Graph Planarity Testing with Hierarchical Embedding Constraints

Giuseppe Liotta
Alessandra Tappini
Ignaz Rutter

University of Perugia, Italy
University of Passau, Germany

EuroCG 2020
36th European Workshop on Computational Geometry
March 16-18, 2020 - Würzburg, Germany
Constraints in Graph Drawings

- In many contexts, data can be represented as networks of interconnected elements
- Information visualization is often based on graph representations
- Graph representations need to take into account layout rules
In many contexts, data can be represented as networks of interconnected elements.

Information visualization is often based on graph representations.

Graph representations need to take into account layout rules.

Database diagrams links between attributes should enter the tables only at the left or right side.
Constraints in Graph Drawings

- In many contexts, data can be represented as networks of interconnected elements
- Information visualization is often based on graph representations
- Graph representations need to take into account layout rules

UML class diagrams
generalization edges should leave a class object at the top and enter a base class object at the bottom
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.
  - Four sets: $E_1, E_2, E_3, E_4$
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.
  - Four sets: $E_1, E_2, E_3, E_4$
  - Fixed cyclic order: $E_1, E_2, E_3, E_4$
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.
  - Four sets: $E_1, E_2, E_3, E_4$
  - Fixed cyclic order: $E_1, E_2, E_3, E_4$
  - The edges of $E_1, E_2, E_3$ can be arbitrarily permuted.
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.

- Four sets: $E_1, E_2, E_3, E_4$
- Fixed cyclic order: $E_1, E_2, E_3, E_4$
- The edges of $E_1, E_2, E_3$ can be arbitrarily permuted
- $E_4$ is partitioned into subsets $E'_4, E''_4, E'''_4$
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.

- Four sets: $E_1, E_2, E_3, E_4$
- Fixed cyclic order: $E_1, E_2, E_3, E_4$
- The edges of $E_1, E_2, E_3$ can be arbitrarily permuted.
- $E_4$ is partitioned into subsets $E_4', E_4'', E_4'''$
- $E_4''$ must appear between $E_4'$ and $E_4'''$
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.

- Four sets: $E_1, E_2, E_3, E_4$
- Fixed cyclic order: $E_1, E_2, E_3, E_4$
- The edges of $E_1, E_2, E_3$ can be arbitrarily permuted.
- $E_4$ is partitioned into subsets $E_4', E_4'', E_4'''$
- $E_4''$ must appear between $E_4'$ and $E_4'''$
- The edges of $E_4'$ can be arbitrarily permuted.
Hierarchical Embedding Constraints

- These layout rules impose restrictions on the admissible embeddings for a graph.
- We consider restrictions on allowed cyclic orders of the edges incident to a vertex.

- Four sets: $E_1, E_2, E_3, E_4$
- Fixed cyclic order: $E_1, E_2, E_3, E_4$
- The edges of $E_1, E_2, E_3$ can be arbitrarily permuted.
- $E_4$ is partitioned into subsets $E_4', E_4'', E_4'''$
- $E_4''$ must appear between $E_4'$ and $E_4'''$
- The edges of $E_4'$ can be arbitrarily permuted.
- The edges of and $E_4''$ have only two possible orders that are the reverse of one another.
FPQ-trees

- Represent a family of permutations on a set of elements
- Each element is a leaf

- **F-nodes**: The order of children is fixed
- **Q-nodes**: The order of children may be reversed
- **P-nodes**: The order of children may be arbitrarily permuted
FPQ-trees

- Represent a family of permutations on a set of elements
  - Each element is a leaf

- Embeddings constraints are modeled by means of FPQ-trees
  - Represent the cyclic orders of the edges incident to a vertex
  - Each edge is a leaf in $T$

- F-nodes: The order of children is fixed
- Q-nodes: The order of children may be reversed
- P-nodes: The order of children may be arbitrarily permuted
Graph Planarity Testing

- **Edge crossings** negatively affect the readability of graph representations

Cognitive experiments:
- Purchase - 1997
- Purchase, Carrington, Allder - 2002
- Ware, Purchase, Colpoys, McGill - 2002
Graph Planarity Testing

- **Edge crossings** negatively affect the readability of graph representations.

Cognitive experiments:
- Purchase - 1997
- Purchase, Carrington, Allder - 2002
- Ware, Purchase, Colpoys, McGill - 2002

- The **graph planarity testing** problem is at the heart of graph algorithms and of their applications.
  - **Remark.** Minimizing the total number of crossings in a graph drawing is NP-hard [Garey, Johnson - 1983]
Graph Planarity Testing + Embedding Constraints

- Introduced by [Gutwenger, Klein, Mutzel - 2008]
- They model each hierarchical embedding constraint as a constraint tree

**oc-nodes:** The order of children is fixed

**mc-nodes:** The order of children may be reversed

**gc-nodes:** The order of children may be arbitrarily permuted
Graph Planarity Testing + Embedding Constraints

- Introduced by [Gutwenger, Klein, Mutzel - 2008]
- They model each hierarchical embedding constraint as a constraint tree

- Constrained planarity testing is linear-time solvable

oc-nodes: The order of children is fixed
mc-nodes: The order of children may be reversed
gc-nodes: The order of children may be arbitrarily permuted
Graph Planarity Testing + Embedding Constraints

- Introduced by [Gutwenger, Klein, Mutzel - 2008]
- They model each hierarchical embedding constraint as a constraint tree

- Constrained planarity testing is linear-time solvable
- Constraint trees ≡ FPQ-trees
FPQ-Choosable Graph

A (multi-)graph $G$ and a mapping $D$ that associates each vertex $v$ of $G$ with a set $D(v)$ of FPQ-trees whose leaves represent the edges incident to $v$. 
FPQ-Choosable Graph

A (multi-)graph $G$ and a mapping $D$ that associates each vertex $v$ of $G$ with a set $D(v)$ of FPQ-trees whose leaves represent the edges incident to $v$.

FPQ-Choosable Planarity Testing

**INPUT:** An FPQ-choosable graph $(G, D)$

**QUESTION:** Does $G$ admit a planar embedding such that, for each vertex $v$, the cyclic order of the edges incident to $v$ is encoded by an FPQ-tree in $D(v)$?
**FPQ-Choosable Planarity Testing**

**FPQ-Choosable Graph**

A (multi-)graph $G$ and a mapping $D$ that associates each vertex $v$ of $G$ with a set $D(v)$ of FPQ-trees whose leaves represent the edges incident to $v$.

**FPQ-Choosable Planarity Testing**

**INPUT:** An FPQ-choosable graph $(G, D)$

**QUESTION:** Does $G$ admit a planar embedding such that, for each vertex $v$, the cyclic order of the edges incident to $v$ is encoded by an FPQ-tree in $D(v)$?

**Remark.** If $|D(v)| = 1$ for each $v$, then the problem can be solved in linear time [Gutwenger et al. - 2008]
## Our Results

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{max}$</td>
<td>NP-complete - (Theorem 1)</td>
</tr>
<tr>
<td>$t$</td>
<td>W[1]-hard - (Theorem 2)</td>
</tr>
<tr>
<td>$D_{max}, t$</td>
<td>FPT - (Theorem 3)</td>
</tr>
</tbody>
</table>

$D_{max}$: Maximum number of FPQ-trees per vertex  
$t$: Treewidth of $G$
### Our Results

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{max}}$</td>
<td>NP-complete - (Theorem 1)</td>
</tr>
<tr>
<td>$t$</td>
<td>W[1]-hard - (Theorem 2)</td>
</tr>
<tr>
<td>$D_{\text{max}}, t$</td>
<td>FPT - (Theorem 3)</td>
</tr>
</tbody>
</table>

- $D_{\text{max}}$: Maximum number of FPQ-trees per vertex
- $t$: Treewidth of $G$

**FPQ-Choosable Planarity Testing** is not FPT if parameterized by $t$ only or by $D_{\text{max}}$ only
Our Results

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_{\text{max}}$</td>
<td>NP-complete - (Theorem 1)</td>
</tr>
<tr>
<td>$t$</td>
<td>W[1]-hard - (Theorem 2)</td>
</tr>
<tr>
<td>$D_{\text{max}}, t$</td>
<td>FPT - (Theorem 3)</td>
</tr>
</tbody>
</table>

$D_{\text{max}}$: Maximum number of FPQ-trees per vertex
$t$: Treewidth of $G$

FPQ-Choosable Planarity Testing is not FPT if parameterized by $t$ only or by $D_{\text{max}}$ only

Theorem 3. FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{\text{max}}$ → $O(D_{\text{max}}^{\frac{9}{4}t} \cdot n^2 + n^3)$-time algorithm
Theorem 1

FPQ-Choosable Planarity Testing with a bounded number of FPQ-trees per vertex (>1) is NP-complete. It remains NP-complete even when the FPQ-trees have only P-nodes.

- Reduction from the 3-edge-coloring problem for triconnected cubic non-planar graphs
**Theorem 1**

FPQ-Choosable Planarity Testing with a bounded number of FPQ-trees per vertex \(>1\) is NP-complete. It remains NP-complete even when the FPQ-trees have only P-nodes.

- Reduction from the 3-edge-coloring problem for triconnected cubic non-planar graphs

**Theorem 2**

FPQ-Choosable Planarity Testing parameterized by treewidth is \(W[1]\)-hard. It remains \(W[1]\)-hard even when the FPQ-trees have only P-nodes.

- Parameterized reduction from the list coloring problem
Theorem 3

FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{\text{max}} \rightarrow O(D_{\text{max}}^{\frac{9}{4}t} \cdot n^2 + n^3)$-time algorithm
Theorem 3

FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{max} \rightarrow O(D_{max}^{4t} \cdot n^2 + n^3)$-time algorithm.

Proof outline:
Theorem 3

FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{\text{max}} \rightarrow O(D_{\text{max}}^{\frac{9}{4}t} \cdot n^2 + n^3)$-time algorithm

Proof outline:
1. Compute the SPQR-decomposition tree $\mathcal{T}$ of $G$ rooted at an arbitrary $Q$-node
Theorem 3

FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{\text{max}} \rightarrow O(D_{\text{max}}^{\frac{9}{4}t} \cdot n^2 + n^3)$-time algorithm

Proof outline:

1. Compute the SPQR-decomposition tree $\mathcal{T}$ of $G$ rooted at an arbitrary Q-node
2. Visit $\mathcal{T}$ from the leaves to the root
FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{\text{max}} \rightarrow O(D_{\text{max}}^{\frac{9}{4}} t \cdot n^2 + n^3)$-time algorithm

Proof outline:
1. Compute the SPQR-decomposition tree $T$ of $G$ rooted at an arbitrary $Q$-node
2. Visit $T$ from the leaves to the root
3. At each step of the visit, equip the current node $\mu$ with the set $\Psi(\mu)$ of admissible tuples
Theorem 3

FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{max} \rightarrow O(D_{max}^{9t} \cdot n^2 + n^3)$-time algorithm

Proof outline:

1. Compute the SPQR-decomposition tree $T$ of $G$ rooted at an arbitrary Q-node
2. Visit $T$ from the leaves to the root
3. At each step of the visit, equip the current node $\mu$ with the set $\Psi(\mu)$ of admissible tuples
4. Do we reach the root?
   - YES $\Rightarrow (G, D)$ is FPQ-choosable planar
   - NO: We find a node such that $\Psi(\mu) = \emptyset \Rightarrow (G, D)$ is not FPQ-choosable planar
• **Assignment** $A$ is a function that assigns to each vertex $v$ an FPQ-tree $T_v \in D(v)$.
Admissible Tuple

- **Assignment** $A$ is a function that assigns to each vertex $v$ an FPQ-tree $T_v \in D(v)$

- **$A$ is compatible with $G$** if there exists a planar embedding $\mathcal{E}$ such that, for each $v, \mathcal{E}$ induces a cyclic order of its incident edges that is described by $T_v$
Admissible Tuple

- **Assignment** $A$ is a function that assigns to each vertex $v$ an FPQ-tree $T_v \in D(v)$.

- $A$ is **compatible** with $G$ if there exists a planar embedding $\mathcal{E}$ such that, for each $v$, $\mathcal{E}$ induces a cyclic order of its incident edges that is described by $T_v$.

- $A$ is **consistent** with $\mathcal{E}$. 
For each internal node $\mu$ of $T$ with poles $u$ and $v$:

- $G_\mu$ is the pertinent graph
- The boundary of $T_u$ is the element that separates the edges that belong to $G_\mu$ and the edges that are external to $G_\mu$
- The boundary can be either a $Q$-node (or $F$-node) or an edge
Admissible Tuple

For each internal node $\mu$ of $T$ with poles $u$ and $v$:

- $G_\mu$ is the pertinent graph
- The boundary of $T_u$ is the element that separates the edges that belong to $G_\mu$ and the edges that are external to $G_\mu$
- The boundary can be either a $Q$-node (or $F$-node) or an edge

- If the boundary of $T_u$ is a $Q$- (or $F$-) node, it imposes an orientation $o_u$ that defines the permutation of its children
- We establish a default orientation and we call it the clockwise orientation
Tuple of a node $\mu$: $\langle T_u, T_v, o_u, o_v \rangle \in D(u) \times D(v) \times \{0,1\} \times \{0,1\}$

clockwise

counter-clockwise
Tuple of a node $\mu$: $\langle T_u, T_v, o_u, o_v \rangle \in D(u) \times D(v) \times \{0,1\} \times \{0,1\}$

A tuple is admissible for $\mu$ if there exists an assignment $A_\mu$ that is consistent with a planar embedding $E_\mu$ of $G_\mu$
Admissible Tuple

Tuple of a node $\mu$: $\langle T_u, T_v, o_u, o_v \rangle \in D(u) \times D(v) \times \{0,1\} \times \{0,1\}$

A tuple is admissible for $\mu$ if there exists an assignment $A_\mu$ that is consistent with a planar embedding $E_\mu$ of $G_\mu$

- $\Psi(\mu)$ is the set of admissible tuples for $\mu$
A tuple is admissible for $\mu$ if there exists an assignment $A_\mu$ that is consistent with a planar embedding $E_\mu$ of $G_\mu$

- $\Psi(\mu)$ is the set of admissible tuples for $\mu$
- $\Psi(\mu)$ is computed from the set of admissible tuples of the children of $\mu$
  - Depending on whether $\mu$ is an S-, P-, Q-, or R-node
FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{\text{max}} \rightarrow O(D_{\text{max}}^\frac{9}{4} t \cdot n^2 + n^3)$-time algorithm
FPQ-Choosable Planarity Testing is FPT for biconnected graphs, where the parameters are $t$ and $D_{\text{max}} \rightarrow O(D_{\text{max}}^{9t/4} n^2 + n^3)$-time algorithm.

For R-nodes, in order to compute the set of admissible tuples:

- We execute the sphere-cut decomposition of the skeleton of $\mu$.
  - It has branchwidth at most $b$ (the branchwidth of $G$).
- For a graph $G$ with treewidth $t$ and branchwidth $b > 1$, it holds

\[ b - 1 \leq t \leq \left\lceil \frac{3}{2} b \right\rceil - 1 \]  
[Robertson, Seymour - 1991]
Remarks

Let $G$ be a clustered $n$-vertex graph whose clusters have size at most $k$. Let $t$ be the treewidth of $G$. If the (multi-)graph obtained by collapsing each cluster of $G$ into a vertex is biconnected, there exists an $O(k^{2t} \cdot n^2 + n^3)$-time algorithm to test whether $G$ is NodeTrix planar with fixed sides.

Each FPQ-tree allows a possible permutation described by the matrix
Open Problems

• Theorem 1 is based on a reduction that associates 6 FPQ-trees to each vertex.
  What is the time complexity if $2 \leq D_{\text{max}} \leq 5$?

• Is it possible to extend Theorem 3 to simply connected graphs?

• Improve the time complexity of Theorem 3.

• Apply our approach to other hybrid representation models.
Thank you for your attention

alessandra.tappini@unipg.it