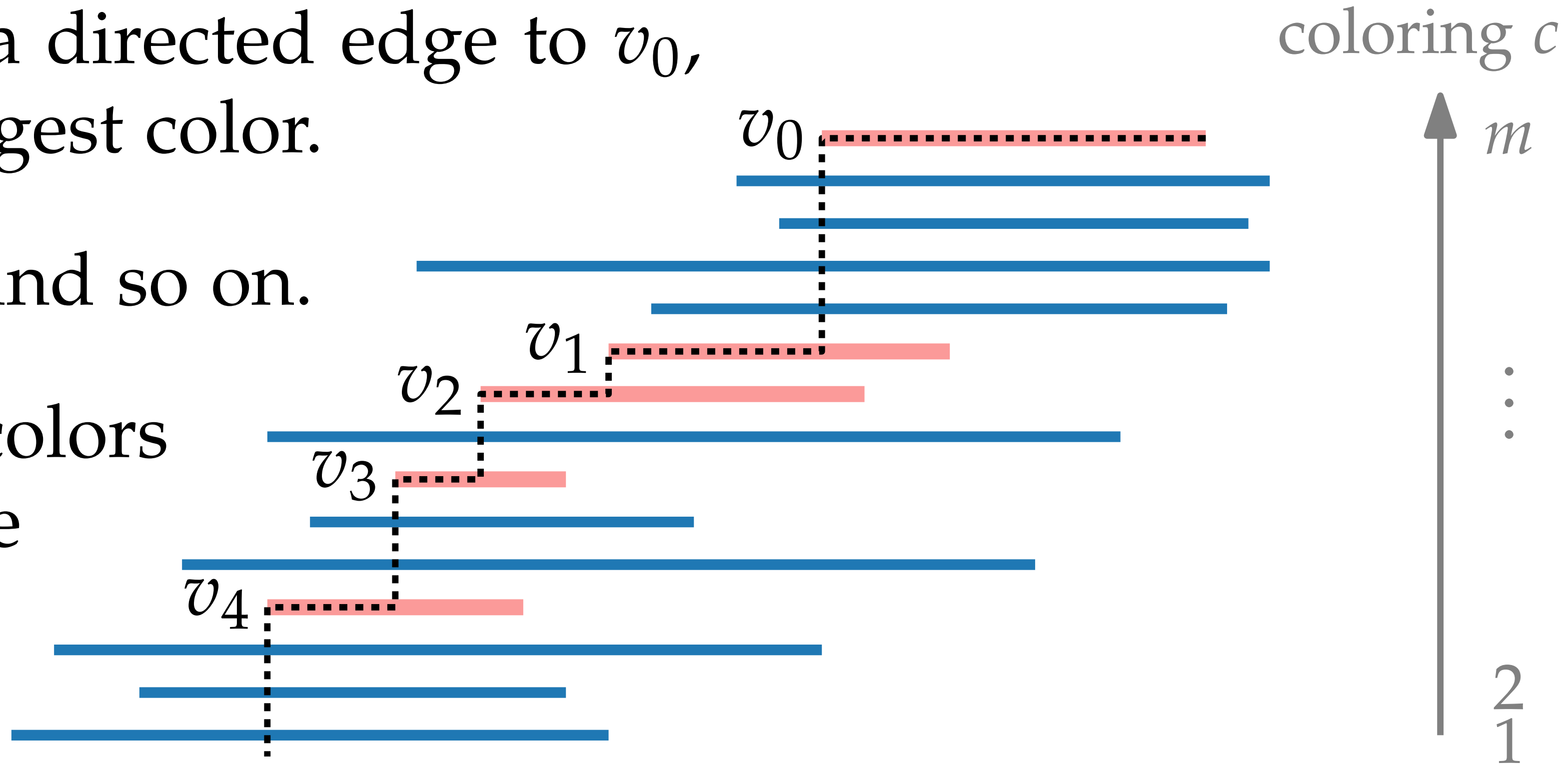


Coloring Directional Interval Graphs

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

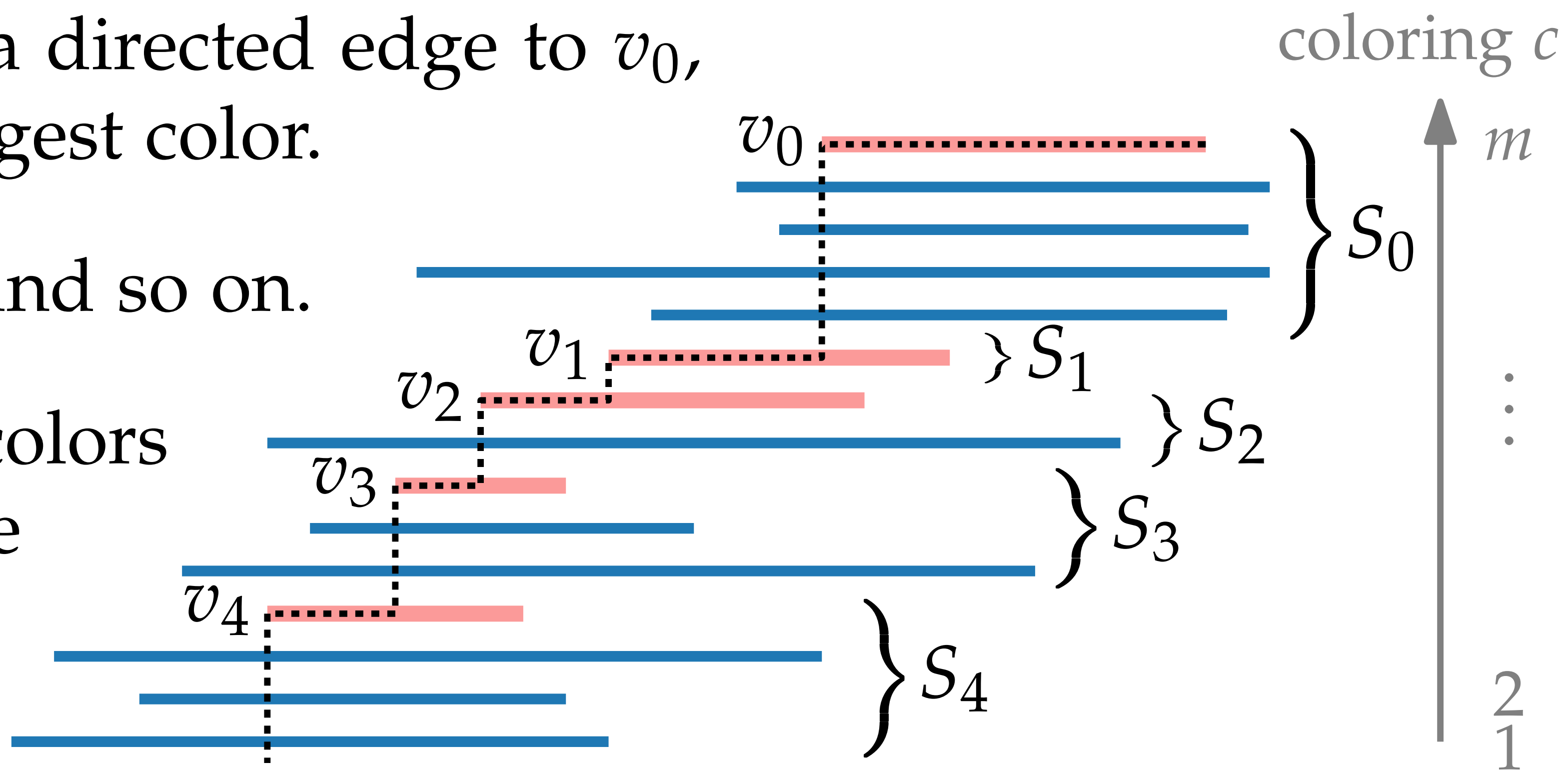


Coloring Directional Interval Graphs

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

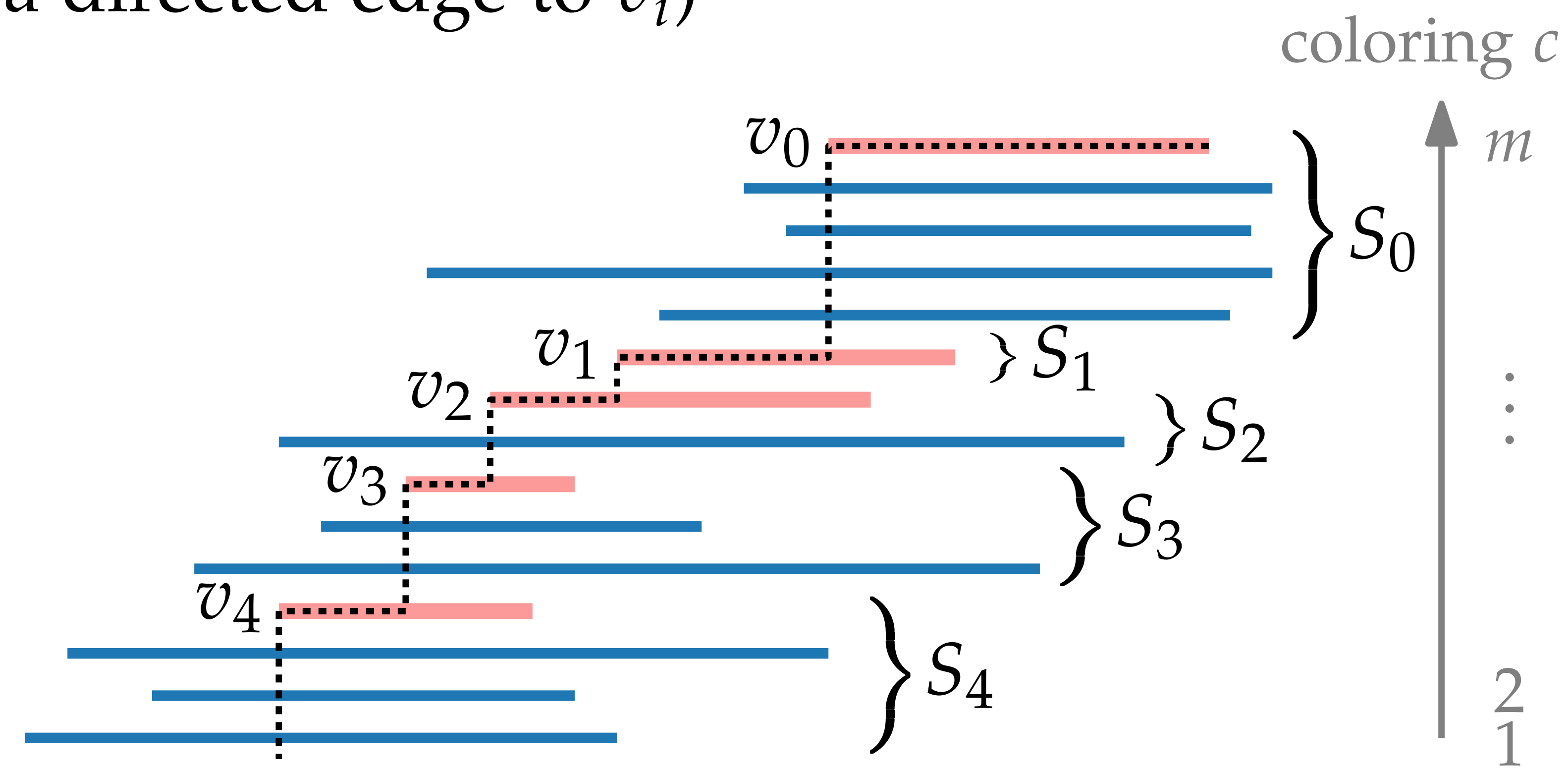


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Clearly, for each $S_i \setminus \{v_i\}$, all intervals contain v_i.
  (otherwise they would have a directed edge to v_i)

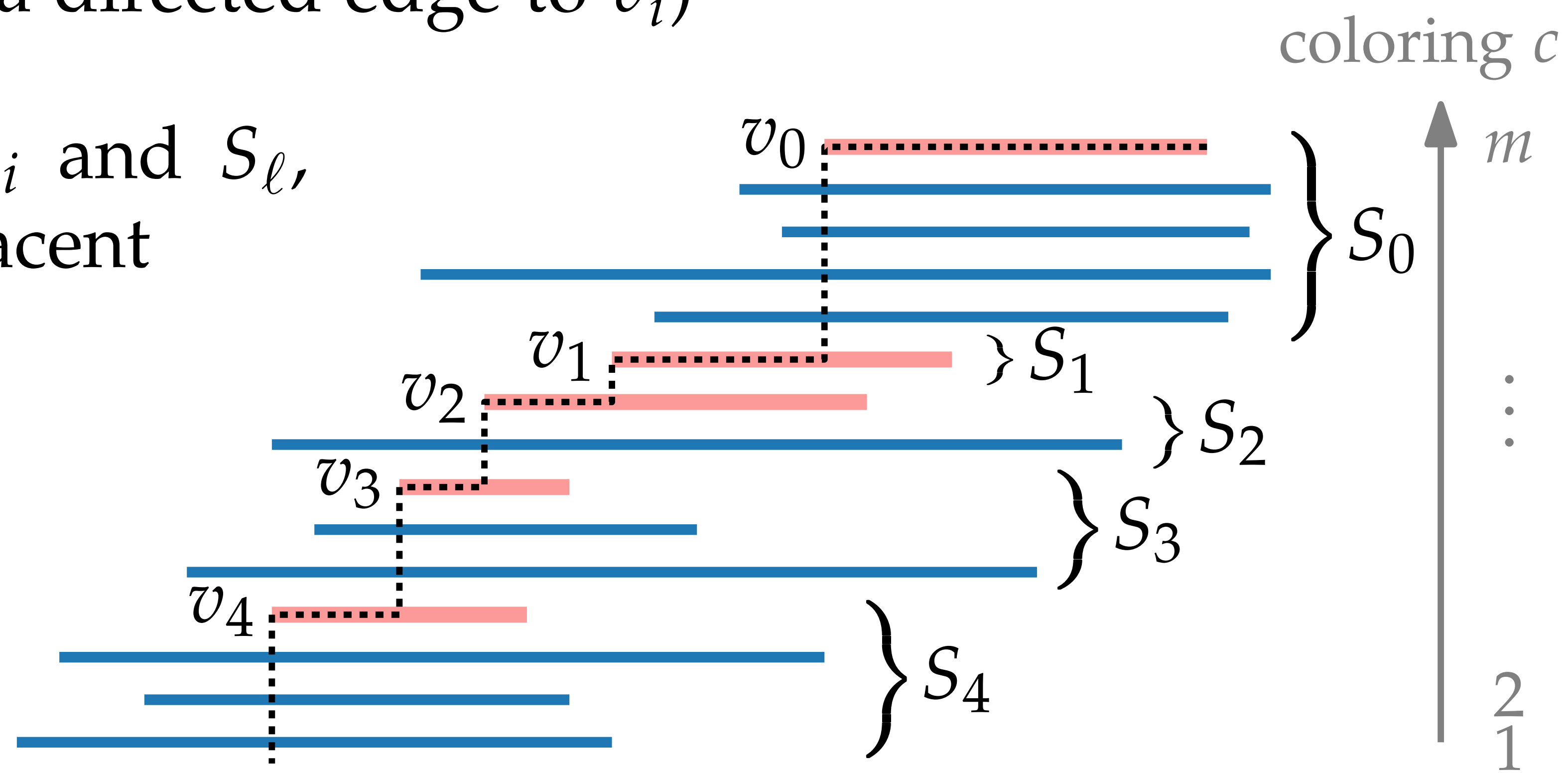


Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Clearly, for each $S_i \setminus \{v_i\}$, 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^+.



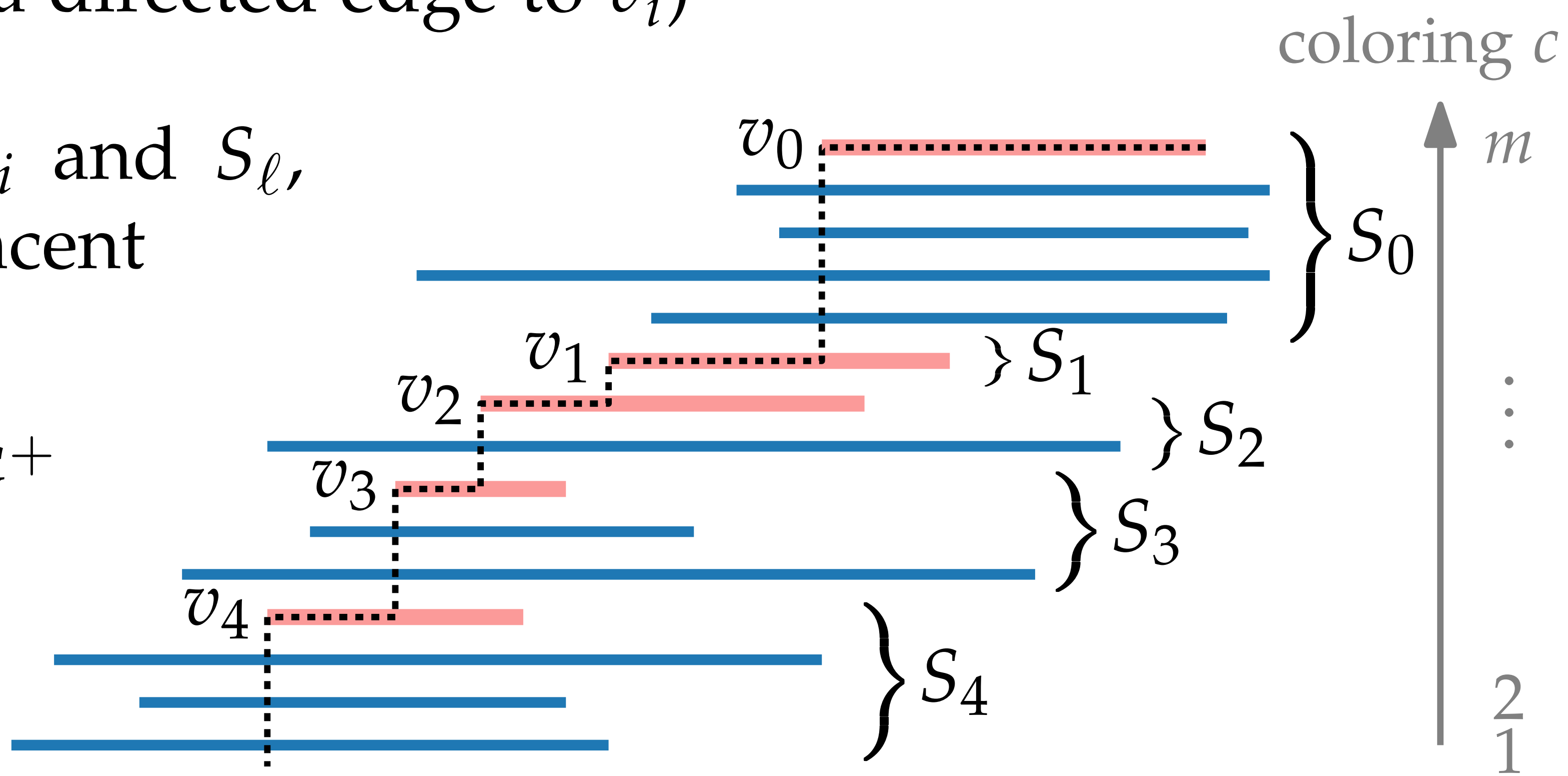
Coloring Directional Interval Graphs

Theorem 1:
A coloring c computed by `GreedyColoring` has the minimum number of colors.

Proof sketch:

- Clearly, for each $S_i \setminus \{v_i\}$, 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.

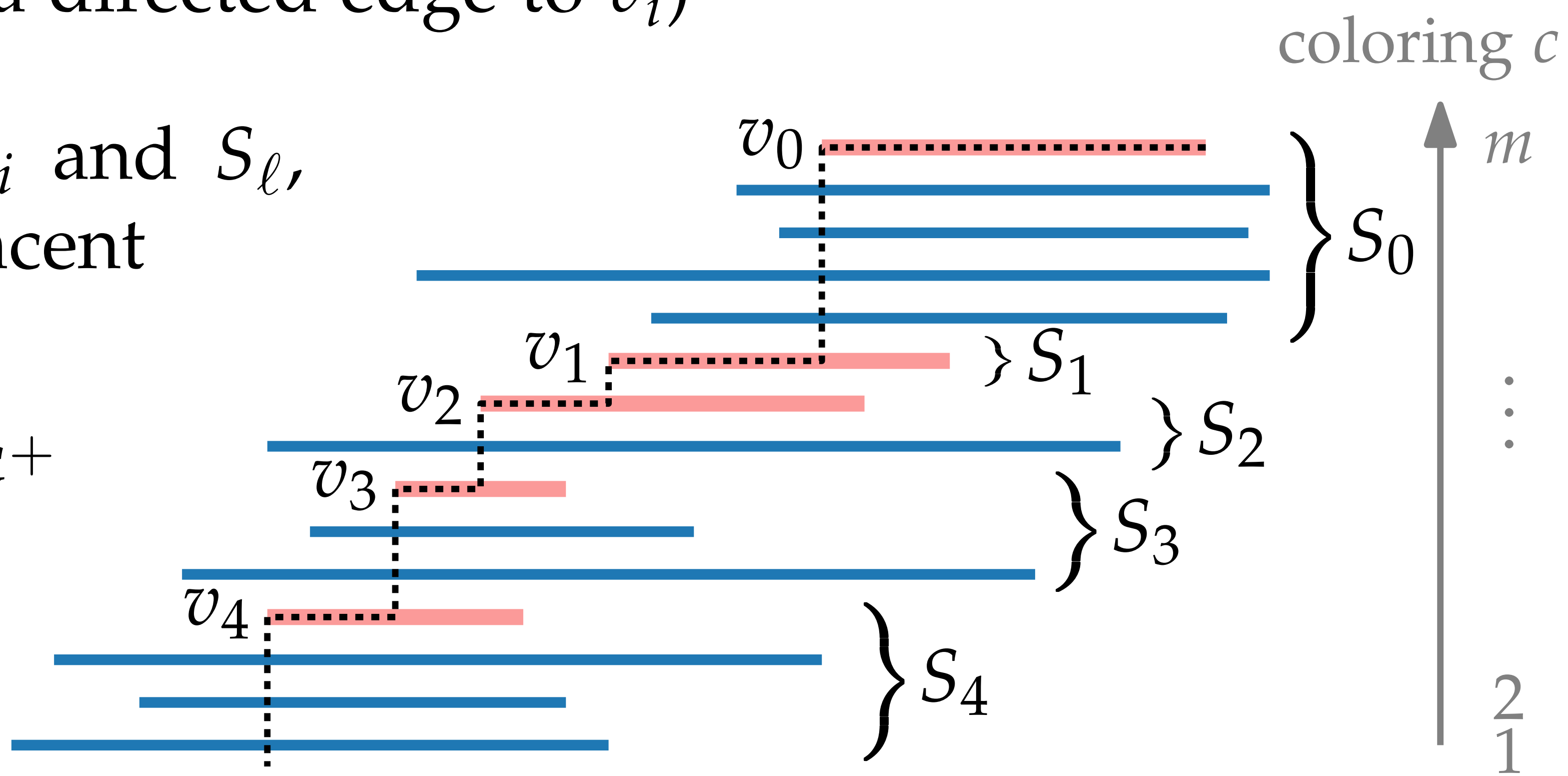
Proof sketch:

- Clearly, for each $S_i \setminus \{v_i\}$, 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^+.

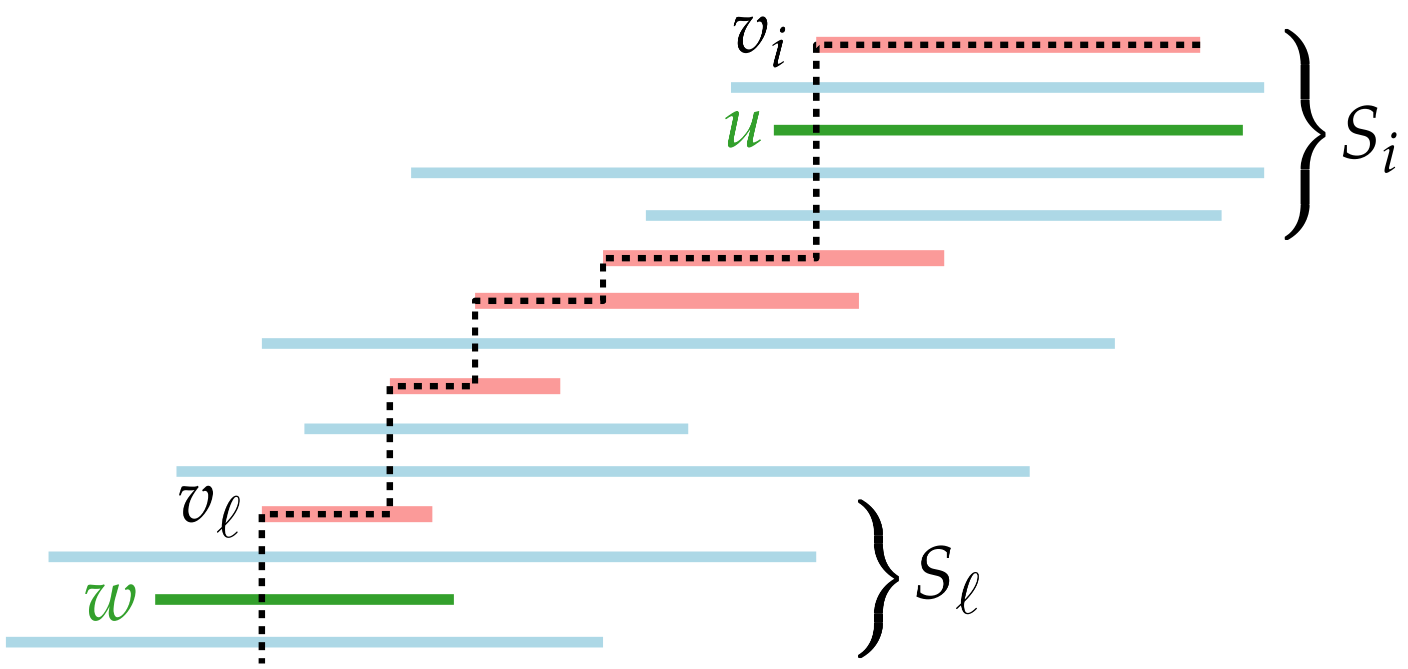
  $\Rightarrow$ $S = \bigcup S_i$ is a clique in G^+

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



Proof of the Claim

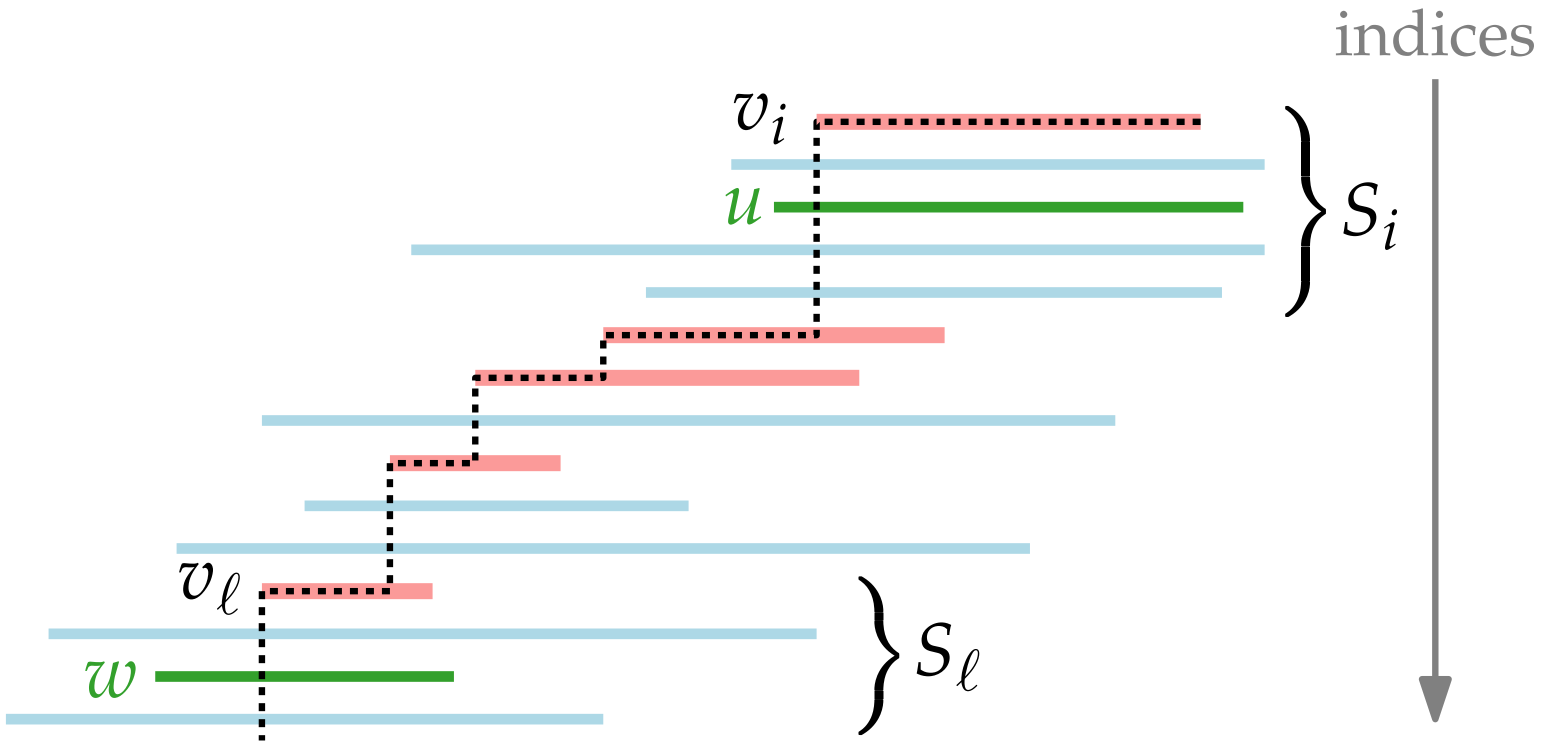
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.



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 $i < \ell$.

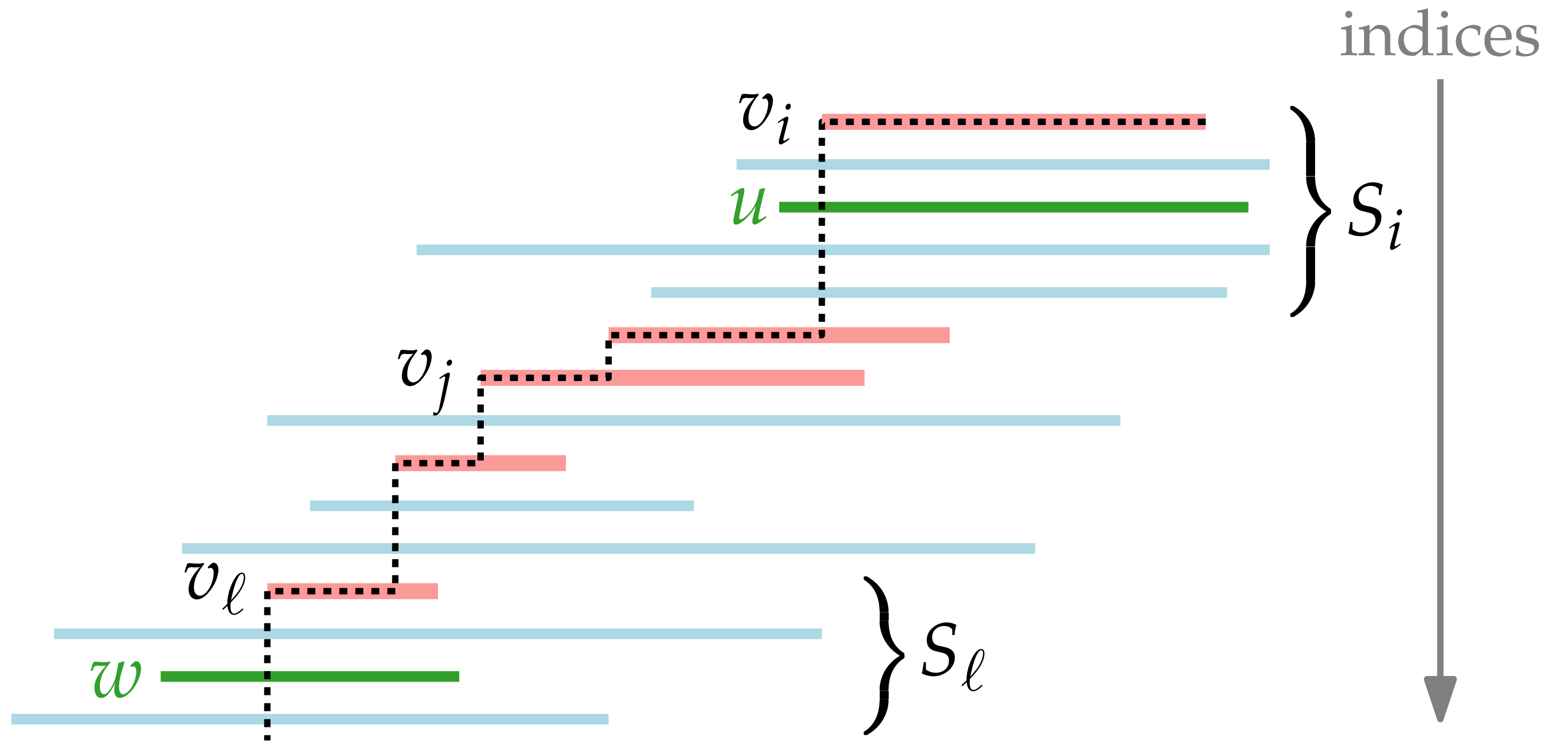


Proof of the Claim

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 $i < \ell$.

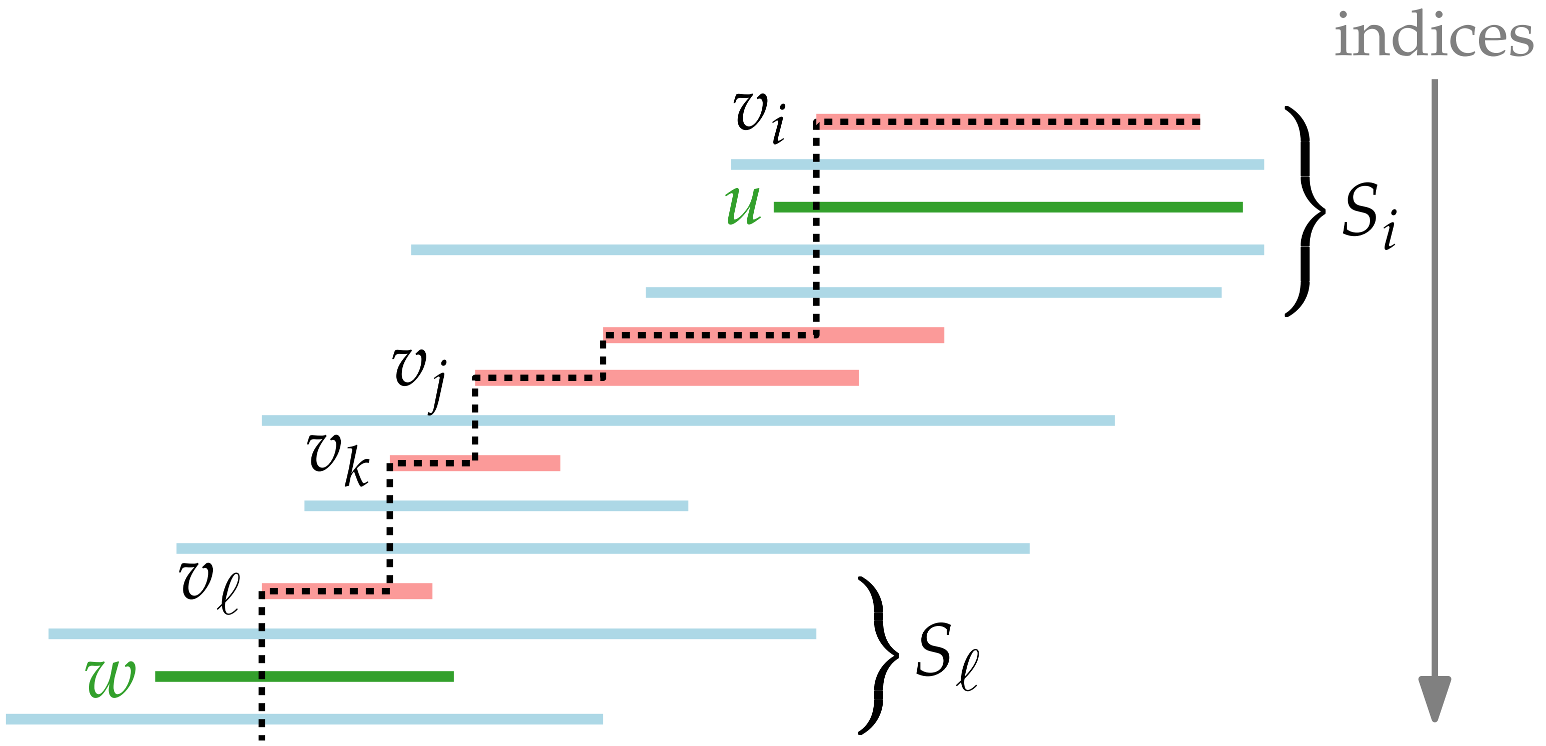
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.



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. $v_k \cap w \neq \emptyset$.



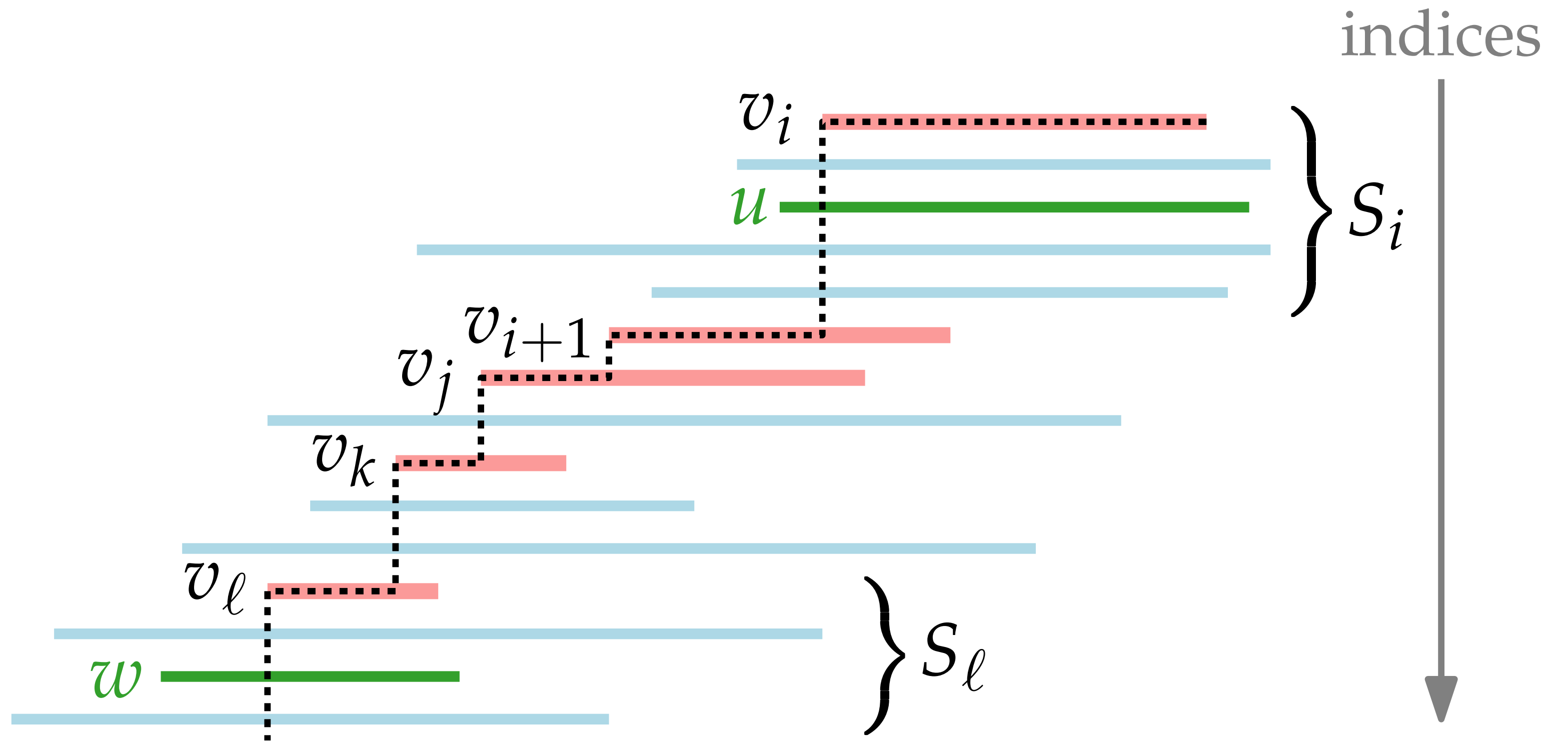
Proof of the Claim

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. $v_k \cap w \neq \emptyset$.

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



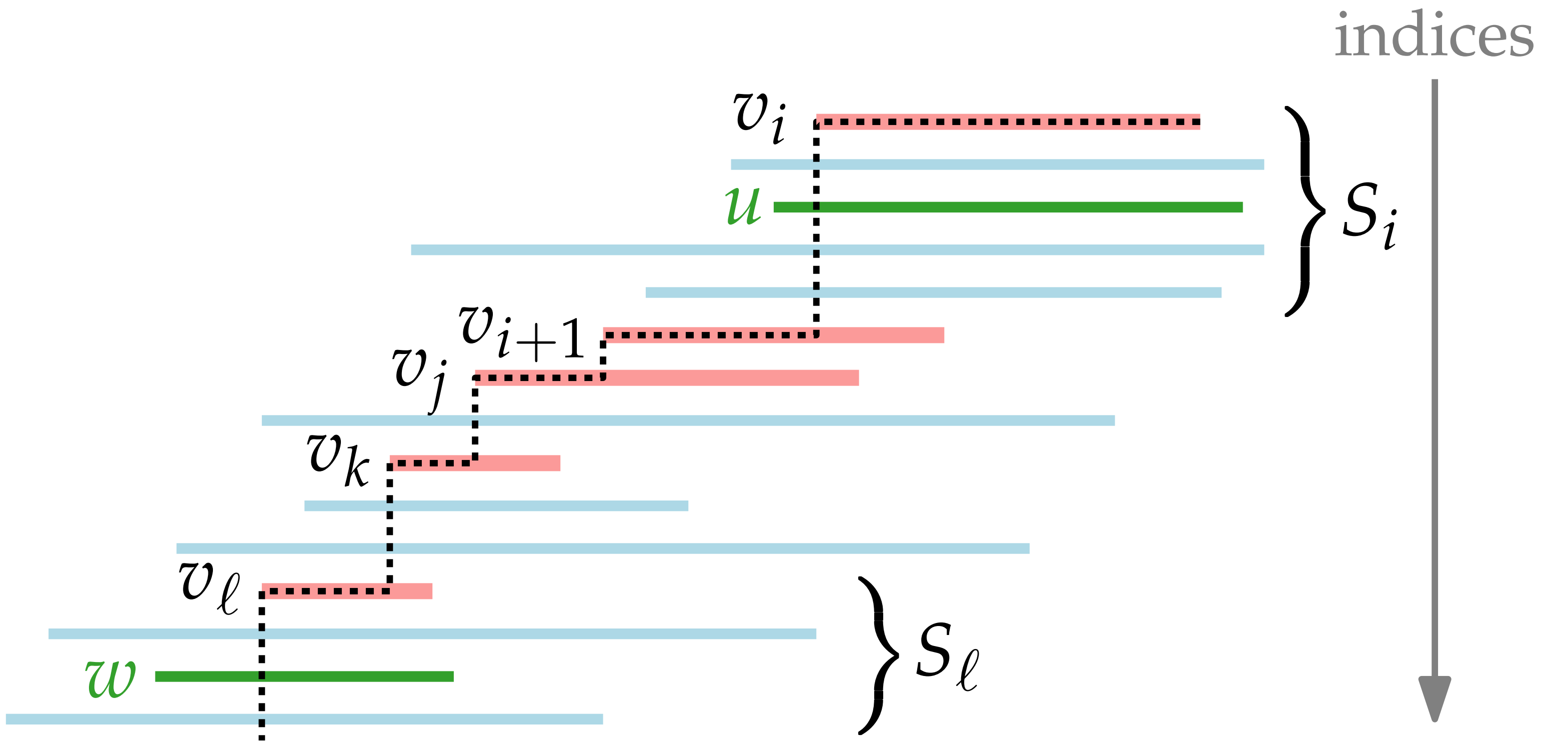
Proof of the Claim

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. $v_k \cap w \neq \emptyset$.

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



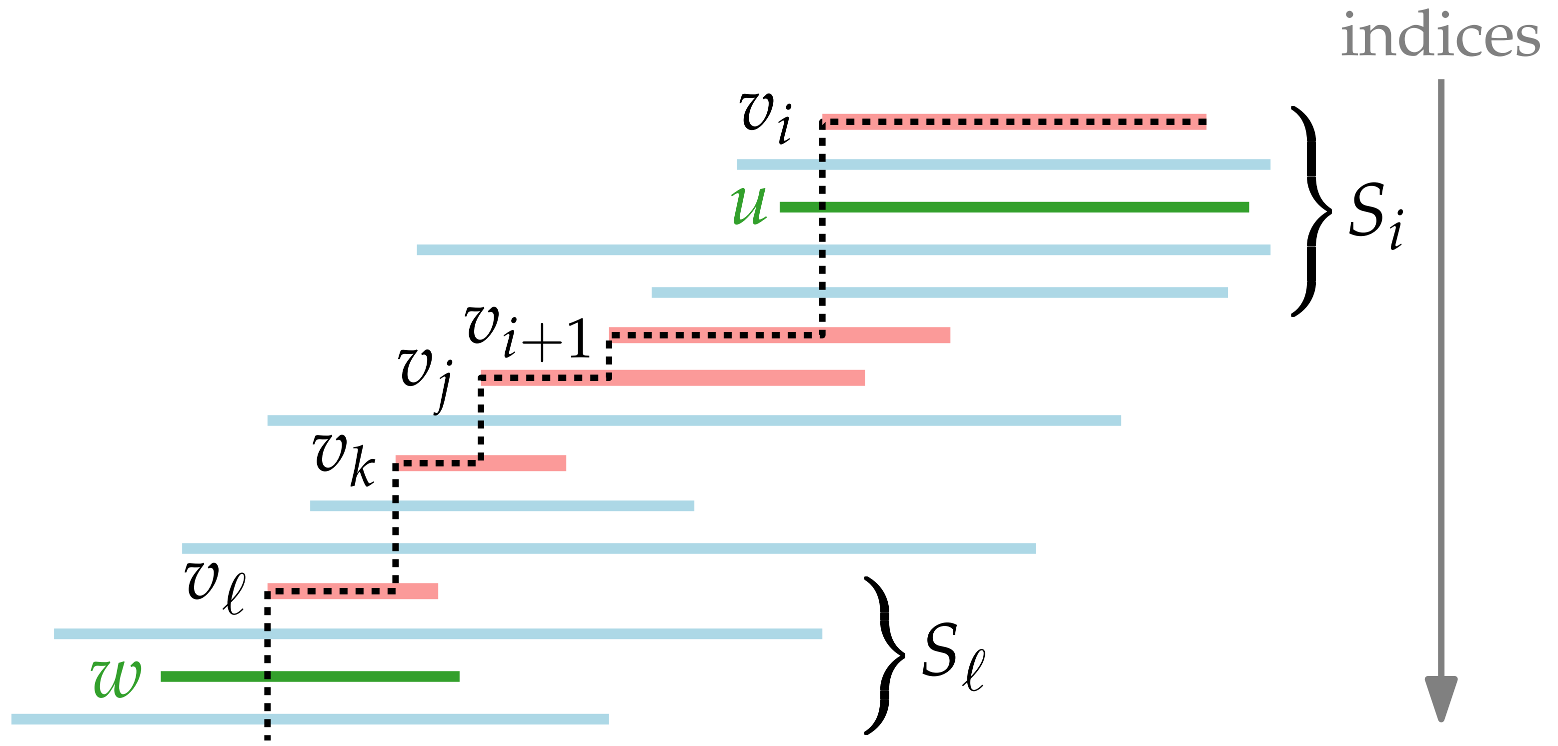
Proof of the Claim

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. $v_k \cap w \neq \emptyset$.

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



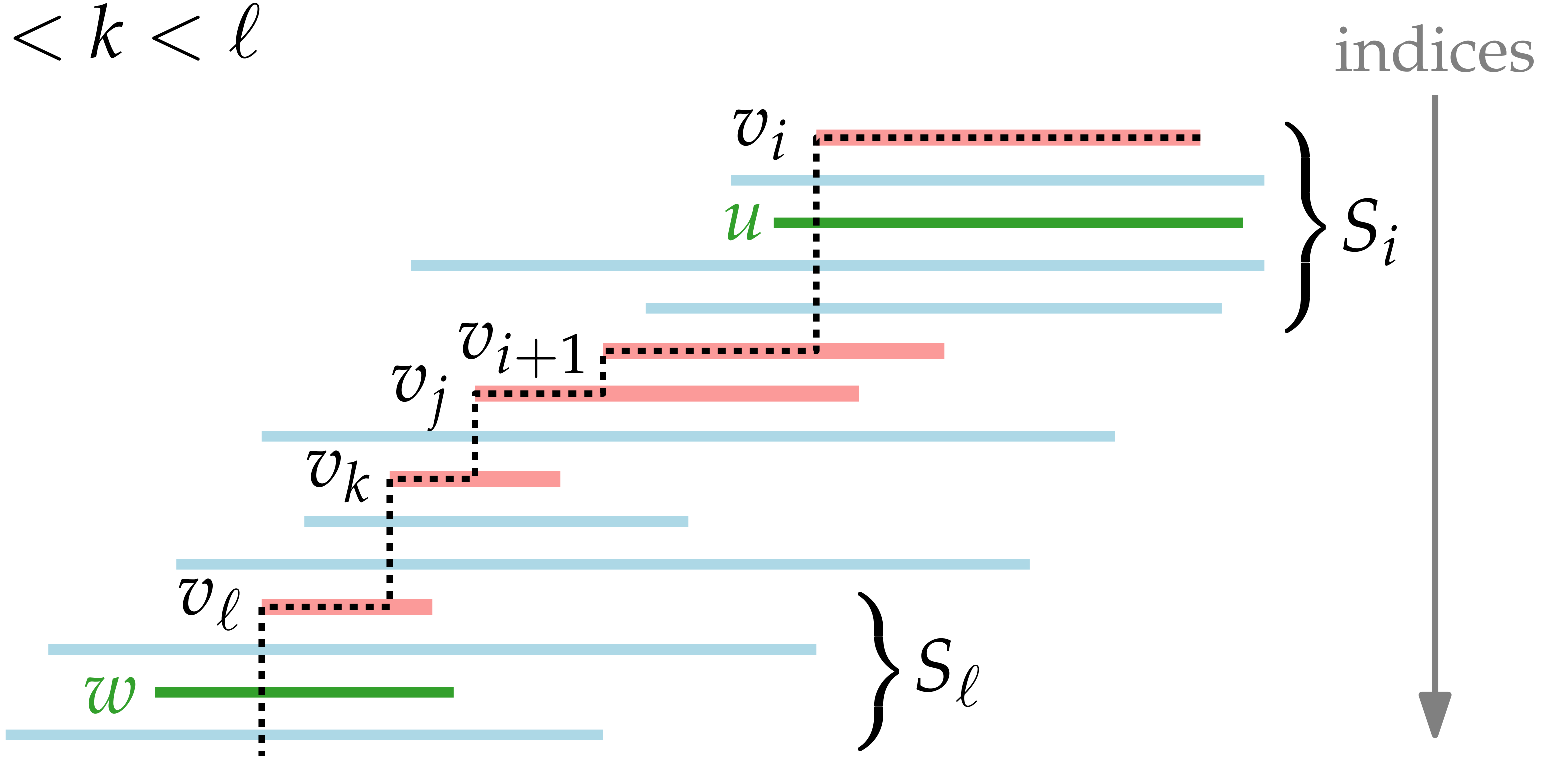
Proof of the Claim

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. $v_k \cap w \neq \emptyset$.

$$\begin{array}{c} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \underset{u \cap w = \emptyset}{\Longrightarrow} \begin{array}{c} i < j < \ell \\ i < k < \ell \end{array}$$



Proof of the Claim

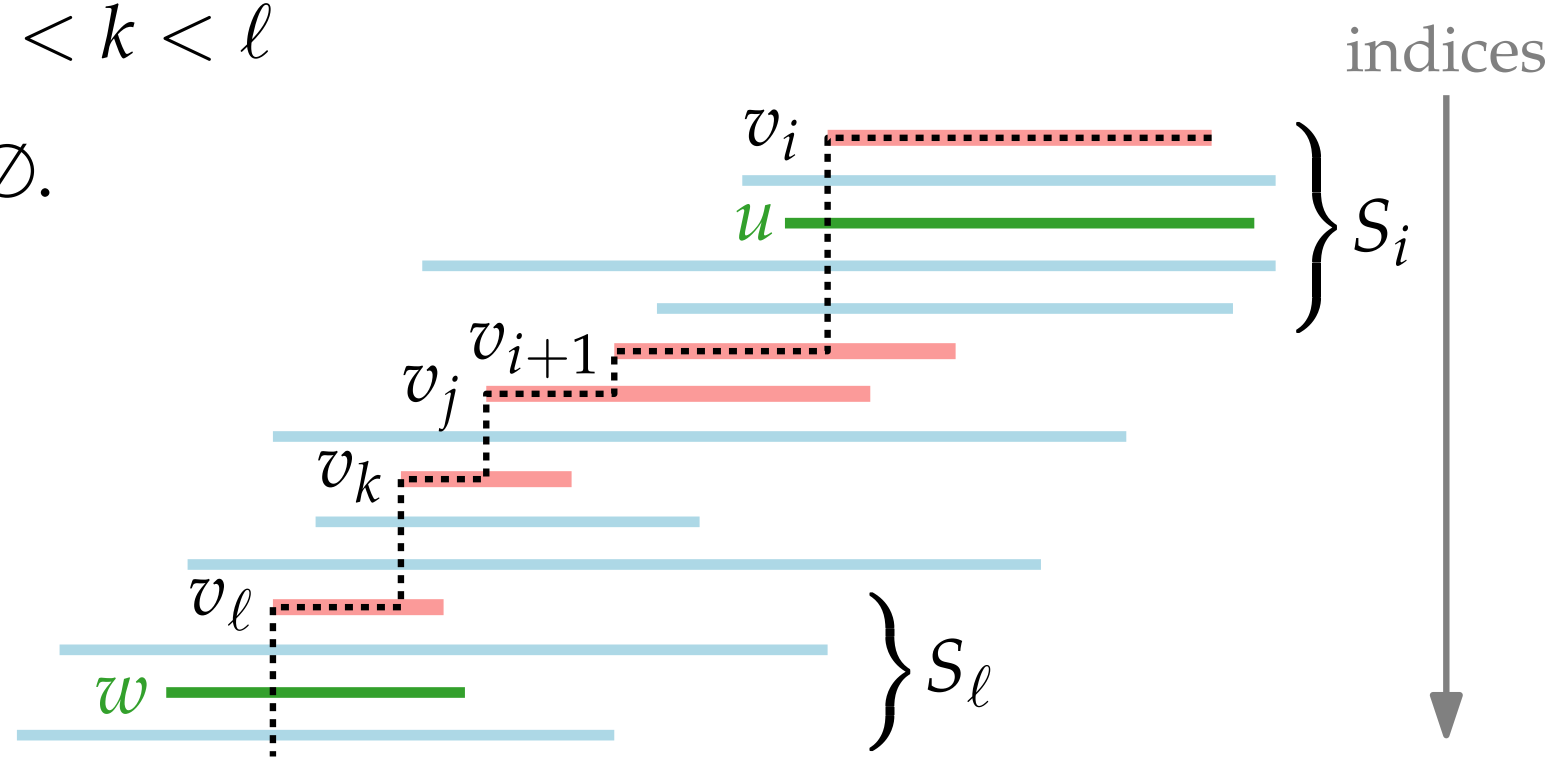
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

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. $v_k \cap w \neq \emptyset$.

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

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



Proof of the Claim

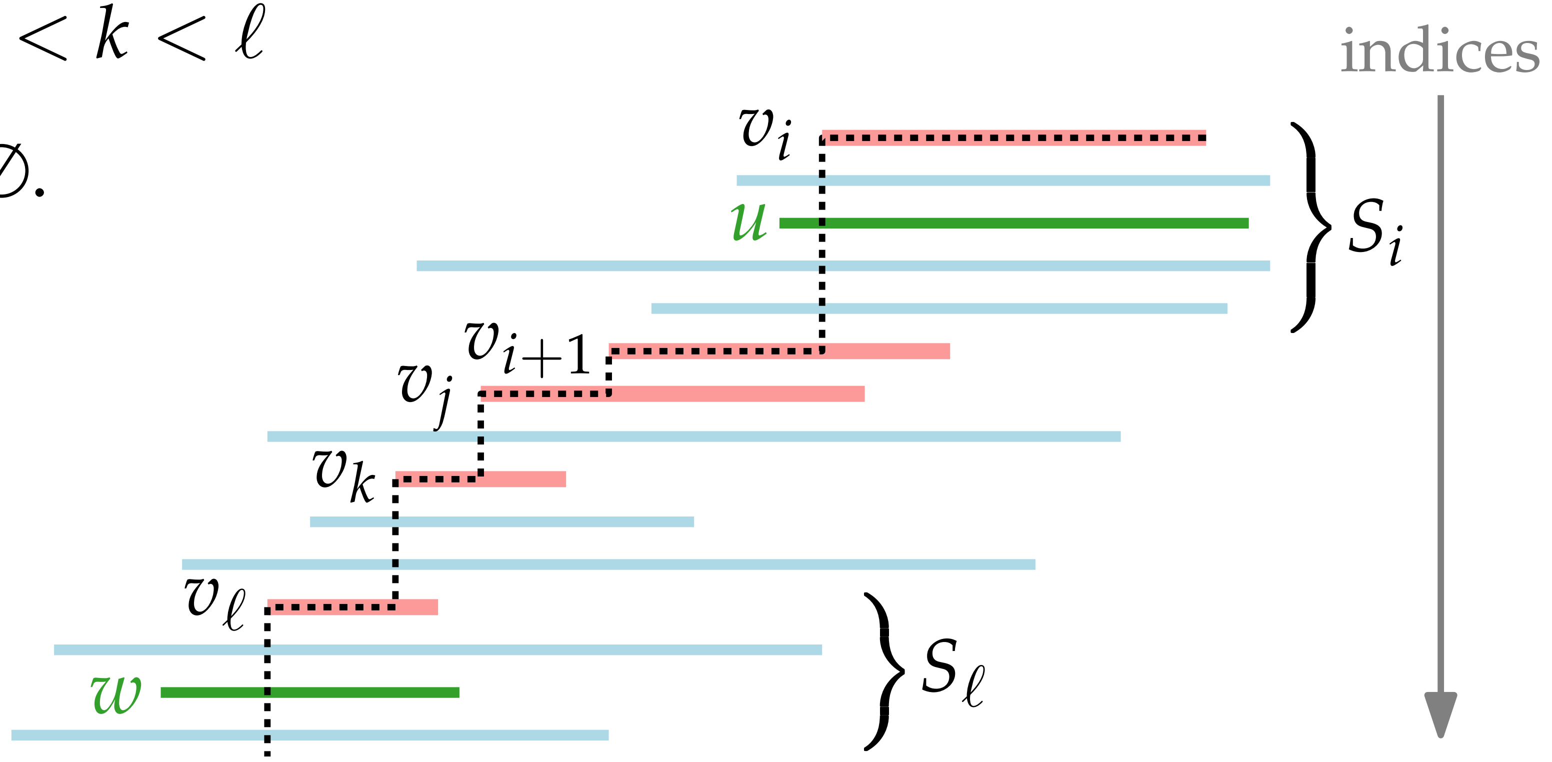
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

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. $v_k \cap w \neq \emptyset$.

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

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



Proof of the Claim

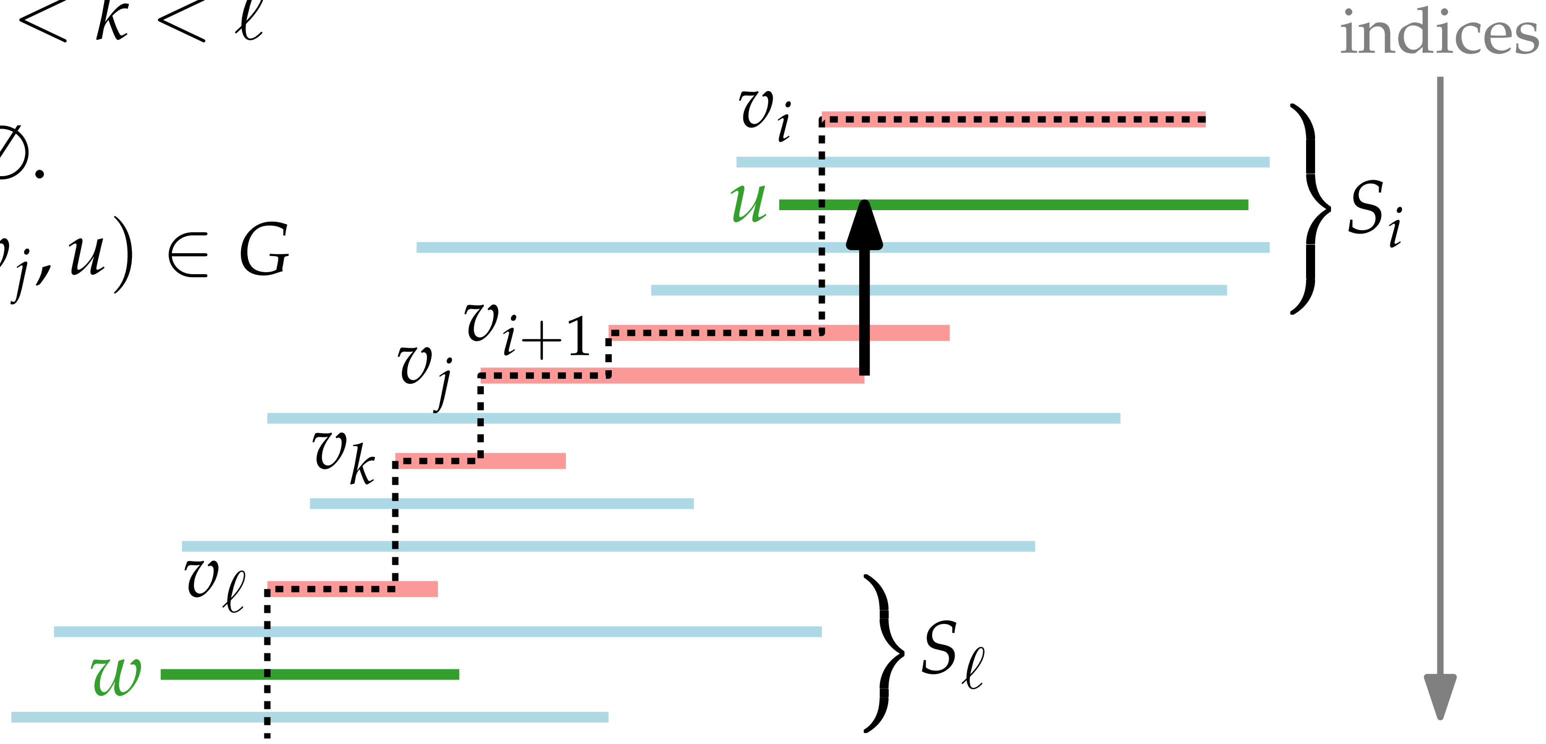
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

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. $v_k \cap w \neq \emptyset$.

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

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



Proof of the Claim

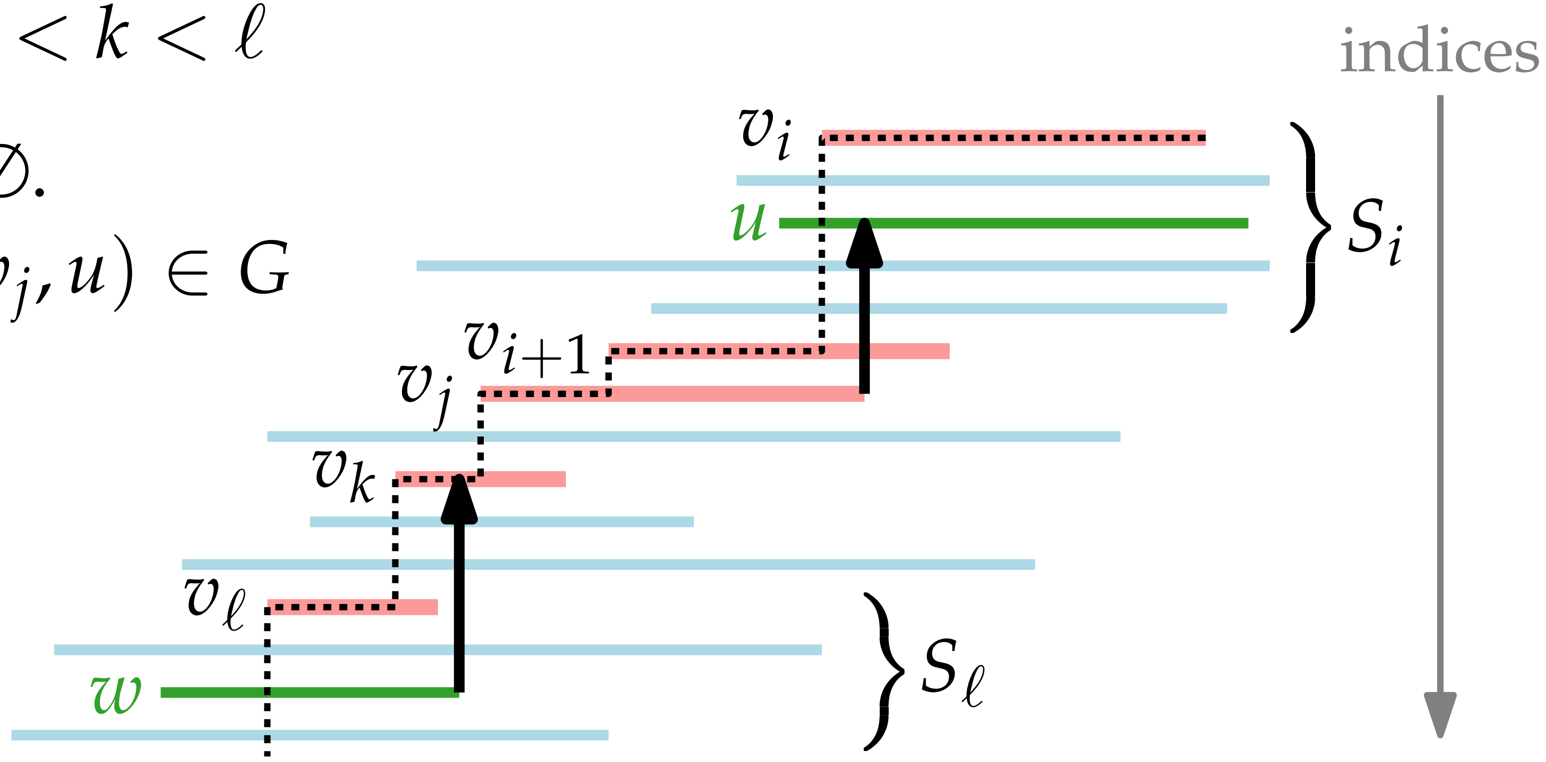
Claim: Any two intervals $u \in S_i$ and $w \in S_\ell$ are adjacent in G^+.

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. $v_k \cap w \neq \emptyset$.

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

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^+.

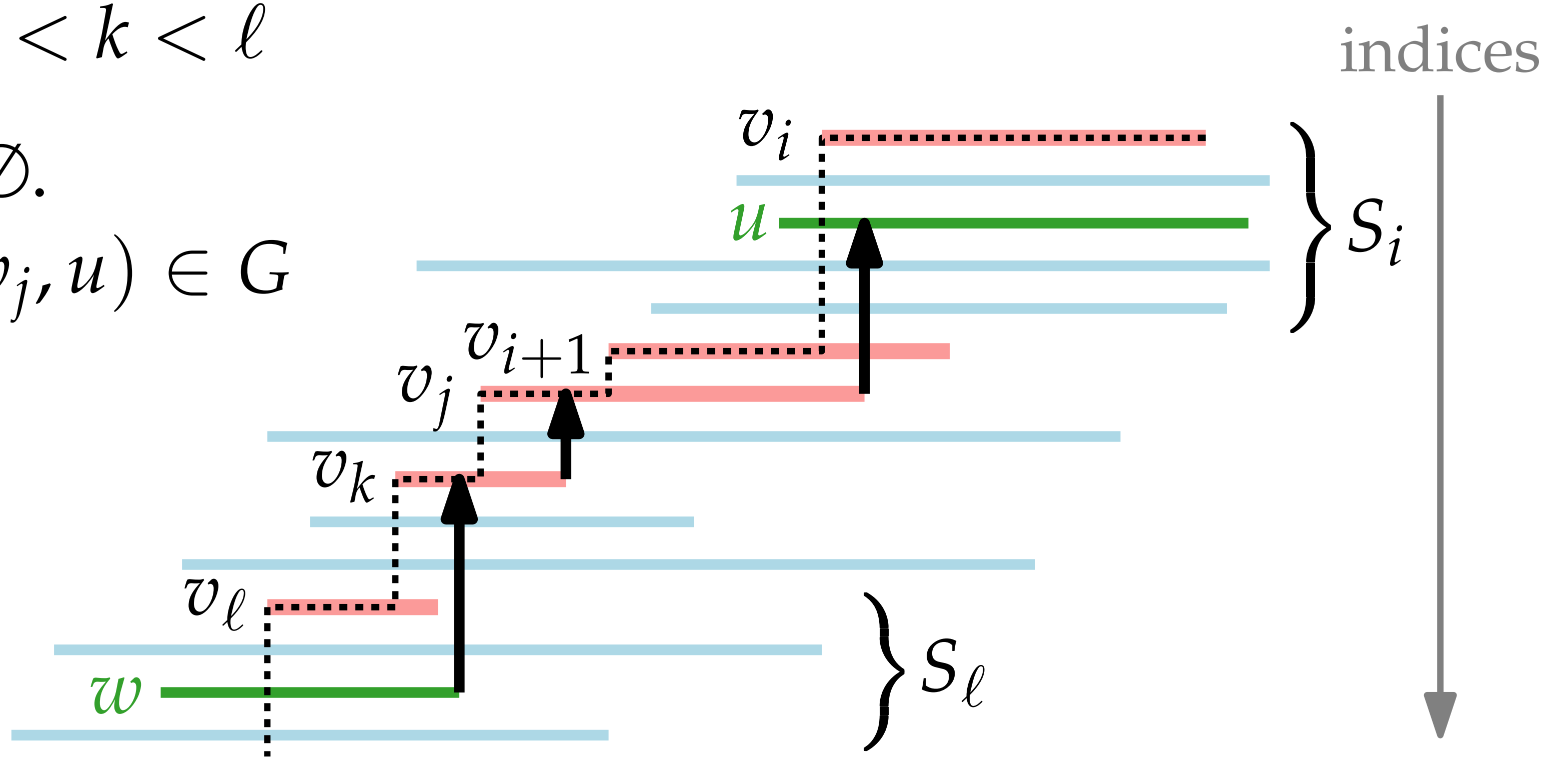
Proof. W.l.o.g., $u \cap w = \emptyset$ and $i < \ell$.

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

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

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 $i < \ell$.

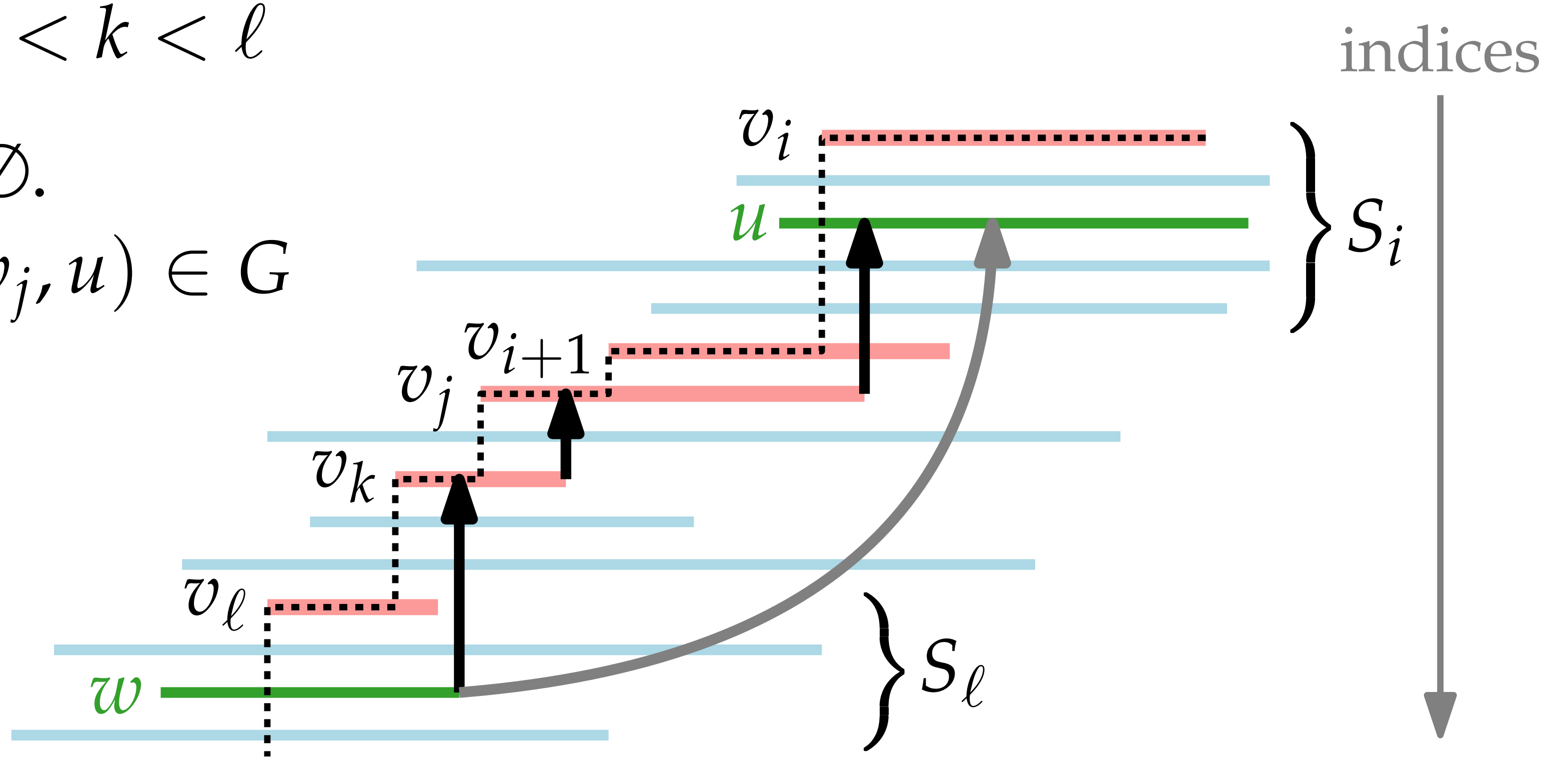
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

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

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 $i < \ell$.

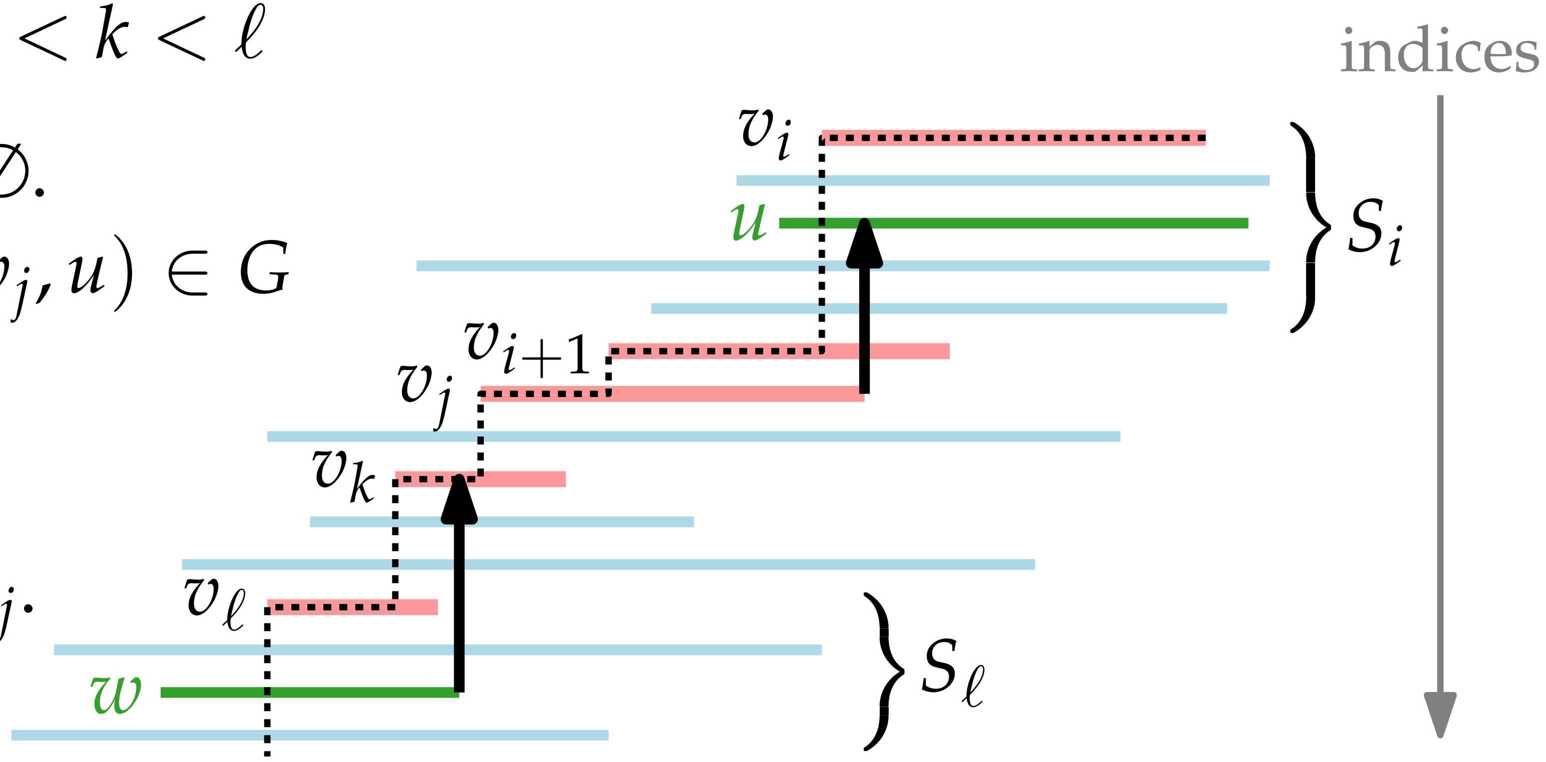
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{array}{c} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{array} \underset{u \cap w = \emptyset}{\Longrightarrow} \begin{array}{c} i < j < \ell \\ i < k < \ell \end{array}$$

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 $i < \ell$.

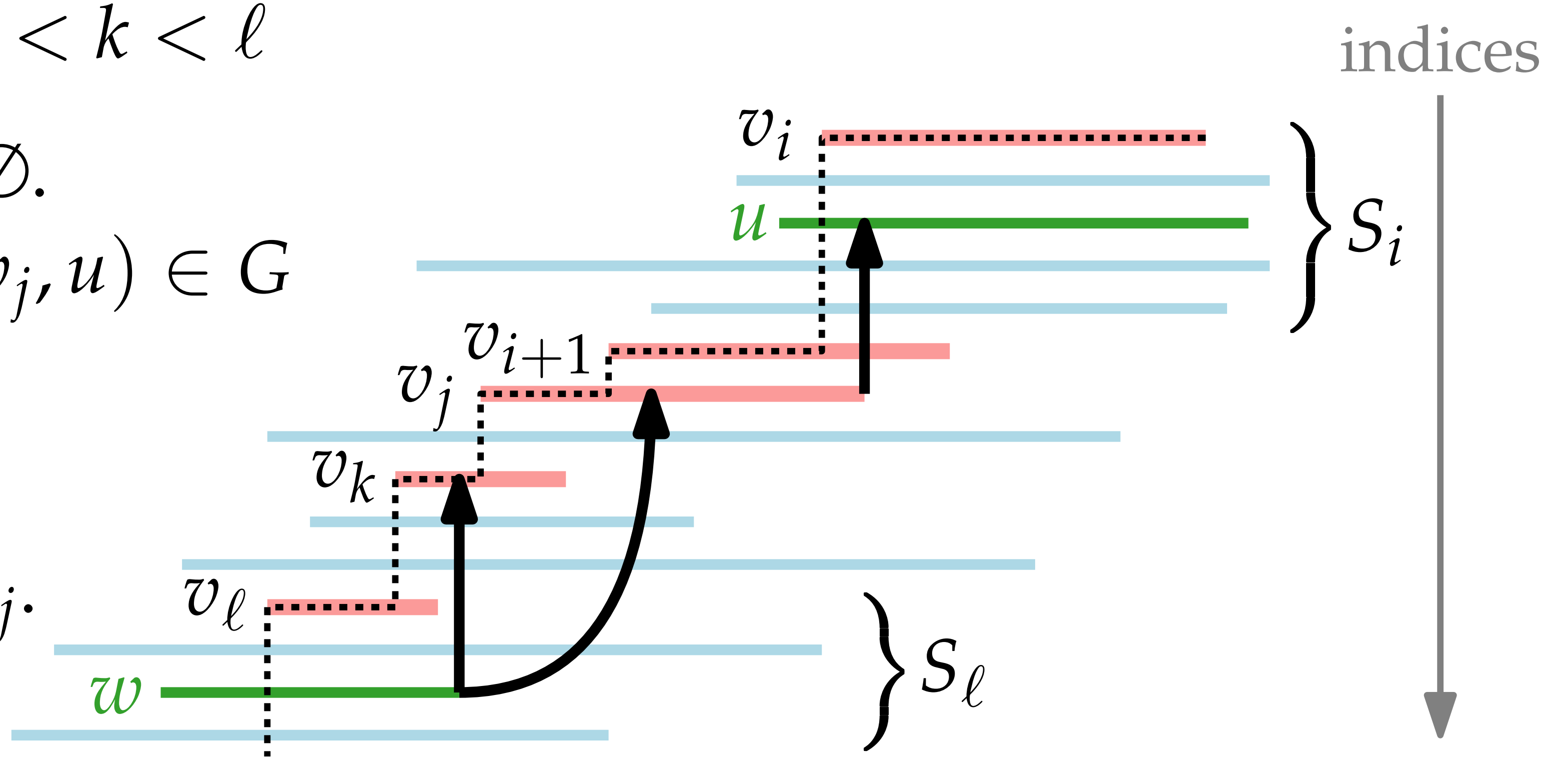
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

$$\begin{matrix} u \cap v_{i+1} \neq \emptyset \\ w \cap v_{\ell-1} \neq \emptyset \end{matrix} \quad \underset{u \cap w = \emptyset}{\Longrightarrow} \quad \begin{matrix} i < j < \ell \\ i < k < \ell \end{matrix}$$

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 $i < \ell$.

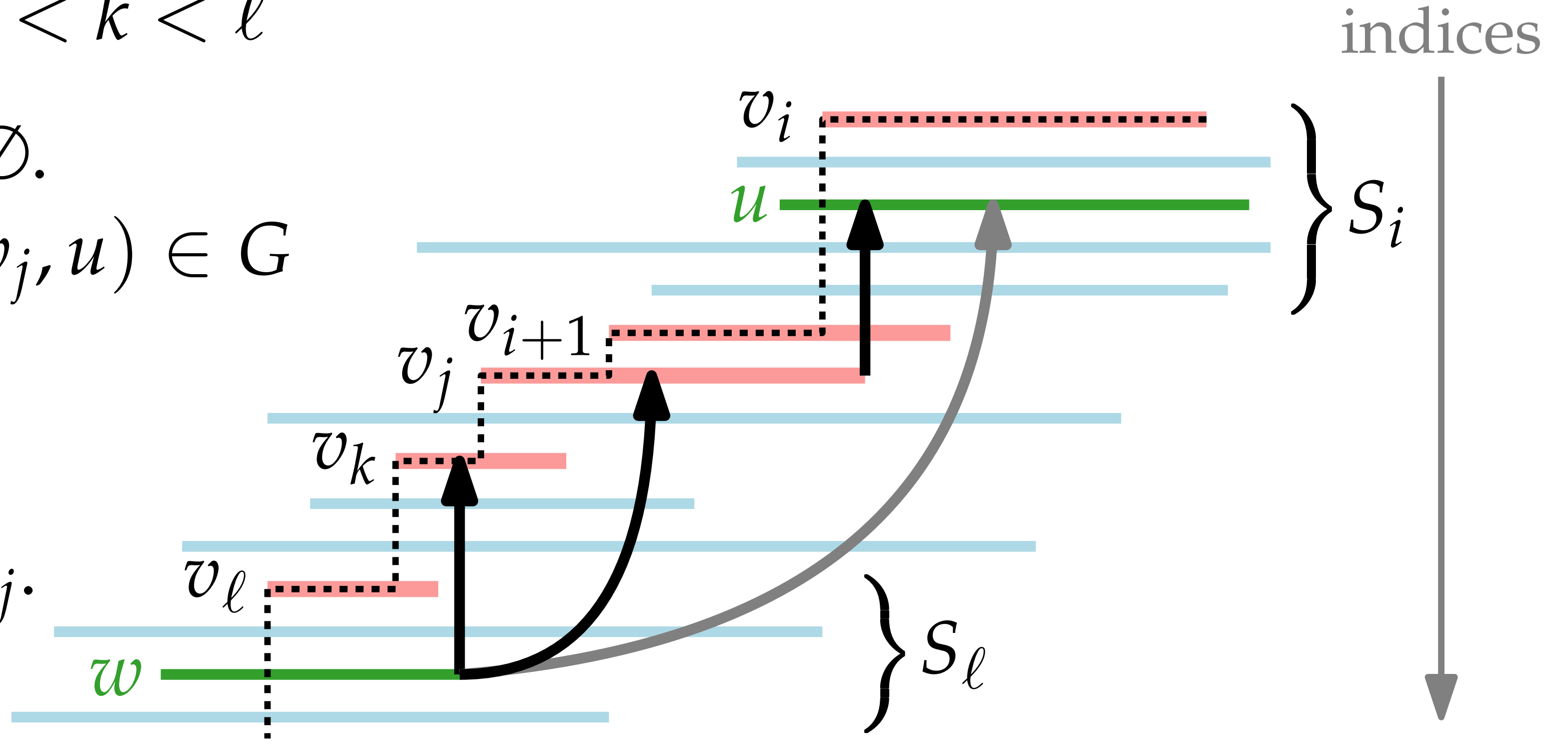
Let j be the largest index s.t. $v_j \cap u \neq \emptyset$.
Let k be the smallest index s.t. $v_k \cap w \neq \emptyset$.

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

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

our contribution

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 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:

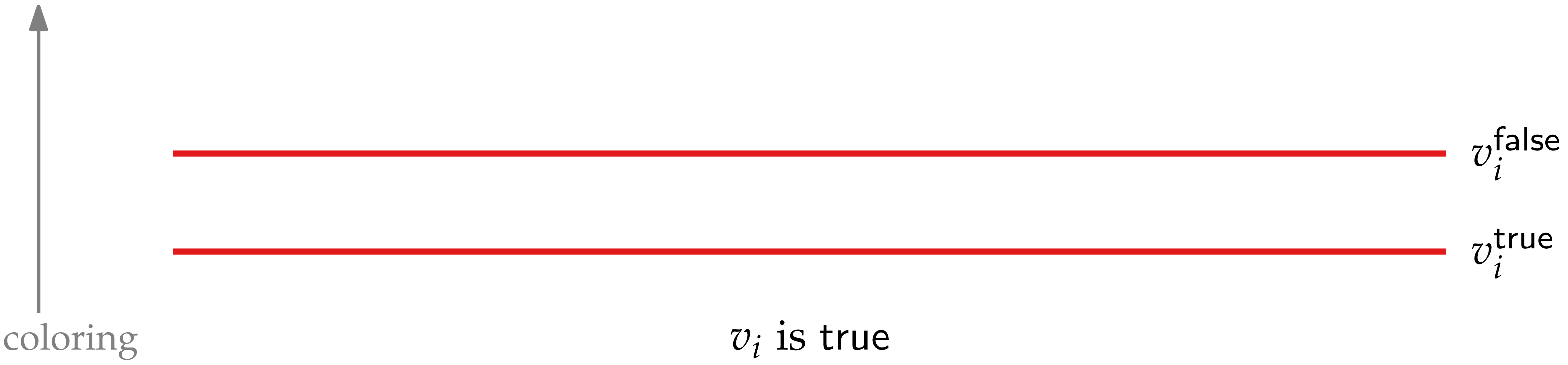
Coloring Mixed Interval Graphs

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:



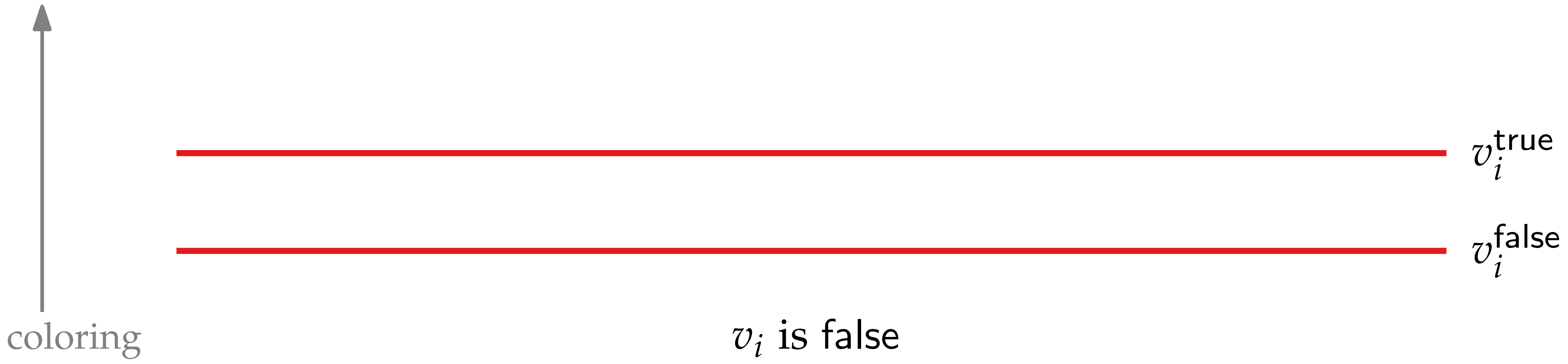
Coloring Mixed Interval Graphs

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:



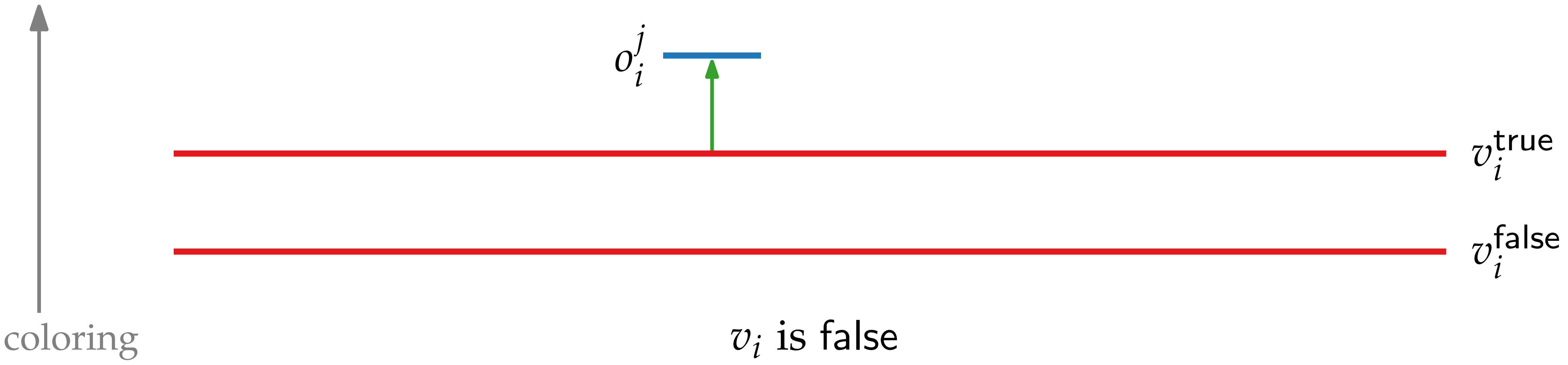
Coloring Mixed Interval Graphs

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:



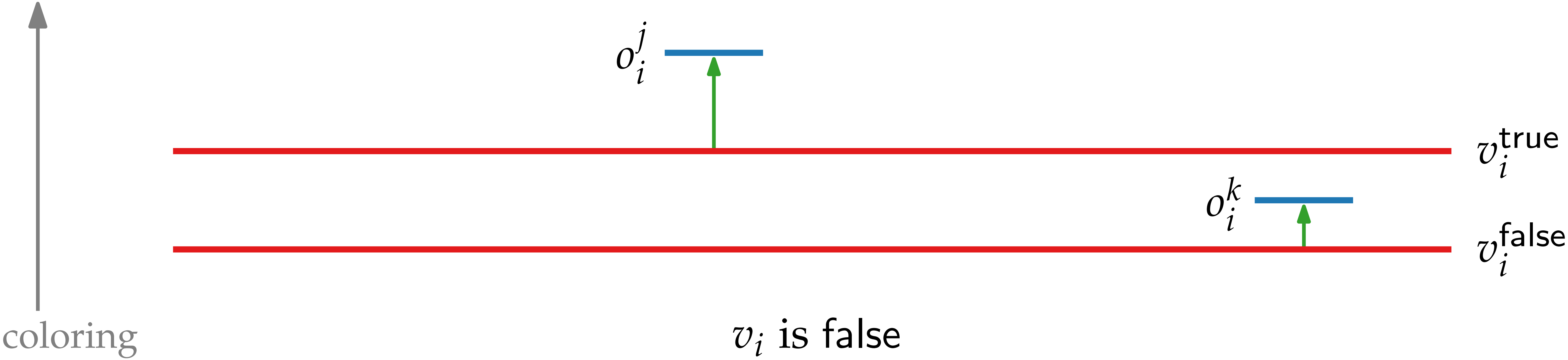
Coloring Mixed Interval Graphs

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 $\neg v_i$:



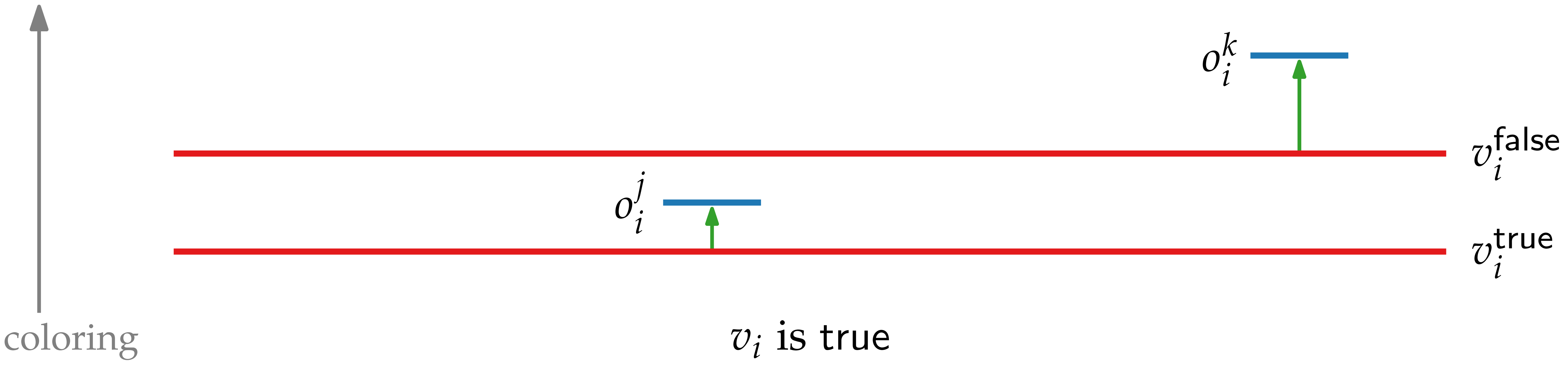
Coloring Mixed Interval Graphs

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 $\neg v_i$:



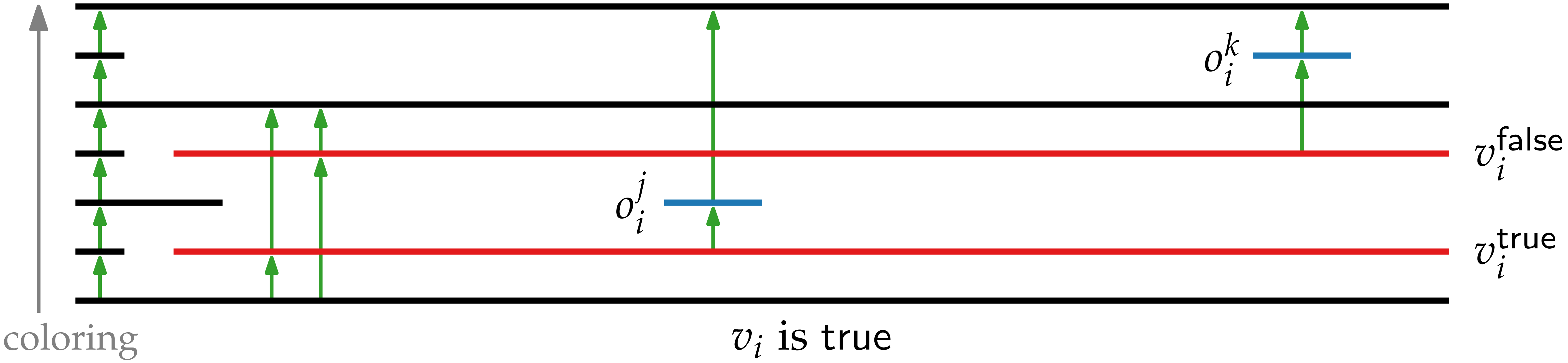
Coloring Mixed Interval Graphs

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



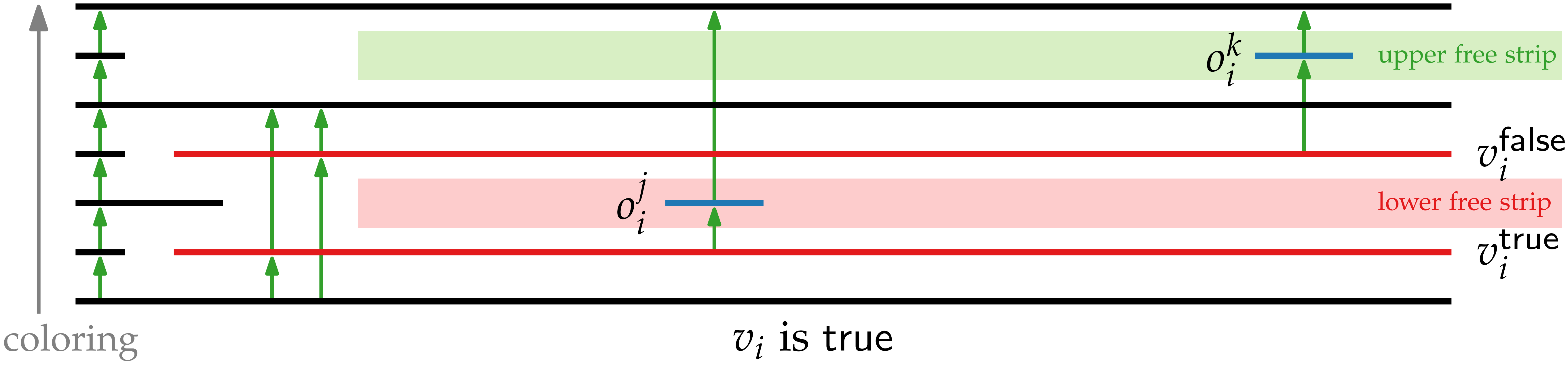
Coloring Mixed Interval Graphs

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



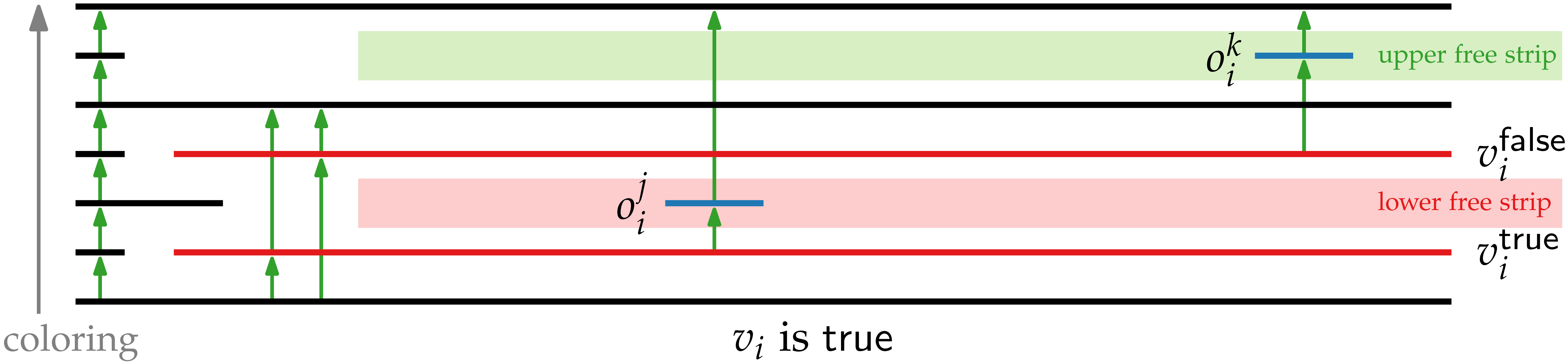
v_i is true

Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

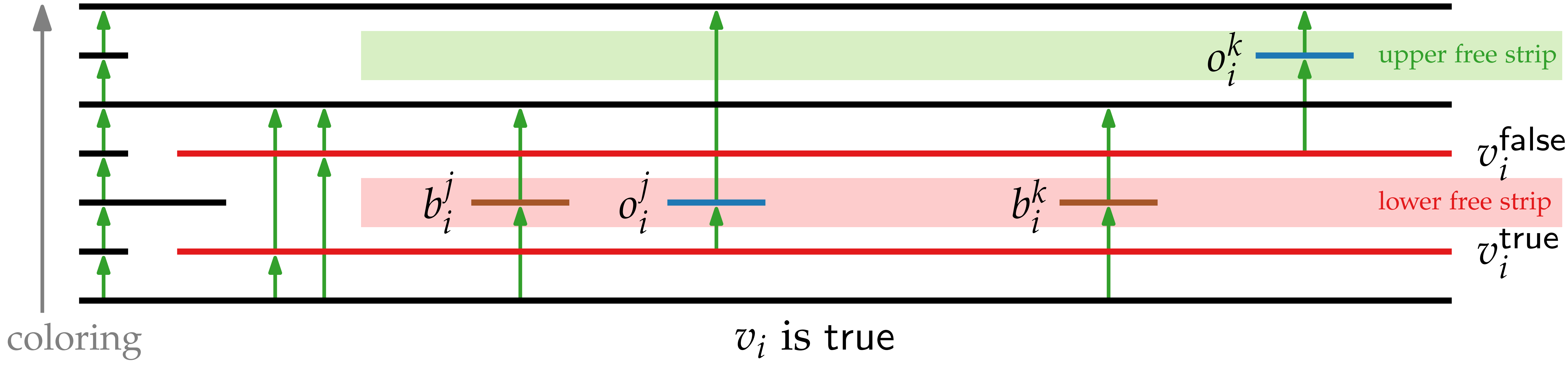


Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

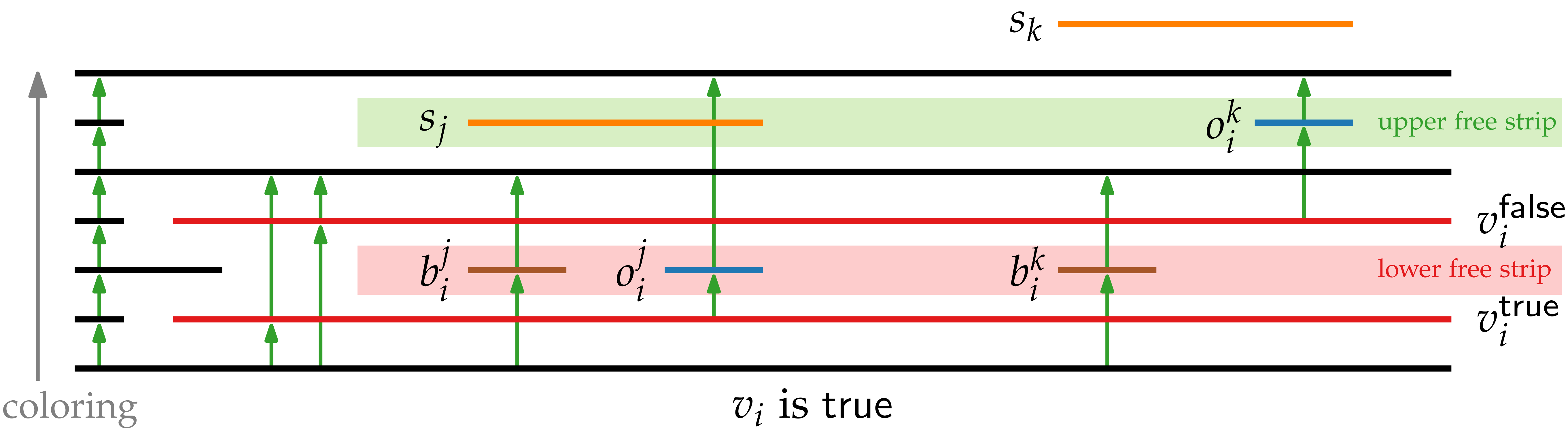


Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

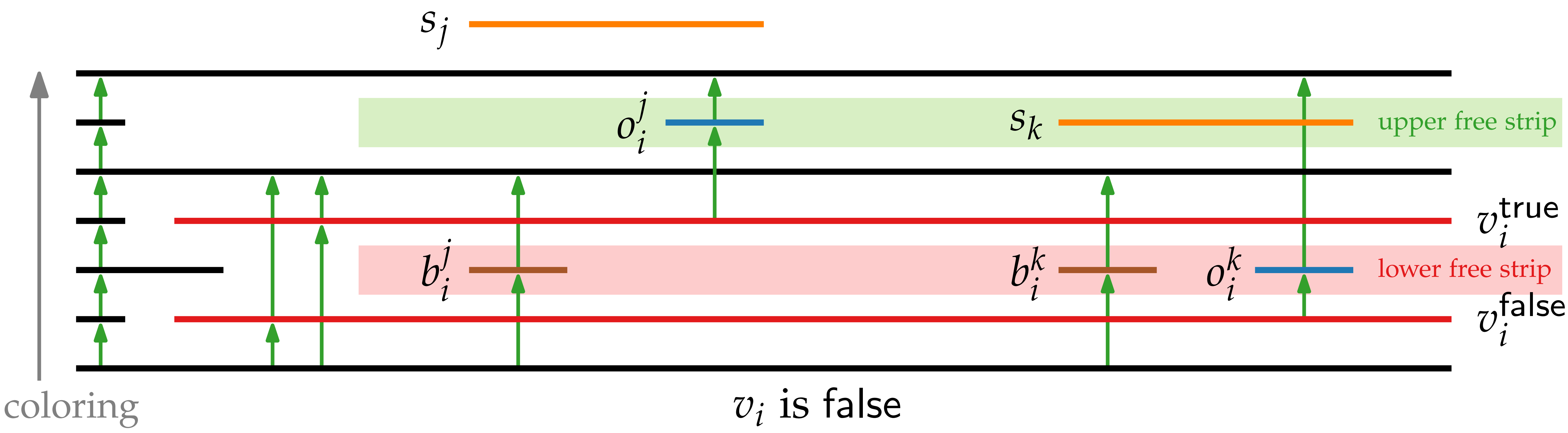


Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

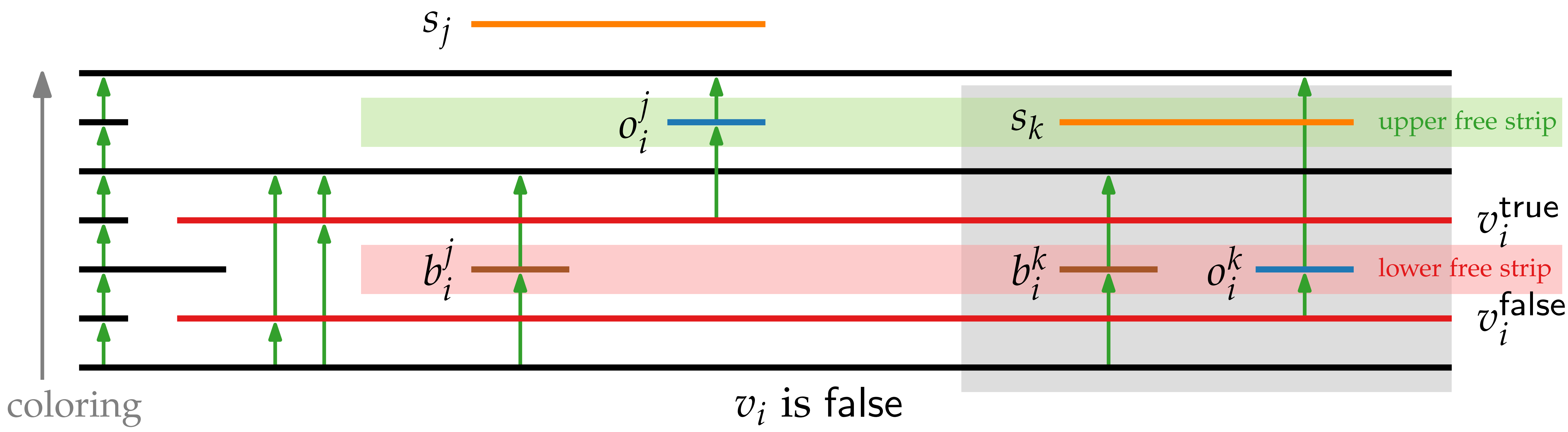


Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

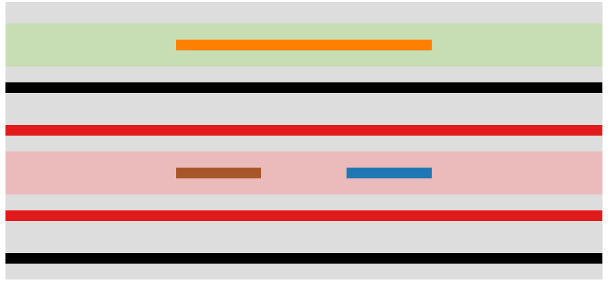


Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:



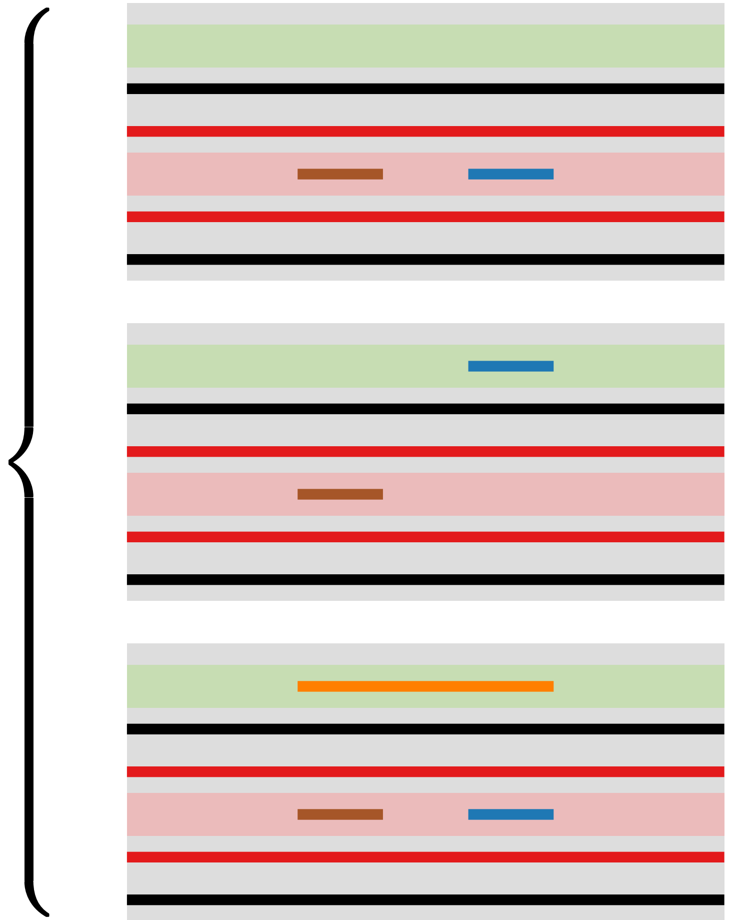
Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

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



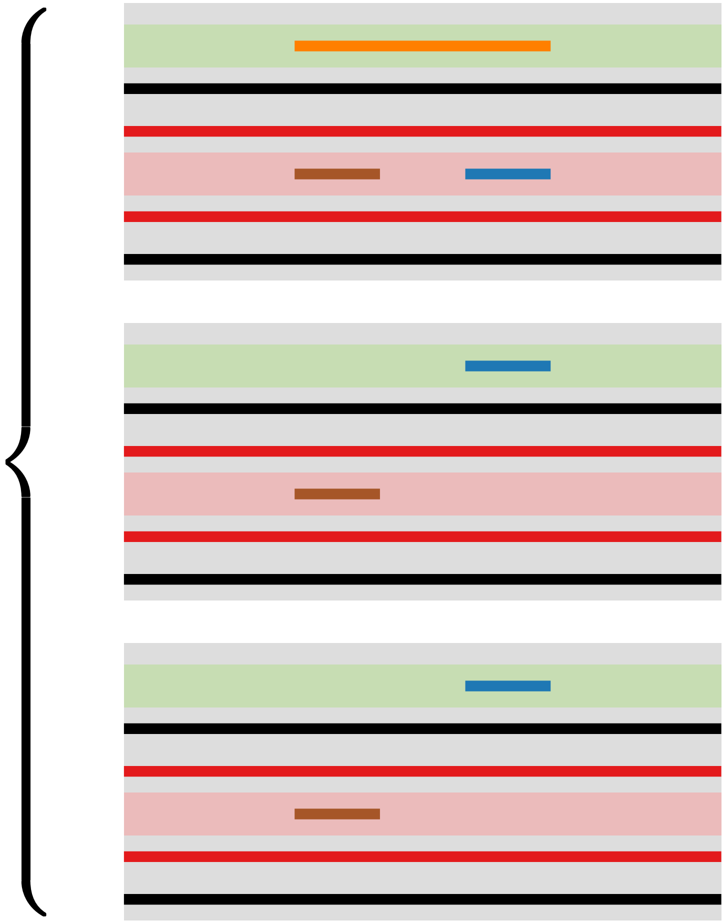
Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

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



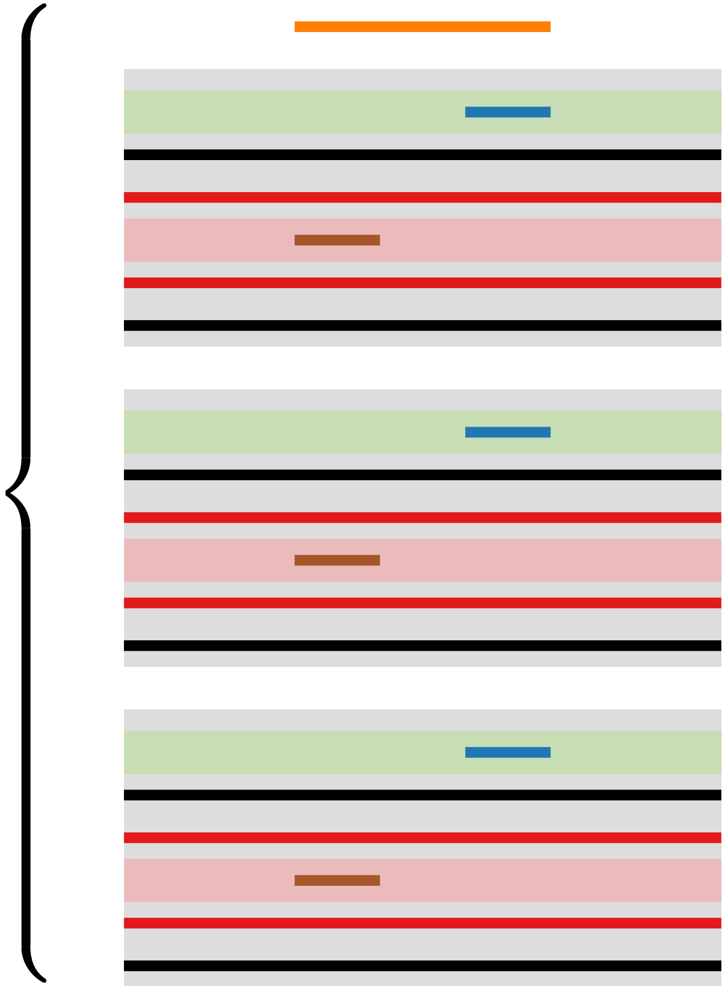
Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

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



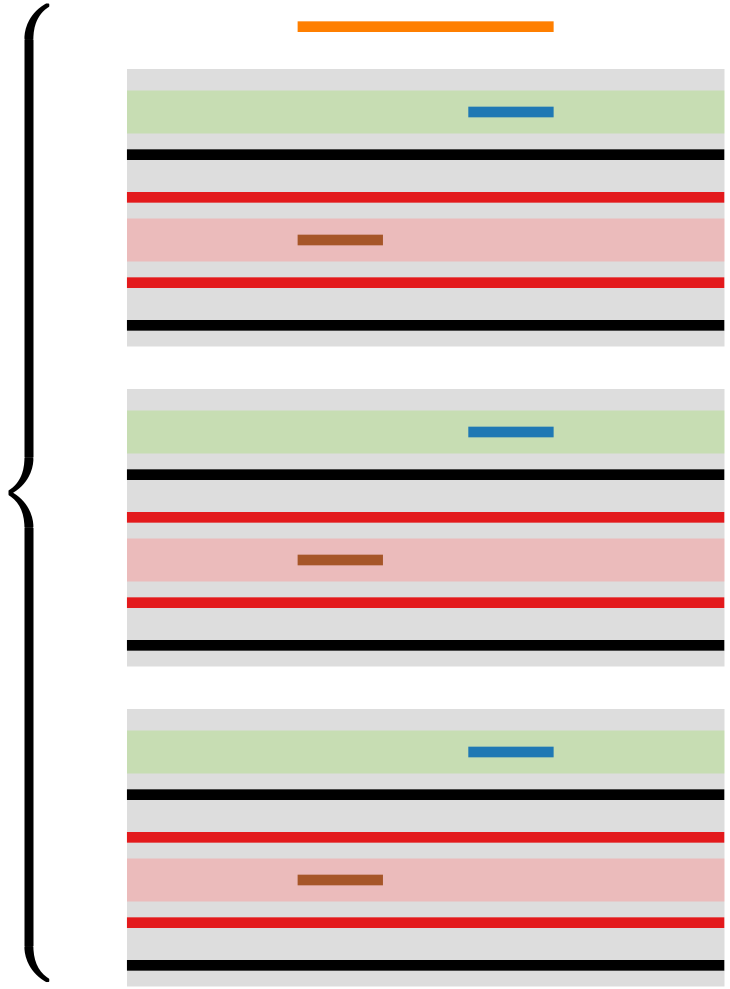
Coloring Mixed Interval Graphs

Theorem 2:
Deciding whether a mixed interval graph admits a k-coloring is NP-complete.

Proof sketch:

clause gadget:

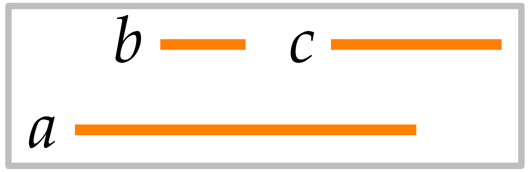
$6n + 1$ colors
($n :=$ # variables)

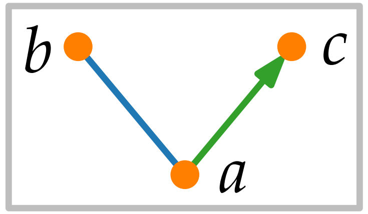


Φ is satisfiable $\Leftrightarrow$ G_Φ admits a coloring with $6n$ colors

$\square$

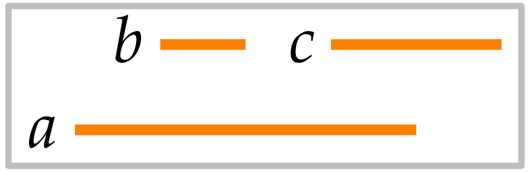
Conclusion and Open Problems

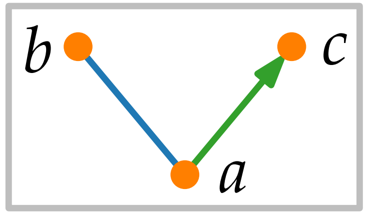




- We have introduced the natural concept of directional interval graphs.

Conclusion and Open Problems

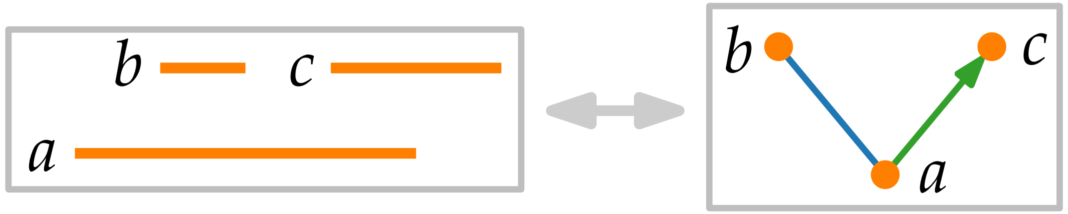




- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

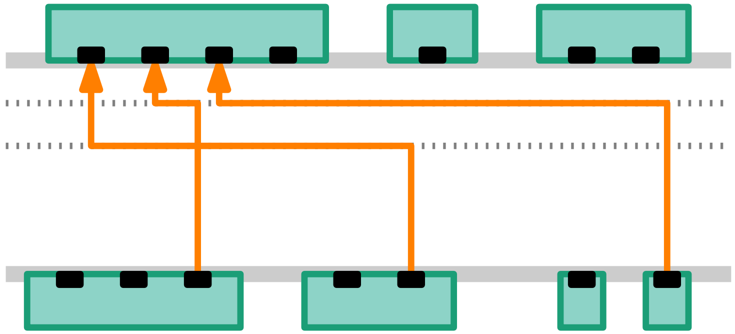
$n :=$ # vertices

Conclusion and Open Problems

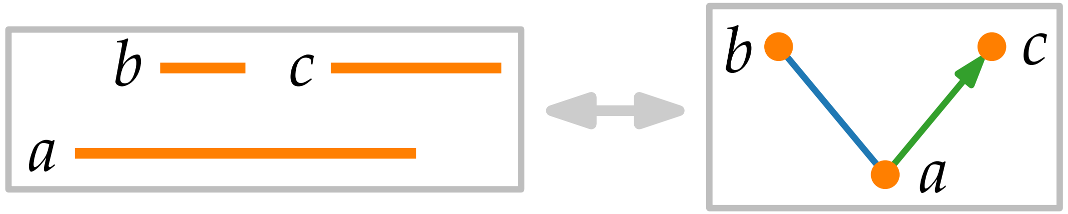


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

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

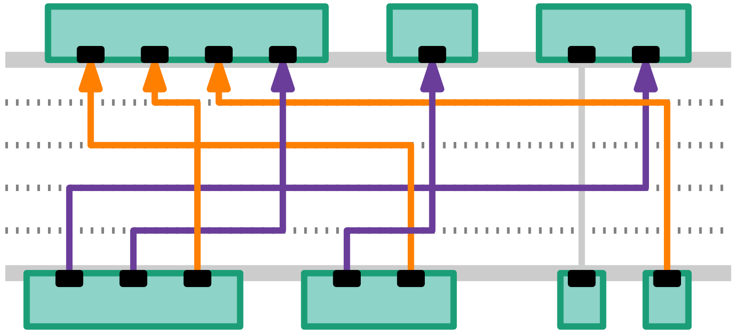


Conclusion and Open Problems

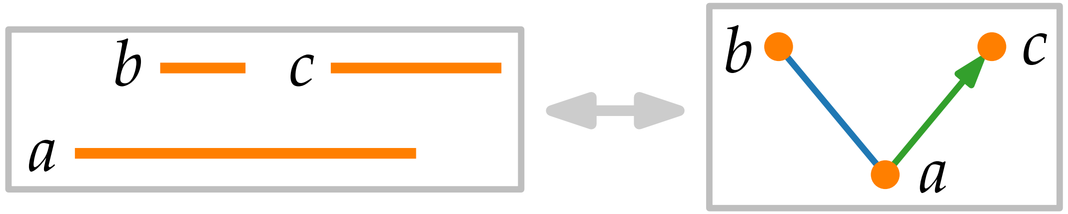


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.
  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically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



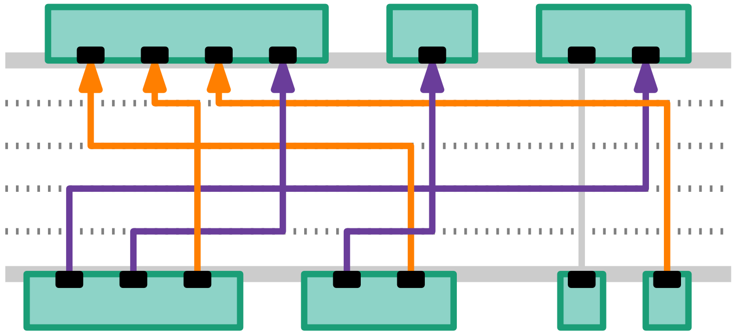
Conclusion and Open Problems



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

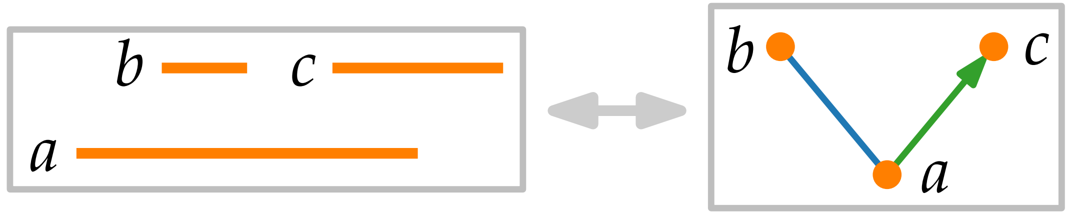
  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically "right-going".)

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



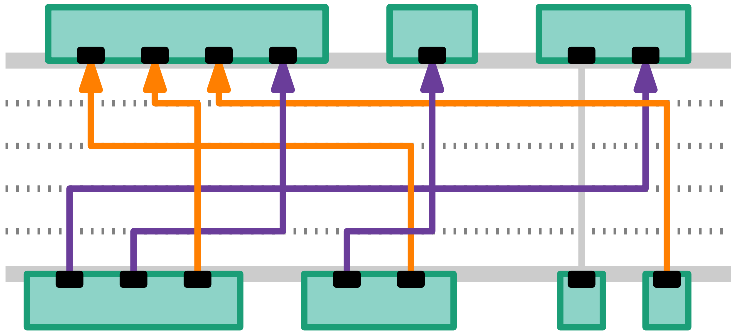
- In our paper, we present a constructive $O(n^2)$-time algorithm for recognizing directional interval graphs, which is based on PQ-trees.

Conclusion and Open Problems



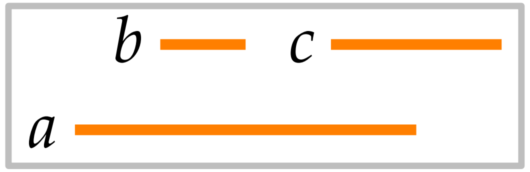
- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time. $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically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

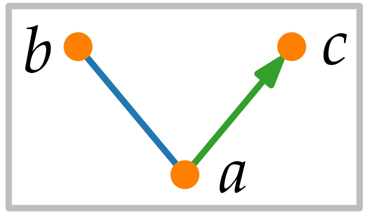
⇒ Combining the drawings of left-going and right-going edges yields a 2-approximation for the number of tracks. (bidirectional interval graphs)



- 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.)

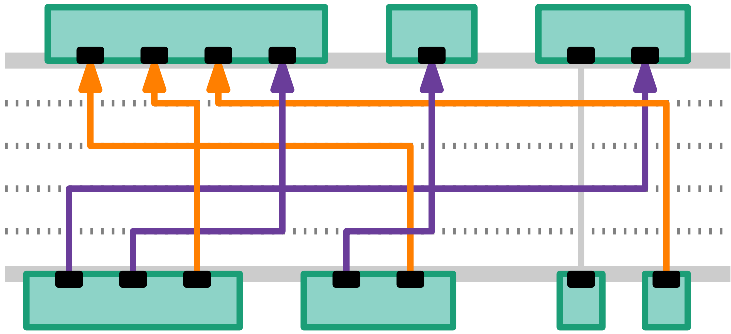
Conclusion and Open Problems





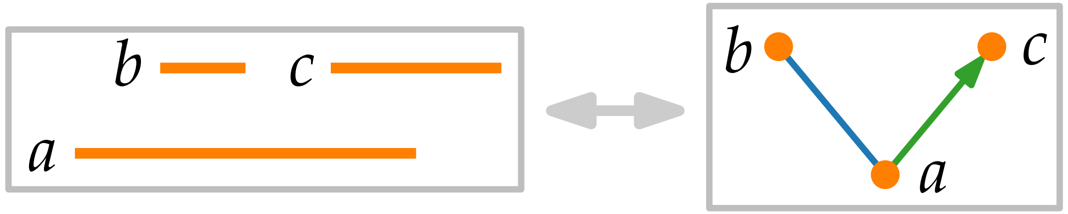
- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.
  $n :=$ # vertices
- In layered graph drawing, this corresponds to routing "left-going" edges orthogonally to the fewest horizontal tracks. (Symmetrically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

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



- 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.)

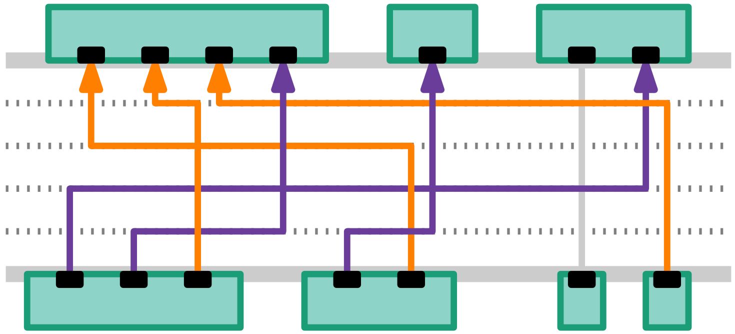
Conclusion and Open Problems



- 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)

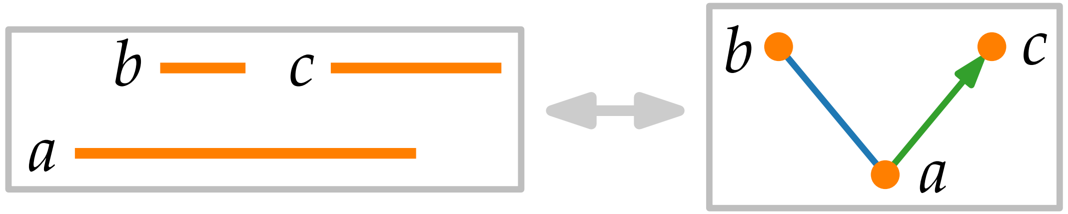


can we do better?

- 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

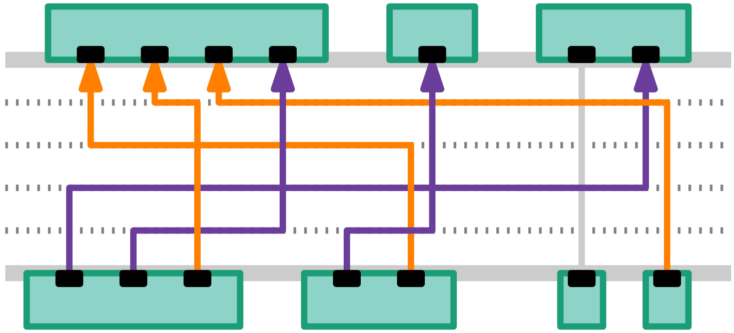


- We have introduced the natural concept of directional interval graphs.
- A simple greedy algorithm colors these graphs optimally in $O(n \log n)$ time.

$n :=$ # vertices

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

⇒ 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.)