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

Motivation


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


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

Consists of five phases:
Motivation


**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


**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


**Input:** directed graph G

Consists of five phases:

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

we want orthogonal edges!

[Walter, Z., Baumeister, Wolff; GD’20, CGTA’22]
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

- 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

\[ \text{Diagram: } a \quad \overline{b \quad c} \]
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 $c : V \rightarrow \mathbb{N}$ 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 $c : V \rightarrow \mathbb{N}$ such that:

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

Interval graphs (no directed edges):

- 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 $c: V \rightarrow \mathbb{N}$ such that:

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

Interval graphs (no directed edges):
- coloring in linear time by a greedy algorithm

★ undirected edge $uv$: $c(u) \neq c(v)$,
★ directed edge $uv$: $c(u) < c(v)$,
★ $\max_{v \in V} c(v)$ is minimized.
Find a graph coloring \( c: V \to \mathbb{N} \) such that:

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

Interval graphs (no directed edges):
- coloring in linear time by a greedy algorithm

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

Directed graphs (only directed edges):
Coloring Mixed Graphs

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

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

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 $c : V \rightarrow \mathbb{N}$ such that:

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

Interval graphs (no directed edges):

- coloring in linear time by a greedy algorithm

Directional interval graphs:

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

Directed graphs (only directed edges):

- coloring in linear time using topological sorting
Coloring Mixed Graphs

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

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

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

$n := \# \text{ intervals}$
Coloring Mixed Graphs

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

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

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

$\star$ 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.

$n := \# \text{ intervals}$
Coloring Mixed Graphs

Find a graph coloring \( c : V \rightarrow \mathbb{N} \) 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

\( n := \# \text{ intervals} \)
Coloring Mixed Graphs

Find a graph coloring \( c : V \rightarrow \mathbb{N} \) such that:

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

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

\( \star \) 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.

\( n := \# \text{ intervals} \)
Find a graph coloring \( c: V \rightarrow \mathbb{N} \) such that:

\[ \text{Sotskov, Tanaev '76; Hansen, Kuplinsky, de Werra '97} \]

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

\( n := \# \text{ intervals} \)
Coloring Mixed Graphs

Find a graph coloring \( c : V \rightarrow \mathbb{N} \) such that:

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

- Interval graphs (no directed edges):
  - coloring in linear time by a greedy algorithm

Directions 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

\( n := \# \text{ intervals} \)
Coloring Mixed Graphs

Find a graph coloring \( c : V \rightarrow \mathbb{N} \) 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

\( n := \# \text{ intervals} \)
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
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
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 $c$.
  \[ \Rightarrow \text{ the coloring } c \text{ uses the minimum number of colors} \]
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., $c(v_0) = m$. 
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., $c(v_0) = m$. 

![Diagram](image)
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.
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.
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.
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 with intervals containing the left endpoint of $v_i$.
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 with intervals containing the left endpoint of $v_i$. 

---

**Diagram:**
- $v_0$, $v_1$, $v_2$, $v_3$, $v_4$, $v_i$, etc., with coloring $c$. The intervals are ordered along a line, with $v_0$ at one end and $v_i$ at another. The colors are assigned from left to right, with $m$ being the maximum color.
Coloring Directional Interval Graphs

Theorem 1: A coloring $c$ computed by $\text{GreedyColoring}$ 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 with 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 with intervals containing the left endpoint of $v_i$. 

\[ \text{coloring } c \]

\[ m \]

\[ S_0 \]

\[ \vdots \]

\[ S_4 \]
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 $v_i$)

- We can show that for any two steps $S_i$ and $S_j$, every pair of intervals is adjacent in the transitive closure $G^+$. 
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 $v_i$)

- We can show that for any two steps $S_i$ and $S_j$, 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 \( v_i \))

- We can show that for any two steps \( S_i \) and \( S_j \), every pair of intervals is adjacent in the transitive closure \( G^+ \).

\[
S = \bigcup S_i \text{ is a clique in } G^+
\]

\[
S \text{ alone requires } m \text{ colors in } G\]

\[
\Rightarrow S = \bigcup S_i \text{ is a clique in } G^+\]

\[
\Rightarrow S \text{ alone requires } m \text{ colors in } G\]
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:
Theorem 2: Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

Proof sketch:
We model an instance $\Phi$ of 3-SAT as a mixed interval graph $G_\Phi$. 
Theorem 2:
Deciding whether a mixed interval graph admits a \( k \)-coloring is NP-complete.

Proof sketch:
We model an instance \( \Phi \) of 3-SAT as a mixed interval graph \( G_\Phi \).

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 $\Phi$ of 3-SAT as a mixed interval graph $G_\Phi$.

variable gadget for each variable $v_i$:

\[ v_i \text{ is true} \]
Theorem 2: Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

Proof sketch:
We model an instance $\Phi$ of 3-SAT as a mixed interval graph $G_\Phi$.

variable gadget for each variable $v_i$: $v_i$ is false
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 $\Phi$ of 3-SAT as a mixed interval graph $G_\Phi$.

clause $c_j$ containing literal $v_i$: 

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

Proof sketch:
We model an instance $\Phi$ of 3-SAT as a mixed interval graph $G_\Phi$.

clause $c_j$ containing literal $v_i$: clause $c_k$ containing literal $\neg v_i$: $v_i$ is false
$v_i$ is true
Theorem 2: Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

Proof sketch:

We model an instance $\Phi$ of 3-SAT as a mixed interval graph $G_\Phi$.

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 $\Phi$ of 3-SAT as a mixed interval graph $G_{\Phi}$.

fix positions by adding "frame" intervals
Theorem 2:
Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

Proof sketch:
We model an instance $\Phi$ of 3-SAT as a mixed interval graph $G_{\Phi}$.

fix positions by adding “frame” intervals
**Theorem 2:**
Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

**Proof sketch:**

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

Proof sketch:

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

Proof sketch:

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

Proof sketch:
clause gadget:

![Diagram]( visualization of the proof sketch)

- Coloring
- $v_i$ is false
- $s_j$
- $o_i^j$
- $s_k$
- $b_i^j$
- $b_i^k$
- $o_i^k$
- $v_i^{true}$
- $v_i^{false}$
- upper free strip
- lower free strip
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:
Theorem 2:
Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

Proof sketch:
clause gadget:

6$n$ colors
($n := \# \text{variables}$)
Theorem 2: Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

Proof sketch:

clause gadget:

6$n$ colors ($n := \# \text{variables}$)
Theorem 2: Deciding whether a mixed interval graph admits a \( k \)-coloring is NP-complete.

Proof sketch:

clause gadget:

6\( n \) + 1 colors
\((n := \# \text{ variables})\)
Theorem 2:
Deciding whether a mixed interval graph admits a $k$-coloring is NP-complete.

Proof sketch:
clause gadget:

$\Phi$ is satisfiable $\iff G_\Phi$ admits a coloring with $6n$ colors

$6n + 1$ colors
($n := \# \text{variables}$)
We have introduced the natural concept of directional interval graphs.
We have introduced the natural concept of directional interval graphs.

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

$n := \# \text{ vertices}$
We have introduced the natural concept of directional interval graphs.

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

In layered graph drawing, this corresponds to routing “left-going” edges orthogonally to the fewest horizontal tracks. (Symmetrically “right-going”.)
We have introduced the natural concept of directional interval graphs.

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

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)
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.

- 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.

- 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.

- 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.
- 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.)

Can we do better?
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.
- 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.)

can we do better?