Visualization of graphs

Partial visibility representation extension

With SPQR-trees

Jonathan Klawitter · Summer semester 2020
SPQR-tree

- An **SPQR-tree** $T$ is a decomposition of a planar graph $G$ by **separation pairs**.
An **SPQR-tree** $T$ is a decomposition of a planar graph $G$ by **separation pairs**.
An **SPQR-tree** $T$ is a decomposition of a planar graph $G$ by *separation pairs*.

The nodes of $T$ are of four types:
- **S** nodes represent a series composition
- **P** nodes represent a parallel composition
- **Q** nodes represent a single edge
- **R** nodes represent 3-connected (*rigid*) subgraphs
An **SPQR-tree** $T$ is a decomposition of a planar graph $G$ by separation pairs.

The nodes of $T$ are of four types:
- $S$ nodes represent a series composition
- $P$ nodes represent a parallel composition
- $Q$ nodes represent a single edge
- $R$ nodes represent 3-connected (*rigid*) subgraphs

A decomposition tree of a series-parallel graph is an SPQR-tree without $R$ nodes.
An **SPQR-tree** $T$ is a decomposition of a planar graph $G$ by *separation pairs*.

The nodes of $T$ are of four types:
- **S** nodes represent a series composition
- **P** nodes represent a parallel composition
- **Q** nodes represent a single edge
- **R** nodes represent 3-connected (*rigid*) subgraphs

A decomposition tree of a series-parallel graph is an SPQR-tree without R nodes.

$T$ represents all planar embeddings of $G$. 
SPQR-tree

- An **SPQR-tree** $T$ is a decomposition of a planar graph $G$ by **separation pairs**.

- The nodes of $T$ are of four types:
  - $S$ nodes represent a series composition
  - $P$ nodes represent a parallel composition
  - $Q$ nodes represent a single edge
  - $R$ nodes represent 3-connected (rigid) subgraphs

- A decomposition tree of a series-parallel graph is an SPQR-tree without $R$ nodes.

- $T$ represents all planar embeddings of $G$.

- $T$ can be computed in $O(n)$ time. [Gutwenger, Mutzel ’01]
SPQR-tree example

G

14
9
8
7
6
5
4
3
2
1
10
13
11
12

SPQR-tree example

G

reference edge

root
SPQR-tree example
SPQR-tree example
SPQR-tree example
SPQR-tree example
SPQR-tree example
SPQR-tree example
SPQR-tree example
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed 0-width vertical sightlines? Do all visibilities induce edges?
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed 0-width vertical sightlines? Do all visibilities induce edges?

Models.
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed **0-width** vertical sightlines? Do all visibilities induce edges?

**Models.**

- **Strong:** Edge $uv \iff$ unobstructed **0-width** vertical sightlines
Bar visibility representation

- Vertices correspond to horizontal open line segments called **bars**
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed **0-width** vertical sightlines? Do all visibilities induce edges?

**Models.**

- **Strong:** Edge $uv \iff$ unobstructed **0-width** vertical sightlines
- **$\varepsilon$:** Edge $uv \iff \varepsilon$ wide vertical sightlines for $\varepsilon > 0$
Bar visibility representation

- Vertices correspond to horizontal open line segments called bars
- **Edges** correspond to vertical unobstructed vertical sightlines
- What about unobstructed 0-width vertical sightlines? Do all visibilities induce edges?

**Models.**

- **Strong:** Edge $uv \iff$ unobstructed 0-width vertical sightlines
- $\varepsilon$: Edge $uv \iff$ $\varepsilon$ wide vertical sightlines for $\varepsilon > 0$
- **Weak:** Edge $uv \Rightarrow$ unobstructed vertical sightlines exists, i.e., any subset of visible pairs
Problems

Weak
Problems

weak

strong
Problems
Problems

weak

strong

$\varepsilon$
Problems

weak

strong

\( \varepsilon \)
Recognition problem.
Given a graph $G$, decide if there exists a weak/strong/ε bar visibility representation $\psi$ of $G$. 
Problems

**Recognition problem.**
Given a graph $G$, decide if there exists a weak/strong/$\varepsilon$ bar visibility representation $\psi$ of $G$.

**Construction problem.**
Given a graph $G$, construct a weak/strong/$\varepsilon$ bar visibility representation $\psi$ of $G$ when one exists.
Problems

Recognition problem.
Given a graph $G$, decide if there exists a weak/strong/$\epsilon$ bar visibility representation $\psi$ of $G$.

Construction problem.
Given a graph $G$, construct a weak/strong/$\epsilon$ bar visibility representation $\psi$ of $G$ when one exists.

Partial Representation Extension (& Construction) problem.
Given a graph $G$ and a set of bars $\psi'$ of $V' \subset V(G)$, decide if there exists a weak/strong/$\epsilon$ bar visibility representation $\psi$ of $G$ where $\psi|_{V'} = \psi'$ (and construct $\psi$ when it exists).
Background
Weak Bar Visibility.

- All planar graphs. [Tamassia & Tollis 1986; Wismath 1985]
- Linear time recognition and construction [T&T ’86]
- Representation Extension is NP-complete [Chaplick et al. ’14]
Weak Bar Visibility.
- All planar graphs. [Tamassia & Tollis 1986; Wismath 1985]
- Linear time recognition and construction [T&T ’86]
- Representation Extension is NP-complete [Chaplick et al. ’14]

Strong Bar Visibility.
- NP-complete to recognize [Andreae ’92]
Background

\[ \varepsilon \text{-Bar Visibility.} \]

- Planar graphs that can be embedded with all \textbf{cut vertices} on the outerface. [T&T 1986, Wismath '85]
- Linear time recognition and construction [T&T '86]
- What about Representation Extension?
Background

ε-Bar Visibility.

■ Planar graphs that can be embedded with all cut vertices on the outerface. [T&T 1986, Wismath '85]
■ Linear time recognition and construction [T&T '86]
■ What about Representation Extension?
  Let’s see!
ε-bar visibility and st-graphs

Recall that an **st-graph** is a planar digraph $G$ with exactly one source $s$ and one sink $t$ where $s$ and $t$ occur on the outer face of an embedding of $G$. 
ε-bar visibility and st-graphs

Recall that an **st-graph** is a planar digraph $G$ with exactly one source $s$ and one sink $t$ where $s$ and $t$ occur on the outer face of an embedding of $G$.

**Observation.**
st-orientations correspond to ε-bar visibility representations.
$\varepsilon$-bar visibility and st-graphs

Recall that an \textit{st-graph} is a planar digraph $G$ with exactly one source $s$ and one sink $t$ where $s$ and $t$ occur on the outer face of an embedding of $G$.

\textbf{Observation.} st-orientations correspond to $\varepsilon$-bar visibility representations.
ε-bar visibility and st-graphs

Recall that an **st-graph** is a planar digraph $G$ with exactly one source $s$ and one sink $t$ where $s$ and $t$ occur on the outer face of an embedding of $G$.

**Note** that testing whether an acyclic planar digraph has a weak bar visibility representation is NP-complete.

Observation.

st-orientations correspond to ε-bar visibility representations.
\( \varepsilon \)-bar visibility and st-graphs

Recall that an **st-graph** is a planar digraph \( G \) with exactly one source \( s \) and one sink \( t \) where \( s \) and \( t \) occur on the outer face of an embedding of \( G \).

**Note** that testing whether an acyclic planar digraph has a weak bar visibility representation is NP-complete.

- This is upward planarity testing!
  
  [Garg & Tamassia ’01]

**Observation.**

st-orientations correspond to \( \varepsilon \)-bar visibility representations.
ε-bar visibility and st-graphs

Recall that an st-graph is a planar digraph $G$ with exactly one source $s$ and one sink $t$ where $s$ and $t$ occur on the outer face of an embedding of $G$.

- ε-bar visibility testing is easily done via st-graph recognition.

Observation.

st-orientations correspond to ε-bar visibility representations.
ε-bar visibility and st-graphs

Recall that an **st-graph** is a planar digraph $G$ with exactly one source $s$ and one sink $t$ where $s$ and $t$ occur on the outer face of an embedding of $G$.

- ε-bar visibility testing is easily done via st-graph recognition.
- Strong bar visibility recognition... open?

**Observation.**

st-orientations correspond to ε-bar visibility representations.
**ε-bar visibility and st-graphs**

Recall that an **st-graph** is a planar digraph $G$ with exactly one source $s$ and one sink $t$ where $s$ and $t$ occur on the outer face of an embedding of $G$.

- ε-bar visibility testing is easily done via st-graph recognition.
- Strong bar visibility recognition... open?
- In a **rectangular** bar visibility representation $\psi(s)$ and $\psi(t)$ span an enclosing rectangle.

**Observation.**

St-orientations correspond to ε-bar visibility representations.
Results and outline

**Theorem 1.** [Chaplick et al. ’18]

**Rectangular $\varepsilon$-Bar Visibility Representation Extension** can be solved in $O(n \log^2 n)$ time for $st$-graphs.

- Dynamic program via SPQR-trees
Results and outline

**Theorem 1.** [Chaplick et al. ’18]
Rectangular $\varepsilon$-Bar Visibility Representation Extension can be solved in $O(n \log^2 n)$ time for $st$-graphs.

- Dynamic program via SPQR-trees
- Easier version: $O(n^2)$
Results and outline

Theorem 1. [Chaplick et al. ’18]
Rectangular $\varepsilon$-Bar Visibility Representation Extension can be solved in $O(n \log^2 n)$ time for $st$-graphs.

- Dynamic program via SPQR-trees
- Easier version: $O(n^2)$

Theorem 2. [Chaplick et al. ’18]
$\varepsilon$-Bar Visibility Representation Ext. is NP-complete.

- Reduction from Planar Monotone 3-SAT
Results and outline

**Theorem 1.** [Chaplick et al. ’18]  
**Rectangular** $\varepsilon$-Bar Visibility Representation Extension can be solved in $O(n \log^2 n)$ time for $st$-graphs.

- Dynamic program via SPQR-trees
- Easier version: $O(n^2)$

**Theorem 2.** [Chaplick et al. ’18]  
$\varepsilon$-Bar Visibility Representation Ext. is NP-complete.

- Reduction from Planar Monotone 3-SAT

**Theorem 3.** [Chaplick et al. ’18]  
$\varepsilon$-Bar Visibility Representation Ext. is NP-complete for (series-parallel) $st$-graphs when restricted to the integer grid (or if any fixed $\varepsilon > 0$ is specified).

- Reduction from 3-Partition
Representation extension for st-graphs

Theorem 1'.

Rectangular $\varepsilon$-Bar Visibility Representation Extension can be solved in $O(n^2)$ time for $st$-graphs.
Theorem 1'.

Rectangular $\varepsilon$-Bar Visibility Representation Extension can be solved in $O(n^2)$ time for $st$-graphs.

- Simplify with assumption on y-coordinates
- Look at connection to SPRQ-trees – tiling
- Solve problems for S, P and R nodes
- Dynamic program via SPQR-tree
y-coordinate invariant

- Let $G$ be an $st$-graph, and $\psi'$ be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \to \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) = \text{the y-coordinate of } \psi'(v)$.
  - for each edge $(u, v)$, $y(u) < y(v)$.
y-coordinate invariant

- Let $G$ be an $st$-graph, and $\psi'$ be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \rightarrow \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) =$ the y-coordinate of $\psi'(v)$.
  - for each edge $(u, v)$, $y(u) < y(v)$.

**Lemma 1.**

$G$ has a representation extending $\psi'$ iff $G$ has a representation $\psi$ extending $\psi'$ where the y-coordinates of the bars are as in $y$. 
y-coordinate invariant

- Let $G$ be an $st$-graph, and $\psi'$ be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \to \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) = \text{the y-coordinate of } \psi'(v)$.
  - for each edge $(u, v)$, $y(u) < y(v)$.

**Lemma 1.**

$G$ has a representation extending $\psi'$ iff $G$ has a representation $\psi$ extending $\psi'$ where the y-coordinates of the bars are as in $y$.

**Proof idea.** The relative positions of adjacent bars must match the order given by $y$.
So, we can adjust the y-coordinates of any solution to be as in $y$ by sweeping from bottom-to-top.
y-coordinate invariant

- Let $G$ be an $st$-graph, and $\psi'$ be a representation of $V' \subseteq V(G)$.
- Let $y : V(G) \rightarrow \mathbb{R}$ such that
  - for each $v \in V'$, $y(v) =$ the $y$-coordinate of $\psi'(v)$.
  - for each edge $(u, v)$, $y(u) < y(v)$.

**Lemma 1.**
$G$ has a representation extending $\psi'$ iff $G$ has a representation $\psi$ extending $\psi'$ where the $y$-coordinates of the bars are as in $y$.

**Proof idea.** The relative positions of adjacent bars must match the order given by $y$. So, we can adjust the $y$-coordinates of any solution to be as in $y$ by sweeping from bottom-to-top.

We can now assume all $y$-coordinates are given!
But why do SPQR-trees help?
But why do SPQR-trees help?
But why do SPQR-trees help?

**Lemma 2.** The SPQR-tree of an \(st\)-graph \(G\) induces a recursive **tiling** of any \(\varepsilon\)-bar visibility representation of \(G\).
But why do SPQR-trees help?

Lemma 2. The SPQR-tree of an $st$-graph $G$ induces a recursive tiling of any $\varepsilon$-bar visibility representation of $G$. 

Solve tiles bottom-up
Tiles

**Convention.** Orange bars are from the partial representation

\[ \psi(t) \]

\[ \psi(s) \]
Tiles

**Convention.** Orange bars are from the partial representation

\[
\psi(t)
\]

\[
\psi(s)
\]

**Observation.**
The bounding box (tile) of any solution \( \psi \), contains the bounding box of the partial representation.
Tiles

Convention. Orange bars are from the partial representation

Observation. The bounding box (tile) of any solution $\psi$, contains the bounding box of the partial representation.

How many different tiles can we really have?
Types of tiles

- Right Fixed – due to the orange bar
- Left Loose – due to the orange bar
Types of tiles

- Right Fixed – due to the orange bar
- Left Loose – due to the orange bar

Left Fixed – due to the orange bar
Right Loose – due to the orange bar
Types of tiles

- **Right Fixed** – due to the orange bar
- **Left Loose** – due to the orange bar

- **Left Fixed** – due to the orange bar
- **Right Loose** – due to the orange bar
Types of tiles

- Right Fixed – due to the orange bar
- Left Loose – due to the orange bar

- Left Fixed – due to the orange bar
- Right Loose – due to the orange bar
Types of tiles

- Right **Fixed** – due to the orange bar
- Left **Loose** – due to the orange bar

Left **Fixed** – due to the orange bar
Right **Loose** – due to the orange bar

Four different types: **FF, FL, LF, LL**
P nodes

\[ \psi(t) \]

\[ \psi(s) \]
P nodes

\[ \psi(t) \]

\[ \psi(s) \]
$P$ nodes

$\psi(t)$

$\psi(s)$

$\psi(s)$
P nodes

- Children of P node with **prescribed bars** occur in given left-to-right order
- But there might be some **gaps**...
Children of P node with prescribed bars occur in given left-to-right order

But there might be some gaps...
Children of P node with prescribed bars occur in given left-to-right order.

But there might be some gaps...

**Idea.**

Greedily fill the gaps by preferring to “stretch” the children with prescribed bars.
P nodes

- Children of P node with prescribed bars occur in given left-to-right order
- But there might be some gaps...

**Idea.**
Greedily fill the gaps by preferring to “stretch” the children with prescribed bars.

**Outcome.**
After processing, we must know the valid types for the corresponding subgraphs.
$\psi(t)$

$\psi(s)$
This fixed vertex means we can only make a Fixed-Fixed representation!
This fixed vertex means we can only make a Fixed-Fixed representation!
This fixed vertex means we can only make a Fixed-Fixed representation!

Here we have a chance to make all (LL, FL, LF, FF) types.
S nodes

Here we have a chance to make all (LL, FL, LF, FF) types.

How does this work?

This fixed vertex means we can only make a Fixed-Fixed representation!
R nodes with 2-SAT formulation
R nodes with 2-SAT formulation
R nodes with 2-SAT formulation

- for each child
  - 2 variables encoding fixed/loose type of its tile
  - restriction clauses to subsets of \{FF, FL, LF, LL\}
R nodes with 2-SAT formulation

- for each child
  - 2 variables encoding fixed/loose type of its tile
  - restriction clauses to subsets of \{FF, FL, LF, LL\}

- for each face
  - 2 variables encoding position of the splitting line
  - consistency clauses
R nodes with 2-SAT formulation

- for each child
  - 2 variables encoding fixed/loose type of its tile
  - restriction clauses to subsets of \{FF, FL, LF, LL\}

- for each face
  - 2 variables encoding position of the splitting line
  - consistency clauses

- ordering clauses
  - quadratically many
R nodes with 2-SAT formulation

- For each child:
  - 2 variables encoding fixed/loose type of its tile
  - Restriction clauses to subsets of \{FF, FL, LF, LL\}

- For each face:
  - 2 variables encoding position of the splitting line
  - Consistency clauses

- Ordering clauses
  - Quadratically many

- Tricky part: use only \(O(n \log^2 n)\) clauses
NP-hardness of RepExt in general case

Theorem 2.
\(\varepsilon\)-Bar Visibility Representation Ext. is NP-complete.

- Reduction from Planar Monotone 3-SAT
NP-hardness of RepExt in general case

Theorem 2.
ε-Bar Visibility Representation Ext. is NP-complete.

- Reduction from Planar Monotone 3-SAT

\[
\begin{align*}
\neg x_1 & \lor x_2 & \lor x_3 \\
\neg x_2 & \lor x_3 & \lor x_4 \\
\neg x_1 & \lor x_2 & \lor x_4 \\
\neg x_1 & \lor x_4 & \lor x_5 \\
\end{align*}
\]

NP-complete [Berg & Khosravi '10]
NP-hardness of RepExt in general case

Wire Transmission
transmitting
true and false
NP-hardness of RepExt in general case

Remark. The following details omit the copying gadgets used for multiple occurrences of the variables.
NP-hardness of RepExt in general case

\[
\begin{array}{c}
\top \\
\downarrow \\
x \\
\rightarrow \\
\downarrow \\
a \\
\rightarrow \\
y \\
\downarrow \\
\bot \\
\end{array}
\]

\[
\begin{array}{c}
\top \\
\downarrow \\
x \\
\rightarrow \\
\downarrow \\
b \\
\rightarrow \\
y \\
\downarrow \\
\bot \\
\end{array}
\]

\[
\begin{array}{c}
\top \\
\downarrow \\
x \\
\rightarrow \\
\downarrow \\
a \\
\rightarrow \\
y \\
\downarrow \\
\bot \\
\end{array}
\]

\[
\begin{array}{c}
\top \\
\downarrow \\
x \\
\rightarrow \\
\downarrow \\
b \\
\rightarrow \\
y \\
\downarrow \\
\bot \\
\end{array}
\]

\[
\begin{array}{c}
\top \\
\downarrow \\
x \\
\rightarrow \\
\downarrow \\
a \\
\rightarrow \\
y \\
\downarrow \\
\bot \\
\end{array}
\]

NOT gate
NP-hardness of RepExt in general case

Note: the bars of $x$ and $y$ cannot occur between $a$ and $b$ since $a$ and $b$ are not supposed to be adjacent to either of $\perp$ and $\top$. 
NP-hardness of RepExt in general case

OR gate
NP-hardness of RepExt in general case

**OR gate**

- **Subtle point:** only need to guarantee that “false” values transmit
NP-hardness of RepExt in general case

OR gate

subtle point: only need to guarantee that “false” values transmit
NP-hardness of RepExt in general case
NP-hardness on the Integer Grid (or fixed $\varepsilon$)

**Theorem 3.**

$\varepsilon$-Bar Visibility Representation Ext. is NP-complete for (series-parallel) $st$-graphs when restricted to the integer grid (or if any fixed $\varepsilon > 0$ is specified).

- Reduction from 3-Partition
NP-hardness on the Integer Grid (or fixed $\varepsilon$)

**Theorem 3.**
$\varepsilon$-Bar Visibility Representation Ext. is NP-complete for (series-parallel) $st$-graphs when restricted to the **integer grid** (or if any fixed $\varepsilon > 0$ is specified).

**3-Partition.**
*Input:* A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.
*Question:* Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into $m$ triples such that the total sum of each triple is exactly $w$?

- Strongly NP-complete [Garey & Johnson ’79]
NP-hardness on the Integer Grid (or fixed $\varepsilon$)

3-Partition.

**Input:** A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.

**Question:** Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into $m$ triples such that the total sum of each triple is exactly $w$?

$$a_i \rightarrow$$
NP-hardness on the Integer Grid (or fixed $\varepsilon$)

3-Partition.

**Input:** A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.

**Question:** Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into $m$ triples such that the total sum of each triple is exactly $w$?

$$a_i \rightarrow H_i$$

\[
\begin{array}{c}
0 & 1 & 2 & a_i-2 & a_i-1 & a_i \\
1 & 2 & 3 & \psi(s_i) & \psi(t_i) & \text{...} \\
0 & 1 & 2 & a_i-2 & a_i-1 & a_i \\
\end{array}
\]
NP-hardness on the Integer Grid (or fixed $\varepsilon$)

**3-Partition.**

**Input:** A set of positive integers $w, a_1, a_2, \ldots, a_{3m}$ such that for each $i = 1, \ldots, 3m$, we have $\frac{w}{4} < a_i < \frac{w}{2}$.

**Question:** Can $\{a_1, \ldots, a_{3m}\}$ be partitioned into $m$ triples such that the total sum of each triple is exactly $w$?
Discussion

- **rectangular** $\varepsilon$-Bar Visibility Representation Extension can be solved in $O(n \log^2 n)$ time for $st$-graphs.

- $\varepsilon$-Bar Visibility Representation Extension is NP-complete.

- $\varepsilon$-Bar Visibility Representation Extension is NP-complete for (series-parallel) $st$-graphs when restricted to the *Integer Grid* (or if any fixed $\varepsilon > 0$ is specified).
Discussion

- **rectangular** $\varepsilon$-Bar Visibility Representation Extension can be solved in $O(n \log^2 n)$ time for $st$-graphs.

- $\varepsilon$-Bar Visibility Representation Extension is NP-complete.

- $\varepsilon$-Bar Visibility Representation Extension is NP-complete for (series-parallel) $st$-graphs when restricted to the Integer Grid (or if any fixed $\varepsilon > 0$ is specified).

Open Problems:

- Can **rectangular** $\varepsilon$-Bar Visibility Representation Extension can be solved in polynomial time on $st$-graphs? DAGs?

- Can **Strong** Bar Visibility Recognition / Representation Extension can be solved in polynomial time on $st$-graphs?
Literature

Main source:
- [Chaplick, Guśpiel, Gutowski, Krawczyk, Liotta ’18] The Partial Visibility Representation Extension Problem

Referenced papers:
- [Gutwenger, Mutzel ’01] A Linear Time Implementation of SPQR-Trees
- [Wismath ’85] Characterizing bar line-of-sight graphs
- [Tamassia, Tollis ’86] Algorithms for visibility representations of planar graphs
- [Andreae ’92] Some results on visibility graphs
- [Chaplick, Dorbec, Kratchovíl, Montassier, Stacho ’14] Contact representations of planar graphs: Extending a partial representation is hard