Algorithms for graph visualization

Divide and Conquer - Series-Parallel Graphs
Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

Lower bound on the area.

Symmetry display for series-parallel graphs.
Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

Lower bound on the area.

Symmetry display for series-parallel graphs.
Series-parallel Graphs

Graph $G$ is **series-parallel**, if

- It contains a single edge $(s, t)$ ($s$-source, $t$-sink)
- It consists of two series-parallel graphs $G_1, G_2$ with sources $s_1, s_2$ and sinks $t_1, t_2$ which are combined using one of the following rules:

**Series composition:**
Identify $t_1$ and $s_2$, $s_1$ is the source of $G$, $t_2$ is the sink of $G$

**Parallel composition:**
Identify $s_1, s_2$ and set it to be source of $G$
Identify $t_1, t_2$ and set it to be sink of $G$
A decomposition tree of $G$, which is a binary tree $T$ with nodes of three types: S,P and Q-type.
Series-parallel Graphs. Decomposition Tree.

A **decomposition tree** of $G$, which is a binary tree $T$ with nodes of three types: S,P and Q-type.

- If $G$ is a single edge, then the corresponding node is Q-node.
A decomposition tree of $G$, which is a binary tree $T$ with nodes of three types: S,P and Q-type.

- If $G$ is a single edge, then the corresponding node is Q-node.
- If $G$ is a parallel composition of $G_1$ (with tree $T_1$) and $G_2$ (with tree $T_2$), then the root of $T$ is P-node and $T_1$ is its left subtree, $T_2$ is its right subtree.
Series-parallel Graphs. Decomposition Tree.

A **decomposition tree** of $G$, which is a binary tree $T$ with nodes of three types: S,P and Q-type.

- If $G$ is a single edge, then the corresponding node is Q-node.
- If $G$ is a parallel composition of $G_1$ (with tree $T_1$) and $G_2$ (with tree $T_2$), then the root of $T$ is P-node and $T_1$ is its left subtree, $T_2$ is its right subtree.
- If $G$ is a series composition of $G_1$ (with tree $T_1$) and $G_2$ (with tree $T_2$), then the root of $T$ is S-node and $T_1$ is its left subtree, $T_2$ is its right subtree.

![Diagram of decomposition tree](image)
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.
Series-parallel Graphs. Decomposition Example.

6 - 10
Series-parallel Graphs. Decomposition Example.
**Lemma**

Series-parallel graphs are acyclic and planar.
Lemma
Series-parallel graphs are acyclic and planar.

Work with your neighbour(s) and then share

- Using the induction on the decomposition prove that a series-parallel graph is planar

5 min

Flowcharts

PERT-Diagrams
(Program Evaluation and Review Technique)

Flowcharts

Computational Complexity: Linear time algorithms for $NP$-hard problems (e.g. Maximum Matching, Maximum Independent Set, Hamiltonian Completion).

PERT-Diagrams

(Program Evaluation and Review Technique)
Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

Lower bound on the area.

Symmetry display for series-parallel graphs.
Conventions & Aesthetics for SP-graphs

Drawing Conventions

Drawing Aesthetics

Let’s brainstorm!
Conventions & Aesthetics for SP-graphs

Drawing Conventions
- Planarity
- Straight-line edges
- Upward

Drawing Aesthetics

Let’s brainstorm!
Conventions & Aesthetics for SP-graphs

**Drawing Conventions**
- Planarity
- Straight-line edges
- Upward

**Drawing Aesthetics**
- Area
- Symmetry

Let’s brainstorm!
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):

![Diagram](image.png)
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

Do you see any problem?
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)
Straight-line Drawing of SP-Graphs

**Divide & Conquer Algorithm, using the decomposition tree**

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

11 - 8

change embedding!
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G')$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G')$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

change embedding!
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

11 - 10

change embedding!
Straight-line Drawing of SP-Graphs

Divide & Conquer Algorithm, using the decomposition tree

- Draw graph $G$ inside a right-angled isosceles bounding triangle $\Delta(G)$
- Q-Nodes (Induction base):
- S-Nodes (series composition)
- P-Nodes (parallel composition)

change embedding!
Straight-line Drawing of SP-Graphs

What makes parallel composition possible without creating crossings?
What makes parallel composition possible without creating crossings?
Straight-line Drawing of SP-Graphs

What makes parallel composition possible without creating crossings?
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?
What makes parallel composition possible without creating crossings?
Straight-line Drawing of SP-Graphs

What makes parallel composition possible without creating crossings?
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?

Assume the following holds:
\[ \text{angle}(v) \text{ does not contain any vertex} \]
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?

Assume the following holds: \( \text{angle}(v) \) does not contain any vertex.

This condition **is** preserved during the induction step.
Straight-line Drawing of SP-Graphs

- What makes parallel composition possible without creating crossings?

Assume the following holds: \( \text{angle}(v) \) does not contain any vertex

This condition is preserved during the induction step.

Lemma

The drawing produced by the algorithm is planar.
Work with your neighbour(s) and then share

What is the asymptotic upper bound on the area of the drawing? **Hint:** Think in terms of edges.
Straight-line Drawing of SP-Graphs

Work with your neighbour(s) and then share

- What is the asymptotic upper bound on the area of the drawing? **Hint:** Think in terms of edges.

**Lemma**

The area of the produced drawing is $O(m^2)$, $m$ is the number of edges.

**Theorem**

A series-parallel graph $G$ (with variable embedding) admits an upward planar straight-line drawing with $O(n^2)$ area. The isomorphic components of $G$ have congruent drawings up to a translation.
Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

Lower bound on the area.

Symmetry display for series-parallel graphs.
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]
There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

Proof:

\[
\begin{align*}
G_0 & \quad G_{n+1} \\
\ \ s_0 & \quad t_0 \\
\ & G_n \\
\ s_n & \quad t_n \\
\ s_{n+1} & \quad t_{n+1}
\end{align*}
\]
Lower Bound for the Area

**Theorem [Bertolazzi et al. 94]**

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$. 

Proof:

![Diagram of a series-parallel graph $G_n$ with vertices $s_0$, $t_0$, $s_n$, $t_n$, $G_0$, and $G_{n+1}$]
Lower Bound for the Area

**Theorem [Bertolazzi et al. 94]**

There exists a $2^n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

**Proof:**

![Diagram showing proof of lower bound for area](image)
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]
There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

Proof:
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

Proof:

\[ G_0 \rightarrow G_n \rightarrow G_{n+1} \]
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

Proof:
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a \(2n\)-vertex series-parallel graph \(G_n\) such that any upward planar drawing of \(G_n\) respecting embedding requires area \(\Omega(4^n)\).

Proof:

\(G_0\) and \(G_{n+1}\) are shown in the diagram. The theorem states that for any upward planar drawing of \(G_n\) respecting embedding, the area required is \(\Omega(4^n)\).
Lower Bound for the Area

**Theorem [Bertolazzi et al. 94]**

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

**Proof:**

![Diagram of the proof](image)
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$. 

Proof:

We have that: $\text{Area}(\Pi) > 2 \cdot \text{Area}(G_n)$
Lower Bound for the Area

**Theorem [Bertolazzi et al. 94]**

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

**Proof:**

- We have that: $\text{Area}(\Pi) > 2 \cdot \text{Area}(G_n)$
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]
There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

Proof:

- We have that: $Area(\Pi) > 2 \cdot Area(G_n)$
Lower Bound for the Area

**Theorem [Bertolazzi et al. 94]**

There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

**Proof:**

- We have that: $\text{Area}(\Pi) > 2 \cdot \text{Area}(G_n)$
- $\text{Area}(G_{n+1}) \geq 2 \cdot \text{Area}(\Pi)$
Lower Bound for the Area

Theorem [Bertolazzi et al. 94]
There exists a $2n$-vertex series-parallel graph $G_n$ such that any upward planar drawing of $G_n$ respecting embedding requires area $\Omega(4^n)$.

Proof:
- We have that: $\text{Area}(\Pi) > 2 \cdot \text{Area}(G_n)$
- $\text{Area}(G_{n+1}) \geq 2 \cdot \text{Area}(\Pi)$
- $\text{Area}(G_{n+1}) \geq 4 \cdot \text{Area}(G_n)$
Overview

Series - parallel graph. Definition and Decomposition.

Algorithm for upward straight-line drawing.

Lower bound on the area.

Symmetry display for series-parallel graphs.
Property of the Algorithm
Property of the Algorithm

Do You Remember?
Property of the Algorithm

nicer???
Property of the Algorithm

Algorithm
closer???

horizontal symmetry

nicer???
Property of the Algorithm

vertical symmetry

nicer???
Property of the Algorithm

rotational symmetry

nicer???
**Graph Automorphism**

**Definition: Automorphism of a digraph**

An **automorphism** of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

- $(u, v) \in E \iff (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \iff (\pi(v), \pi(u)) \in E$
Definition: Automorphism of a digraph

An **automorphism** of a directed graph \( G = (V, E) \) is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

- \((u, v) \in E \iff (\pi(u), \pi(v)) \in E\), or
- \((u, v) \in E \iff (\pi(v), \pi(u)) \in E\)

**Example of automorphism** is a permutation of vertex set \( a, b, c, d, e, f, h, g \) defined by the rules \( b \rightarrow c \rightarrow b, a \rightarrow d \rightarrow a, \) so \( d, c, b, a, e, f, h, g \)
Graph Automorphism

Definition: Automorphism of a digraph

An automorphism of a directed graph \( G = (V, E) \) is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

\[
(u, v) \in E \iff (\pi(u), \pi(v)) \in E, \text{ or } (u, v) \in E \iff (\pi(v), \pi(u)) \in E
\]

The set of all automorphisms (direction preserving and reversing) forms the automorphism group of \( G \).
Graph Automorphism

Definition: Automorphism of a digraph

An automorphism of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

- $(u, v) \in E \iff (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \iff (\pi(v), \pi(u)) \in E$

The set of all automorphisms (direction preserving and reversing) forms the automorphism group of $G$.

Finding an automorphism group of a graph is isomorphism complete, that is equivalent to testing whether two graphs are isomorphic.
Graph Automorphism

Definition: Automorphism of a digraph

An **automorphism** of a directed graph $G = (V, E)$ is a permutation of the vertex set which preserves adjacency of the vertices and either preserves or reverses all the directions of the edges:

- $(u, v) \in E \iff (\pi(u), \pi(v)) \in E$, or
- $(u, v) \in E \iff (\pi(v), \pi(u)) \in E$

- The set of all automorphisms (direction preserving and reversing) forms the **automorphism group** of $G$.

- Finding an automorphism group of a graph is *isomorphism complete*, that is equivalent to testing whether two graphs are isomorphic.

- For planar graphs, graphs with bounded degree isomorphism problem has polynomial-time algorithms (for more see citations in [HEL00]).
Geometric Automorphism

Geometric realizability of automorphisms:
Geometric Automorphism

- Geometric realizability of automorphisms:

Pick one of the graphs above and try to list all its automorphisms. Which of them are represented as symmetries?

2 min
Geometric Automorphism

Geometric realizability of automorphisms:

This drawing displays the automorphism $1 \to 2 \to 3 \to 4 \to 1$ as rotational symmetry. But does not show the $1 \to 2 \to 3 \to 1$.

This drawing displays the automorphism $1 \to 2 \to 3 \to 1$ as rotational symmetry but not the $1 \to 2 \to 3 \to 4 \to 1$.

19 - 3
Geometric Automorphism

Geometric realizability of automorphisms:

This drawing displays the automorphism $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$ as rotational symmetry. But does not shows the $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$

Automorphisms $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$ and $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ can not be displayed simultaneously.

19 - 4
Geometric Automorphism

Geometric realizability of automorphisms:

- Automorphisms 1 → 2 → 3 → 4 → 1 and 1 → 2 → 3 → 1 cannot be displayed simultaneously.

This drawing displays the automorphism 1 → 2 → 3 → 4 → 1 as rotational symmetry. But does not shows the 1 → 2 → 3 → 1.

Automorphism 1 → 2 → 3 → 1, 4 → 5 → 4 is not geometrically representable. But 1 → 3 → 1, 4 → 5 → 4 is representable as vertical symmetry.
Geometric Automorphism

Geometric realizability of automorphisms:

- This drawing displays the automorphism $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$ as rotational symmetry. But does not show the $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$.

- This drawing displays the automorphism $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$ as rotational symmetry but not the $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$.

- Automorphism $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$, $4 \rightarrow 5 \rightarrow 4$ is not geometrically representable. But $1 \rightarrow 3 \rightarrow 1$, $4 \rightarrow 5 \rightarrow 4$ is representable as vertical symmetry.

- An automorphism group $P$ of a graph is geometric, if there exists a drawing of $G$ that displays each element of $P$ as a symmetry.

- For general graphs it is $NP$-hard to find a geometric automorphism of a graph.

- For planar graphs, planar geometric automorphisms can be found in polynomial time. For outerplanar graphs and trees in linear time.

19 - 6
Symmetries in SP-Graphs

\[ \pi_{\text{vert}} \]

\[ \nu \quad \pi(\nu) \]

\[ \nu \quad \pi(\nu) \]

\[ \pi(\nu) \quad \pi(u) \]

\[ \nu \quad \pi(\nu) \]

\[ \pi(u) \]

20 - 1
Symmetries in SP-Graphs

\[ \pi_{\text{vert}} \]

\[ \pi_{\text{hor}} \]

\[ \pi(u) \]

\[ \pi(v) \]

\[ u \quad \pi(u) \]

\[ v \quad \pi(v) \]
Symmetries in SP-Graphs

\[ \pi_{\text{vert}} \]

\[ \pi_{\text{hor}} \]

\[ \pi_{\text{rot}} \]
Symmetries in SP-Graphs

\( \pi_{\text{vert}} \)

\( u \)

\( \pi(u) \)

\( v \)

\( \pi(v) \)

\( \pi_{\text{hor}} \)

\( \pi(u) \)

\( \pi(v) \)

\( \pi_{\text{rot}} \)

\( \pi(u) \)

\( \pi(v) \)

\( \{ \pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}} \} \)
A geometric automorphism group $P$ of a graph $G$ is upward planar, if there exists an upward planar drawing of $G$ that displays each element of $P$ as a symmetry.
Symmetries in SP-Graphs

- A geometric automorphism group $P$ of a graph $G$ is upward planar, if there exists an upward planar drawing of $G$ that displays each element of $P$ as a symmetry.

- How does a geometric automorphism group for a series-parallel graph look like?
Symmetries in SP-Graphs

Theorem (Hong, Eades, Lee '00) [HEL00]

An upward planar automorphism group of a series-parallel digraph is either

\[ \{ \text{id} \} \]
\[ \{ \text{id}, \pi \} \text{ with } \pi \in \{ \pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}} \} \]
\[ \{ \text{id}, \pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}} \} \].
Symmetries in SP-Graphs

The automorphism group of maximum size can be found in linear time.

\[
\{ \pi_{\text{vert}}, \pi_{\text{hor}}, \pi_{\text{rot}} \}
\]
Symmetries in SP-Graphs

- The automorphism group of maximum size can be found in linear time.
- Given a maximum size automorphism group of a series-parallel graph, a polyline upward planar drawing that displays this automorphism can be constructed in linear time as well.
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism

- \( \text{code}(G) \) - two graphs at the same level have the same code iff they are isomorphic
- \( \text{tuple}(G) \) - set of the codes of the children (sorted for a P-node)
Vertical Automorphism

- \textit{code}(G) - two graphs at the same level have the same code iff they are isomorphic.
- \textit{tuple}(G) - set of the codes of the children (sorted for a P-node).

\[ \text{tuple}(G) = <1, 1, 2> \]
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]

\[
\langle 1, 1 \rangle \quad \langle 1, 1 \rangle \quad \langle 1, 1 \rangle \quad \langle 1, 1 \rangle
\]

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]

\[
\langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle \quad \langle 0 \rangle
\]
Vertical Automorphism
Vertical Automorphism
Vertical Automorphism

21 - 15
Vertical Automorphism
Vertical Automorphism

Algorithm constructing codes

- Set $\text{tuple}(G_i) = \langle 0 \rangle$ for all Q-nodes $G_i$ of $G$. 
Algorithm constructing codes

- Set $\text{tuple}(G_i) = \langle 0 \rangle$ for all Q-nodes $G_i$ of $G$.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node $G'$ at depth $t$ with children $G_1, \ldots, G_k$ set $\text{tuple}(G') = \langle \text{code}(G_1), \ldots, \text{code}(G_k) \rangle$. If $G'$ is a P-node, sort $\text{tuple}(G')$ in non-decreasing order.
Vertical Automorphism

**Algorithm constructing codes**

- Set $\text{tuple}(G_i) = \langle 0 \rangle$ for all Q-nodes $G_i$ of $G$.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node $G'$ at depth $t$ with children $G_1, \ldots, G_k$, set $\text{tuple}(G') = \langle \text{code}(G_1), \ldots, \text{code}(G_k) \rangle$. If $G'$ is a P-node, sort $\text{tuple}(G')$ in non-decreasing order.
  - Sort all the nodes at depth $t$ lexicographically according to tuples.
Vertical Automorphism

Algorithm constructing codes

- Set \( \text{tuple}(G_i) = \langle 0 \rangle \) for all Q-nodes \( G_i \) of \( G \).
- For each \( t = \max \text{depth}(G), \ldots, 0 \)
  - For each S- or P-node \( G' \) at depth \( t \) with children \( G_1, \ldots, G_k \)
    set \( \text{tuple}(G') = \langle \text{code}(G_1), \ldots, \text{code}(G_k) \rangle \). If \( G' \) is a P-node, sort \( \text{tuple}(G') \) in non-decreasing order.
- Sort all the nodes at depth \( t \) lexicographically according to tuples.
- For each component \( G' \) at depth \( t \), compute \( \text{code}(G') \) as follows. Assign the integer 1 to those components represented by the first distinct tuple, assign 2 to the components with the second type of tuple, and etc.
Vertical Automorphism

Algorithm constructing codes

- Set $\text{tuple}(G_i) = \langle 0 \rangle$ for all Q-nodes $G_i$ of $G$.
- For each $t = \max \text{depth}(G), \ldots, 0$
  - For each S- or P-node $G'$ at depth $t$ with children $G_1, \ldots, G_k$ set $\text{tuple}(G') = \langle \text{code}(G_1), \ldots, \text{code}(G_k) \rangle$. If $G'$ is a P-node, sort $\text{tuple}(G')$ in non-decreasing order.
- Sort all the nodes at depth $t$ lexicographically according to tuples.
- For each component $G'$ at depth $t$, compute $\text{code}(G')$ as follows. Assign the integer 1 to those components represented by the first distinct tuple, assign 2 to the components with the second type of tuple, and etc.

Lemma

Two nodes $u$ and $v$ at the same depth of the decomposition tree of $G$ represent isomorphic subgraphs of $G$ iff $\text{code}(u) = \text{code}(v)$. 
Vertical Automorphism

- Let $G$ be composed out of $G_1 \ldots G_n$ through series or parallel composition, $tuple(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $tuple(G)$ to decide whether $G$ has a vertical automorphism?
**Vertical Automorphism**

- Let $G$ be composed out of $G_1 \ldots G_n$ through series or parallel composition, $\text{tuple}(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $\text{tuple}(G)$ do decide whether $G$ has a vertical automorphism?

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is an S-node, then $G$ has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.
Vertical Automorphism

- Let $G$ be composed out of $G_1 \ldots G_n$ through series or parallel composition, $\text{tuple}(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $\text{tuple}(G)$ to decide whether $G$ has a vertical automorphism?

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is an S-node, then $G$ has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

**Proof:**
- Assume $G$ has a vertical automorphism $\alpha$
Vertical Automorphism

- Let $G$ be composed out of $G_1 \ldots G_n$ through series or parallel composition, $\text{tuple}(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $\text{tuple}(G)$ to decide whether $G$ has a vertical automorphism?

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is an S-node, then $G$ has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

**Proof:**
- Assume $G$ has a vertical automorphism $\alpha$
- Then $\alpha$ “fixes” all the components
Vertical Automorphism

- Let $G$ be composed out of $G_1 \ldots G_n$ through series or parallel composition, $\text{tuple}(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $\text{tuple}(G)$ do decide whether $G$ has a vertical automorphism?

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is an S-node, then $G$ has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

**Proof:**

- Assume $G$ has a vertical automorphism $\alpha$
- Then $\alpha$ “fixes” all the components
- Therefore each of the series components has a vertical automorphism
Vertical Automorphism

- Let $G$ be composed out of $G_1 \ldots G_n$ through series or parallel composition, $\text{tuple}(G)$ contains the codes of $G_1, \ldots, G_n$.
- How can we use $\text{tuple}(G)$ do decide whether $G$ has a vertical automorphism?

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is an S-node, then $G$ has a vertical automorphism iff each of $G_1, \ldots, G_k$ has a vertical automorphism.

**Proof:**

- Assume $G$ has a vertical automorphism $\alpha$
- Then $\alpha$ “fixes” all the components
- Therefore each of the series components has a vertical automorphism
- If each of $G_1, \ldots, G_n$ has a vertical isomorphism, arrange them as in Figure.

23 $G_6$ is an S-node
**Vertical Automorphism**

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is a P-node, consider a partition of $C_j = \{G_i : 1 \leq i \leq k, \text{code}(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|C_j|$ are even $\implies$ has a vertical automorphism.
- If there exists a unique $j$, such that $|C_j|$ is odd $\implies G$ has a vertical automorphism iff graphs of $C_j$ have a vertical automorphism.
- If there exists $|C_i|, |C_j|$ with $i \neq j$, both odd $\implies G$ does not have a vertical automorphism.

**Proof:**

- Arrange components as in Figure.

$G$ is P-node, $\text{tuple}(G) = <1 \ldots 1, 2 \ldots 2, \ldots>$

24 - 1
**Vertical Automorphism**

**Lemma (Hong, Eades, Lee '00) [HEL00]**

If $G$ is a P-node, consider a partition of $C_j = \{G_i : 1 \leq i \leq k, \text{code}(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|C_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique $j$, such that $|C_j|$ is odd $\Rightarrow G$ has a vertical automorphism iff graphs of $C_j$ have a vertical automorphism.
- If there exists $|C_i|, |C_j|$ with $i \neq j$, both odd $\Rightarrow G$ does not have a vertical automorphism.

**Proof:**

$$\text{tuple}(G) = \langle 1 \ldots 1, 2 \ldots 2, 3 \ldots 3, \ldots \rangle$$

24 - 2 $\begin{cases} \text{odd} & \text{even} \\ \text{even} \end{cases}$
Vertical Automorphism

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is a P-node, consider a partition of $C_j = \{G_i : 1 \leq i \leq k, \text{code}(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|C_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique $j$, such that $|C_j|$ is odd $\Rightarrow G$ has a vertical automorphism iff graphs of $C_j$ have a vertical automorphism.
- If there exists $|C_i|, |C_j|$ with $i \neq j$, both odd $\Rightarrow G$ does not have a vertical automorphism.

**Proof:**

- Any vertical automorphism “fixes” a member of $C_j$, therefore it has a vertical automorphism.

$$\text{tuple}(G) = < 1 \ldots 1, 2 \ldots 2, 3 \ldots 3, \ldots >$$

24 - 3

**odd even even**
**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If \( G \) is a P-node, consider a partition of \( \mathcal{C}_j = \{G_i : 1 \leq i \leq k, \text{code}(G_i) = j\} \), \( j = 1, \ldots, k \) into classes of isomorphic graphs.

- If \( \forall j\), \( |\mathcal{C}_j| \) are even \( \Rightarrow \) has a vertical automorphism.
- If there exists a unique \( j \), such that \( |\mathcal{C}_j| \) is odd \( \Rightarrow \) \( G \) has a vertical automorphism iff graphs of \( \mathcal{C}_j \) have a vertical automorphism.
- If there exists \( |\mathcal{C}_i|, |\mathcal{C}_j| \) with \( i \neq j \), both odd \( \Rightarrow \) \( G \) does not have a vertical automorphism.

\[
\text{tuple}(G) = \langle 1\ldots1, 2\ldots2, 3\ldots3, \ldots \rangle
\]

24 - 4

**Proof:**

- Any vertical automorphism “fixes” a member of \( \mathcal{C}_j \), therefore it has a vertical automorphism.
- Conversely, arrange as in figure.
Vertical Automorphism

Lemma (Hong, Eades, Lee ’00) [HEL00]

If $G$ is a P-node, consider a partition of $C_j = \{G_i : 1 \leq i \leq k, code(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|C_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique $j$, such that $|C_j|$ is odd $\Rightarrow G$ has a vertical automorphism iff graphs of $C_j$ have a vertical automorphism.
- If there exists $|C_i|, |C_j|$ with $i \neq j$, both odd $\Rightarrow G$ does not have a vertical automorphism.

Proof:

tuple$(G) = \langle 1 \ldots 1, 2 \ldots 2, 3 \ldots 3, \ldots \rangle$

24 - 5 $\langle$ odd, odd, even $\rangle$
Vertical Automorphism

**Lemma (Hong, Eades, Lee ’00) [HEL00]**

If $G$ is a P-node, consider a partition of $C_j = \{G_i : 1 \leq i \leq k, \text{code}(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|C_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique $j$, such that $|C_j|$ is odd $\Rightarrow$ $G$ has a vertical automorphism iff graphs of $C_j$ have a vertical automorphism.
- If there exists $|C_i|, |C_j|$ with $i \neq j$, both odd $\Rightarrow$ $G$ does not have a vertical automorphism.

**Proof:**

- Any vertical automorphism has to “fix” two distinct components.

\[
tuple(G) = \langle 1 \ldots 1, 2 \ldots 2, 3 \ldots 3, \ldots \rangle
\]

\[24 - 6 \quad \text{odd} \quad \text{odd} \quad \text{even}\]
Lemma (Hong, Eades, Lee ’00) [HEL00]

If $G$ is a P-node, consider a partition of $C_j = \{G_i : 1 \leq i \leq k, \text{code}(G_i) = j\}$, $j = 1, \ldots, k$ into classes of isomorphic graphs.

- If $\forall j$, $|C_j|$ are even $\Rightarrow$ has a vertical automorphism.
- If there exists a unique $j$, such that $|C_j|$ is odd $\Rightarrow G$ has a vertical automorphism iff graphs of $C_j$ have a vertical automorphism.
- If there exists $|C_i|, |C_j|$ with $i \neq j$, both odd $\Rightarrow G$ does not have a vertical automorphism.

Proof:

- Any vertical automorphism has to “fix” two distinct components.
- In both components we can find a path on which some vertices are aligned on the axis. Contradicts planarity.

$$\text{tuple}(G) = <1 \ldots 1, 2 \ldots 2, 3 \ldots 3, \ldots>$$

24 - 7

odd even
Series-parallel graphs

- Book Di Battista et al: Chapter 3.2, 11.1
- Skript: Chapter 6.2
- [HEL00] Hong, Eades, Lee *Drawing series parallel digraphs symmetrically* CGTA 2000