Algorithms for Graph Visualization

Summer Semester 2016
Lecture # 3

Graph Drawing via Canonical Orders

(Partly based on lecture slides by Philipp Kindermann & Alexander Wolff)
Planar Graphs: Background

The Canonical Order of a Planar Graph

Straight-line Drawing using a Canonical Order

Geometric Representations using Canonical Orders
Outline

Planar Graphs: Background

The Canonical Order of a Planar Graph

Straight-line Drawing using a Canonical Order

Geometric Representations using Canonical Orders
Planar Graphs: basics

A graph is **planar** when its vertices and edges can be mapped to points and curves in $\mathbb{R}^2$ such that the curves are non-crossing. A graph is **plane** when it is given with an embedding of its vertices and edges in $\mathbb{R}^2$ which certifies its planarity.

<table>
<thead>
<tr>
<th>Embeddings of $K_4$</th>
<th>Non-planar graphs $K_5$ and $K_{3,3}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Embedding 1" /></td>
<td><img src="image2" alt="Non-planar 1" /></td>
</tr>
<tr>
<td><img src="image3" alt="Embedding 2" /></td>
<td><img src="image4" alt="Non-planar 2" /></td>
</tr>
<tr>
<td><img src="image5" alt="Embedding 3" /></td>
<td><img src="image6" alt="Non-planar 3" /></td>
</tr>
<tr>
<td><img src="image7" alt="Embedding 4" /></td>
<td><img src="image8" alt="Non-planar 4" /></td>
</tr>
</tbody>
</table>
Planar Graphs: basics

A graph is **planar** when its vertices and edges can be mapped to points and curves in $\mathbb{R}^2$ such that the curves are non-crossing. A graph is **plane** when it is given with an **embedding** of its vertices and edges in $\mathbb{R}^2$ which certifies its planarity.

<table>
<thead>
<tr>
<th>Embeddings of $K_4$</th>
<th>Non-planar graphs $K_5$ and $K_{3,3}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Embeddings of $K_4$" /></td>
<td><img src="image2" alt="Non-planar graphs $K_5$ and $K_{3,3}$" /></td>
</tr>
</tbody>
</table>

How do we define the **equivalence** of planar embeddings?
Planar Graphs: basics

A graph is **planar** when its vertices and edges can be mapped to points and curves in $\mathbb{R}^2$ such that the curves are non-crossing. A graph is **plane** when it is given with an **embedding** of its vertices and edges in $\mathbb{R}^2$ which certifies its planarity.

<table>
<thead>
<tr>
<th>Embeddings of $K_4$</th>
<th>Non-planar graphs $K_5$ and $K_{3,3}$.</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Embeddings of $K_4$" /></td>
<td><img src="image2.png" alt="Non-planar graphs $K_5$ and $K_{3,3}$" /></td>
</tr>
</tbody>
</table>

▶ How do we define the **equivalence** of planar embeddings? By the sets of **inner faces** and the **outerface**.
Characterizations, Recognition, and Drawings

1. [Kuratowski 1930: *Sur le problème des courbes gauches en topologie*]
   A graph is planar iff it contains neither a $K_5$ nor a $K_{3,3}$ minor.

```
\begin{center}
\begin{tikzpicture}
  \node (1) at (0,0) [circle,fill,inner sep=2pt]{};
  \node (2) at (1,0) [circle,fill,inner sep=2pt]{};
  \node (3) at (1,1) [circle,fill,inner sep=2pt]{};
  \node (4) at (0,1) [circle,fill,inner sep=2pt]{};
  \node (5) at (-1,0) [circle,fill,inner sep=2pt]{};
  \node (6) at (-1,1) [circle,fill,inner sep=2pt]{};

  \draw (1) -- (2) -- (3) -- (4) -- (5) -- (6) -- (1);

  \node (7) at (2,0) [circle,fill,inner sep=2pt]{};
  \node (8) at (3,0) [circle,fill,inner sep=2pt]{};
  \node (9) at (3,1) [circle,fill,inner sep=2pt]{};
  \node (10) at (2,1) [circle,fill,inner sep=2pt]{};
  \node (11) at (1,0) [circle,fill,inner sep=2pt]{};
  \node (12) at (1,1) [circle,fill,inner sep=2pt]{};

  \draw (7) -- (8) -- (9) -- (10) -- (11) -- (12) -- (7);
\end{tikzpicture}
\end{center}
```
Characterizations, Recognition, and Drawings

1. [Kuratowski 1930: *Sur le problème des courbes gauches en topologie*]
   A graph is planar iff it contains neither a $K_5$ nor a $K_{3,3}$ minor.

2. [Hopcroft & Tarjan, J. ACM 1974]
   For a graph $G$ with $n$ vertices, there is an $O(n)$ time algorithm to test if $G$ is planar.
Characterizations, Recognition, and Drawings

1. [Kuratowski 1930: *Sur le problème des courbes gauches en topologie*] A graph is planar iff it contains neither a $K_5$ nor a $K_{3,3}$ minor.

2. [Hopcroft & Tarjan, J. ACM 1974] For a graph $G$ with $n$ vertices, there is an $O(n)$ time algorithm to test if $G$ is planar.

3. [Wagner 1936, Fáry 1948, Stein 1951] Every planar graph has an embedding where the edges are straight-lines.
Characterizations, Recognition, and Drawings

1. [Kuratowski 1930: *Sur le problème des courbes gauches en topologie*]
   A graph is planar iff it contains neither a $K_5$ nor a $K_{3,3}$ minor.

2. [Hopcroft & Tarjan, J. ACM 1974]
   For a graph $G$ with $n$ vertices, there is an $O(n)$ time algorithm to test if $G$ is planar.

3. [Wagner 1936, Fáry 1948, Stein 1951]
   Every planar graph has an embedding where the edges are straight-lines.

4. [Koebe 1936: *Kontaktprobleme der konformen Abbildung*]
   Every planar graph is a circle contact graph (*coin graph*). (this implies 3).
1. [Tutte 1963: \textit{How to draw a graph}]
   Every 3-connected planar has an embedding with convex polygons as its faces (i.e., implies straight-lines).
1. [Tutte 1963: *How to draw a graph*]
   Every 3-connected planar has an embedding with convex polygons as its faces (i.e., implies straight-lines).
   - Idea: place vertices in the *barycentre* of neighbours.
   - Drawback: requires large grids.

2. [de Fraysseix, Pack, Pollack 1988] Every plane triangulation can be drawn with straight-lines such that the vertices reside on a \((2n-4) \times (n-2)\) grid.
1. [Tutte 1963: *How to draw a graph*]
   Every 3-connected planar has an embedding with convex polygons as its faces (i.e., implies straight-lines).
   - Idea: place vertices in the barycentre of neighbours.
     Drawback: requires large grids.

2. [de Fraysseix, Pack, Pollack 1988]
   Every plane triangulation can be drawn with straight-lines such that the vertices reside on a \((2n - 4) \times (n - 2)\) grid.
Characterizations, Recognition, and Drawings (cont)

1. [Tutte 1963: *How to draw a graph*]
   Every 3-connected planar has an embedding with convex polygons as its faces (i.e., implies straight-lines).
   - Idea: place vertices in the *barycentre* of neighbours.
     Drawback: requires large grids.

2. [de Fraysseix, Pack, Pollack 1988]
   Every plane triangulation can be drawn with straight-lines such that the vertices reside on a \((2n - 4) \times (n - 2)\) grid.
We focus on triangulations.

- **plane triangulation** is a plane graph where every face is a triangle.

- **plane inner triangulation** is a plane graph where every face except the outerface is a triangle.
We focus on triangulations.

- **plane triangulation** is a plane graph where every face is a triangle.
- **plane inner triangulation** is a plane graph where every face except the outer face is a triangle.

... But why??
We focus on triangulations.

- **plane triangulation** is a plane graph where every face is a triangle.
- **plane inner triangulation** is a plane graph where every face except the outerface is a triangle.

... But why??

- Easy to construct from any plane graph. Many ways to triangulate each face:
We focus on triangulations.

- **plane triangulation** is a plane graph where every face is a triangle.
- **plane inner triangulation** is a plane graph where every face except the outer face is a triangle.

... But why??

- Easy to construct from any plane graph. Many ways to triangulate each face:

  ![Diagram showing triangulations]

- Triangulations are precisely the maximal planar graphs, i.e., every planar graph is a subgraph of one such graph.
We focus on triangulations.

- **plane triangulation** is a plane graph where every face is a triangle.

- **plane inner triangulation** is a plane graph where every face except the outerface is a triangle.

... But why??

- Easy to construct from any plane graph. Many ways to triangulate each face:

  - Triangulations are precisely the maximal planar graphs, i.e., every planar graph is a subgraph of one such graph.

- Can we “nicely” describe all triangulations?
Outline

Planar Graphs: Background

The Canonical Order of a Planar Graph

Straight-line Drawing using a Canonical Order

Geometric Representations using Canonical Orders
How to construct a plane triangulation?

- Start with a single edge $u_1u_2$. Let $G_2$ be this graph.
How to construct a plane triangulation?

▶ Start with a single edge \( u_1 u_2 \). Let \( G_2 \) be this graph.
▶ Add a new vertex \( u_{i+1} \) to \( G_i \) so that the neighbours of \( u_{i+1} \) are on the outerface of \( G_i \). Let \( G_{i+1} \) be this new graph.
How to construct a plane triangulation?

- Start with a single edge $u_1 u_2$. Let $G_2$ be this graph.
- Add a new vertex $u_{i+1}$ to $G_i$ so that the neighbours of $u_{i+1}$ are on the outerface of $G_i$. Let $G_{i+1}$ be this new graph.

1. Is $G_i$ a triangulation?

Steven Chaplick  ·  Lehrstuhl für Informatik I  ·  Universität Würzburg
How to construct a plane triangulation?

- Start with a single edge $u_1u_2$. Let $G_2$ be this graph.
- Add a new vertex $u_{i+1}$ to $G_i$ so that the neighbours of $u_{i+1}$ are on the outerface of $G_i$. Let $G_{i+1}$ be this new graph.

1. Is $G_i$ a triangulation?
   - No, the neighbours of $u_{i+1}$ need to be a path.

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
How to construct a plane triangulation?

- Start with a single edge $u_1 u_2$. Let $G_2$ be this graph.
- Add a new vertex $u_{i+1}$ to $G_i$ so that the neighbours of $u_{i+1}$ are on the outerface of $G_i$. Let $G_{i+1}$ be this new graph.

1. Is $G_i$ a triangulation?
   - No, the neighbours of $u_{i+1}$ need to be a path.
   - No, $u_{i+1}$ also needs at least two neighbours in $G_i$. 

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
How to construct a plane triangulation?

- Start with a single edge $u_1u_2$. Let $G_2$ be this graph.
- Add a new vertex $u_{i+1}$ to $G_i$ so that the neighbours of $u_{i+1}$ are on the outerface of $G_i$. Let $G_{i+1}$ be this new graph.

1. Is $G_i$ a triangulation?
   - No, the neighbours of $u_{i+1}$ need to be a path.
   - No, $u_{i+1}$ also needs at least two neighbours in $G_i$.
   - No, the last vertex $v_n$ needs to cover the outerface of $G_{n-1}$.
How to construct a plane triangulation?

- Start with a single edge $u_1 u_2$. Let $G_2$ be this graph.
- Add a new vertex $u_{i+1}$ to $G_i$ so that the neighbours of $u_{i+1}$ are on the outerface of $G_i$. Let $G_{i+1}$ be this new graph.

1. Is $G_i$ a triangulation?
   - No, the neighbours of $u_{i+1}$ need to be a path.
   - No, $u_{i+1}$ also needs at least two neighbours in $G_i$.
   - No, the last vertex $v_n$ needs to cover the outerface of $G_{n-1}$.
   - Yes!
How to construct a plane triangulation?

- Start with a single edge $u_1u_2$. Let $G_2$ be this graph.
- Add a new vertex $u_{i+1}$ to $G_i$ so that the neighbours of $u_{i+1}$ are on the outerface of $G_i$. Let $G_{i+1}$ be this new graph.

1. Is $G_i$ a triangulation?
   - No, the neighbours of $u_{i+1}$ need to be a path.
   - No, $u_{i+1}$ also needs at least two neighbours in $G_i$.
   - No, the last vertex $v_n$ needs to cover the outerface of $G_{n-1}$.
   - Yes!

2. Do we get all plane triangulations?
How to construct a plane triangulation?

- Start with a single edge $u_1u_2$. Let $G_2$ be this graph.
- Add a new vertex $u_{i+1}$ to $G_i$ so that the neighbours of $u_{i+1}$ are on the outerface of $G_i$. Let $G_{i+1}$ be this new graph.

1. Is $G_i$ a triangulation?
   - No, the neighbours of $u_{i+1}$ need to be a path.
   - No, $u_{i+1}$ also needs at least two neighbours in $G_i$.
   - No, the last vertex $v_n$ needs to cover the outerface of $G_{n-1}$.
   - Yes!

2. Do we get all plane triangulations?
   - Yes! But how can we prove this? (first we formalize the canonical order)
Canonical Order

A **canonical order** is a permutation $v_1, \ldots, v_n$ of the vertex set of a plane graph $G$ such that:

- $v_{i+1}$ has at least two neighbours in $G_i$.
- The neighbours of $v_{i+1}$ are consecutive in:
  
  $$C_i = (v_1 = w_1, w_2, \ldots, w_{k-1}, w_k = v_2).$$

- The neighbourhood of $v_n$ is $C_{n-1}$.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Example: How to find a Canonical Order

Idea: Start from the “last” vertex and find a “peeling” order.
Lemma: Every Plane Inner Triangulation Has a Canonical Order (CO)

- For $G = (V, E)$, proceed by induction on $|V|$. 

  Base Case: $|V| = 3$ (i.e., $G$ Triangle)

  Inductive Case: $|V| > 3$, assume we have a CO for inner plane triangulations with $|V| - 1$ vertices.

Def: A chord of $G$ is an edge connecting non-consecutive vertices on $G$'s outerface.

Claim 1: If $G$ has a vertex $v$ on its outerface which does not belong to a chord, then $G \setminus v$ is an inner plane triangulation.

Claim 2: $G$ has a vertex on its outerface which does not belong to a chord.

Proof of Claim 2: The chords are nested, i.e., some chord has no chord “above” it. This “top” chord has a vertex “above” it.

qed.
Lemma: Every Plane Inner Triangulation Has a Canonical Order (CO)

- For $G = (V, E)$, proceed by induction on $|V|$.
- Base Case: $|V| = 3$ (i.e., $G$ Triangle)
Lemma: Every Plane Inner Triangulation Has a Canonical Order (CO)

- For \( G = (V, E) \), proceed by induction on \( |V| \).
- Base Case: \( |V| = 3 \) (i.e., \( G \) Triangle)
- Inductive Case: \( |V| > 3 \), assume we have a CO for inner plane triangulations with \( |V| - 1 \) vertices.
Lemma: Every Plane Inner Triangulation Has a Canonical! Order (CO)

- For $G = (V, E)$, proceed by induction on $|V|$.
- Base Case: $|V| = 3$ (i.e., $G$ Triangle)
- Inductive Case: $|V| > 3$, assume we have a CO for inner plane triangulations with $|V| - 1$ vertices.
- Def: A **chord** of $G$ is an edge connecting non-consecutive vertices on $G$’s outerface.

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
Lemma: Every Plane Inner Triangulation Has a Canonical Order (CO)

- For $G = (V, E)$, proceed by induction on $|V|$.
- Base Case: $|V| = 3$ (i.e., G Triangle)
- Inductive Case: $|V| > 3$, assume we have a CO for inner plane triangulations with $|V| - 1$ vertices.
- Def: A **chord** of $G$ is an edge connecting non-consecutive vertices on $G$'s outerface.
- Claim 1: If $G$ has a vertex $v$ on its outerface which does not belong to a chord, then $G \setminus v$ is an inner plane triangulation.
Lemma: Every Plane Inner Triangulation Has a Canonical Order (CO)

- For $G = (V, E)$, proceed by induction on $|V|$.
- Base Case: $|V| = 3$ (i.e., $G$ Triangle)
- Inductive Case: $|V| > 3$, assume we have a CO for inner plane triangulations with $|V| - 1$ vertices.
- Def: A **chord** of $G$ is an edge connecting non-consecutive vertices on $G$’s outerface.
- Claim 1: If $G$ has a vertex $v$ on its outerface which does not belong to a chord, then $G \setminus v$ is an inner plane triangulation.
- Claim 2: $G$ has a vertex on its outerface which does not belong to a chord.

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
Lemma: Every Plane Inner Triangulation Has a Canonical Order (CO)

- For $G = (V, E)$, proceed by induction on $|V|$.
- Base Case: $|V| = 3$ (i.e., $G$ Triangle)
- Inductive Case: $|V| > 3$, assume we have a CO for inner plane triangulations with $|V| - 1$ vertices.
- Def: A chord of $G$ is an edge connecting non-consecutive vertices on $G$'s outerface.
- Claim 1: If $G$ has a vertex $v$ on its outerface which does not belong to a chord, then $G \setminus v$ is an inner plane triangulation.
- Claim 2: $G$ has a vertex on its outerface which does not belong to a chord.
- Proof of Claim 2: The chords are nested, i.e., some chord has no chord “above” it. This “top” chord has a vertex “above” it.
Lemma: Every Plane Inner Triangulation Has a Canonical Order (CO)

- For $G = (V, E)$, proceed by induction on $|V|$.
- Base Case: $|V| = 3$ (i.e., $G$ Triangle)
- Inductive Case: $|V| > 3$, assume we have a CO for inner plane triangulations with $|V| - 1$ vertices.
- Def: A **chord** of $G$ is an edge connecting non-consecutive vertices on $G$'s outerface.
- Claim 1: If $G$ has a vertex $v$ on its outerface which does not belong to a chord, then $G \setminus v$ is an inner plane triangulation.
- Claim 2: $G$ has a vertex on its outerface which does not belong to a chord.
- Proof of Claim 2: The chords are nested, i.e., some chord has no chord “above” it. This “top” chord has a vertex “above” it.
- qed.

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
Canonical Order: Algorithm

forall the \( v \in V \) do
  \[
  \text{chords}(v) \leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false};
  \]
out\( (v_1) \), out\( (v_2) \), out\( (v_n) \) \( \leftarrow \) T;
for \( k = n \) to 3 do
  \[
  \text{pick } v \neq v_1, v_2 \text{ with } \text{mark}(v) = \text{false}, \text{out}(v) = \text{true} ;
  \]
  \( v_k \leftarrow v \); \text{mark}(v) \leftarrow \text{true};
  \[
  (w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2) \leftarrow \text{Outerface}(G_{k-1});
  \]
  \( (w_p, \ldots, w_q) \leftarrow \text{unmarked neighbours of } v_k; \)
  for \( i = p \) to \( q \) do
    \[
    \text{out}(w_i) \leftarrow \text{true};
    \]
  update \( \text{chords}(\cdot) \) for \( w_p, \ldots, w_q \) and their neighbours;

▶ chords\( (v) \) is the number of chords incident to \( v \).
▶ mark\( (v) = \text{true} \iff v \) has been picked.
▶ out\( (v) = \text{true} \iff v \) is on the outerface of \( G_k \).
Canonical Order: Algorithm

forall the $v \in V$ do

- chords($v$) ← 0; out($v$) ← false; mark($v$) ← false;
- out($v_1$), out($v_2$), out($v_n$) ← T;

for $k = n$ to 3 do

- pick $v \neq v_1, v_2$ with mark($v$) = F, out($v$) = T, chords($v$) = 0;

- chords($v$) is the number of chords incident to $v$.
- mark($v$) = T $\iff$ $v$ has been picked.
- out($v$) = T $\iff$ $v$ is on the outerface of $G_k$. 

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
Canonical Order: Algorithm

forall the $v \in V$ do
  \begin{itemize}
  \item chords($v$) $\leftarrow$ 0; out($v$) $\leftarrow$ false; mark($v$) $\leftarrow$ false;
  \end{itemize}
out($v_1$), out($v_2$), out($v_n$) $\leftarrow$ T;
for $k = n$ to 3 do
  \begin{itemize}
  \item pick $v \neq v_1, v_2$ with mark($v$) = F, out($v$) = T, chords($v$) = 0;
  \item $v_k \leftarrow v$; mark($v$) $\leftarrow$ T;
  \end{itemize}

\begin{itemize}
\item chords($v$) is the number of chords incident to $v$.
\item mark($v$) = T $\iff$ $v$ has been picked.
\item out($v$) = T $\iff$ $v$ is on the outerface of $G_k$.
\end{itemize}
Canonical Order: Algorithm

\[
\text{forall the } v \in V \text{ do} \\
\quad \text{chords}(v) \leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false}; \\
\text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{T}; \\
\text{for } k = n \text{ to } 3 \text{ do} \\
\quad \text{pick } v \neq v_1, v_2 \text{ with mark}(v) = \text{F}, \text{out}(v) = \text{T}, \text{chords}(v) = 0; \\
\quad v_k \leftarrow v; \text{mark}(v) \leftarrow \text{T}; \\
\quad (w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2) \leftarrow \text{Outerface}(G_{k-1});
\]

- chords(v) is the number of chords incident to v.
- mark(v) = T ⇔ v has been picked.
- out(v) = T ⇔ v is on the outerface of G_k.
Canonical Order: Algorithm

forall the $v \in V$ do
\hspace{1em} chords($v$) ← 0; out($v$) ← false; mark($v$) ← false;
out($v_1$), out($v_2$), out($v_n$) ← T;
for $k = n$ to 3 do
\hspace{2em} pick $v \neq v_1, v_2$ with mark($v$) = F, out($v$) = T, chords($v$) = 0;
\hspace{2em} $v_k$ ← $v$; mark($v$) ← T;
\hspace{2em} ($w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$) ← Outerface($G_{k-1}$);
\hspace{2em} ($w_p, \ldots, w_q$) ← unmarked neighbours of $v_k$;

$\triangleright$ chords($v$) is the number of chords incident to $v$.
$\triangleright$ mark($v$) = T $\iff$ $v$ has been picked.
$\triangleright$ out($v$) = T $\iff$ $v$ is on the outerface of $G_k$. 
Canonical Order: Algorithm

\[
\text{forall the } v \in V \text{ do } \\
\begin{array}{c}
\_ \text{ chords}(v) \leftarrow 0; \text{ out}(v) \leftarrow \text{false}; \text{ mark}(v) \leftarrow \text{false}; \\
\text{out}(v_1), \text{ out}(v_2), \text{ out}(v_n) \leftarrow T; \\
\text{for } k = n \text{ to } 3 \text{ do } \\
\begin{array}{c}
\_ \text{ pick } v \neq v_1, v_2 \text{ with mark}(v) = F, \text{ out}(v) = T, \text{ chords}(v) = 0; \\
\_ v_k \leftarrow v; \text{ mark}(v) \leftarrow T; \\
\_ (w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2) \leftarrow \text{Outerface}(G_{k-1}); \\
\_ (w_p, \ldots, w_q) \leftarrow \text{unmarked neighbours of } v_k; \\
\_ \text{ for } i = p \text{ to } q \text{ do } \text{ out}(w_i) \leftarrow T; \\
\end{array}
\end{array}
\]

- chords\((v)\) is the number of chords incident to \(v\).
- mark\((v) = T \iff v\) has been picked.
- out\((v) = T \iff v\) is on the outerface of \(G_k\).
Canonical Order: Algorithm

\[
\text{forall the } v \in V \text{ do}
\]
\[
\quad \text{chords}(v) \leftarrow 0; \text{out}(v) \leftarrow \text{false}; \text{mark}(v) \leftarrow \text{false};
\]
\[
\text{out}(v_1), \text{out}(v_2), \text{out}(v_n) \leftarrow \text{T};
\]
\[
\text{for } k = n \text{ to } 3 \text{ do}
\]
\[
\quad \text{pick } v \neq v_1, v_2 \text{ with mark}(v) = \text{F}, \text{out}(v) = \text{T}, \text{chords}(v) = 0;
\]
\[
\quad v_k \leftarrow v; \text{mark}(v) \leftarrow \text{T};
\]
\[
\quad (w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2) \leftarrow \text{Outerface}(G_{k-1});
\]
\[
\quad (w_p, \ldots, w_q) \leftarrow \text{unmarked neighbours of } v_k;
\]
\[
\quad \text{for } i = p \text{ to } q \text{ do } \text{out}(w_i) \leftarrow \text{T};
\]
\[
\quad \text{update chords(·) for } w_p, \ldots, w_q \text{ and their neighbours;}
\]

- chords\((v)\) is the number of chords incident to \(v\).
- mark\((v) = \text{T} \iff v \text{ has been picked.}
- out\((v) = \text{T} \iff v \text{ is on the outerface of } G_k.\)
Canonical Order: Algorithm

forall the $v \in V$ do
  chords($v$) ← 0; out($v$) ← false; mark($v$) ← false;
  out($v_1$), out($v_2$), out($v_n$) ← T;
for $k = n$ to 3 do
  pick $v \neq v_1, v_2$ with mark($v$) = F, out($v$) = T, chords($v$) = 0;
  $v_k ← v$; mark($v$) ← T;
  ($w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2) ← Outerface(G_{k-1});$
  ($w_p, \ldots, w_q) ←$ unmarked neighbours of $v_k$;
  for $i = p$ to $q$ do out($w_i$) ← T;
  update chords($\cdot$) for $w_p, \ldots, w_q$ and their neighbours;

$\triangleright$ chords($v$) is the number of chords incident to $v$.
$\triangleright$ mark($v$) = T $\iff$ $v$ has been picked.
$\triangleright$ out($v$) = T $\iff$ $v$ is on the outerface of $G_k$.

Time: $O(n)$
Outline

Planar Graphs: Background

The Canonical Order of a Planar Graph

Straight-line Drawing using a Canonical Order

Geometric Representations using Canonical Orders
The main idea:

**Invariant:** $G_{k-1}$ has been drawn so that:
- $v_1$ is at $(0, 0)$ and $v_2$ is at $(2k - 6, 0)$.
- The outerface forms an $x$-monotone curve with slopes $\pm 1$. 
The main idea:

**Invariant**: $G_{k-1}$ has been drawn so that:

- $v_1$ is at $(0, 0)$ and $v_2$ is at $(2k - 6, 0)$.
- The outerface forms an $x$-monotone curve with slopes $\pm 1$.
The main idea:

Invariant: $G_{k-1}$ has been drawn so that:

- $v_1$ is at $(0, 0)$ and $v_2$ is at $(2k - 6, 0)$.
- The outerface forms an x-monotone curve with slopes $\pm 1$. 

![Diagram showing the invariant of $G_{k-1}$ with $v_1$, $v_2$, and $v_k$ marked, and the outerface forming an x-monotone curve with slopes $\pm 1$.]
The main idea:

Invariant: $G_{k-1}$ has been drawn so that:

- $v_1$ is at $(0,0)$ and $v_2$ is at $(2k - 6, 0)$.
- The outerface forms an $x$-monotone curve with slopes $\pm 1$. 

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
The main idea:

**Invariant:** $G_{k−1}$ has been drawn so that:

- $v_1$ is at $(0, 0)$ and $v_2$ is at $(2k − 6, 0)$.
- The outerface forms an $x$-monotone curve with slopes $±1$. 
The main idea:

**Invariant:** $G_{k-1}$ has been drawn so that:

- $v_1$ is at $(0, 0)$ and $v_2$ is at $(2k - 6, 0)$.
- The outerface forms an $x$-monotone curve with slopes $\pm 1$. 

Why is it a grid point?

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
Example Shift Algorithm
Example Shift Algorithm

Steven Chaplick
Lehrstuhl für Informatik I
Universität Würzburg
Example Shift Algorithm

Steven Chaplick · Lehrstuhl für Informatik I · Universität Würzburg
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm
Example Shift Algorithm

Steven Chaplick
· Lehrstuhl für Informatik I · Universität Würzburg
How do we define the “lower” set $L(v)$?

- Each inner node is covered exactly once.
- In $G$, this cover relation defines a rooted tree.
- In each $G_i$ ($i \in \{2, \ldots, n - 1\}$), it defines a forest where the outerface contains the “roots”.

![Diagram showing inner neighbours and root node $v_k$](image)
How do we define the “lower” set \( L(v) \)?

- Each inner node is covered exactly once.
- In \( G \), this cover relation defines a rooted tree.
- In each \( G_i \) \((i \in \{2, \ldots, n-1\})\), it defines a forest where the outerface contains the “roots”.

\[
\begin{align*}
V_k & \quad G_{k-1} \\
W_1 & \quad W_2 \\
W_p & \quad W_q \\
W_t & \quad W_{t-1}
\end{align*}
\]
How do we define the “lower” set $L(v)$?

- Each inner node is covered exactly once.
- In $G$, this cover relation defines a rooted tree.
- In each $G_i$ ($i \in \{2, \ldots, n-1\}$), it defines a forest where the outerface contains the “roots”.
- The trees in this forest are the “bags” shown here.
How do we define the “lower” set $L(v)$?

- Each inner node is covered exactly once.
- In $G$, this cover relation defines a rooted tree.
- In each $G_i$ ($i \in \{2, \ldots, n-1\}$), it defines a forest where the outerface contains the “roots”.
- The trees in this forest are the “bags” shown here.

Lemma

**Applying the shift algorithm maintains monotone $x$-coordinates of the outerface.**
The Shift Method: de Fraysseix, Pach und Pollack

$v_1, \ldots, v_n$: a canonical order of $G$;
for $i = 1$ to $n$ do $L(v_i) \leftarrow \{v_i\}$;
$P(v_1) \leftarrow (0, 0); P(v_2) \leftarrow (2, 0); P(v_3) \leftarrow (1, 1)$;
for $k = 4$ to $n$ do

Let $w_1 = v_1, w_2, \ldots, w_{t-1}, w_t = v_2$ be the outerface of $G_{k-1}$;
Let $w_p, \ldots, w_q$ be the neighbours of $v_k$;
for $v \in \bigcup_{j=p+1}^{q-1} L(w_j)$ do

$x(v) \leftarrow x(v) + 1$

for $v \in \bigcup_{j=q}^{t} L(w_j)$ do

$x(v) \leftarrow x(v) + 2$

$P(v_i) \leftarrow$ intersection point of the lines with slope $\pm 1$ from $P(w_p)$ and $P(w_q)$;
$L(v_i) = \bigcup_{j=p+1}^{q-1} L(w_j) \cup \{v_i\}$

Timing: $O(n^2)$. Can we do it faster?
Linear Time Shifting

- Idea 1: To compute \( x(v_k), y(v_k) \), we only need: the \( y \)-coordinates of \( w_p \) and \( w_q \) and the difference \( x(w_q) - x(w_p) \).

\[
\begin{align*}
 x(v_k) &= \frac{1}{2} (x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
y(v_k) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
x(v_k) - x(w_p) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
Linear Time Shifting

- Idea 1: To compute \( x(v_k), y(v_k) \), we only need: the \( y \)-coordinates of \( w_p \) and \( w_q \) and the difference \( x(w_q) - x(w_p) \).

\[
\begin{align*}
  x(v_k) &= \frac{1}{2} (x(w_q) + x(w_p) + y(w_q) - y(w_p)) \quad (1) \\
  y(v_k) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) + y(w_p)) \quad (2) \\
  x(v_k) - x(w_p) &= \frac{1}{2} (x(w_q) - x(w_p) + y(w_q) - y(w_p)) \quad (3)
\end{align*}
\]
Linear Time Shifting

- **Idea 1:** To compute $x(v_k), y(v_k)$, we only need: the $y$-coordinates of $w_p$ and $w_q$ and the difference $x(w_q) - x(w_p)$.

- **Idea 2:** Instead of storing explicit $x$-coordinates we store certain $x$ differences.

\[
\begin{align*}
\text{V}_k & \quad \text{W}_p \quad \text{W}_{p+1} \quad \text{W}_{q-1} \quad \text{W}_q \\
\text{W}_1 & \quad \text{V}_1 & \quad \text{W}_2 & \quad \text{W}_3 & \quad \text{V}_2 = \text{W}_t \\
\end{align*}
\]

\[
\begin{align*}
\text{(1)}: \quad x(v_k) &= \frac{1}{2}(x(w_q) + x(w_p) + y(w_q) - y(w_p)) \\
\text{(2)}: \quad y(v_k) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) + y(w_p)) \\
\text{(3)}: \quad x(v_k) - x(w_p) &= \frac{1}{2}(x(w_q) - x(w_p) + y(w_q) - y(w_p))
\end{align*}
\]
Linear Time Shifting

Idea 2: Instead of storing explicit $x$-coordinates we store certain $x$ differences. Namely, the edges from this “augmented” version of the cover tree.
Linear Time Shifting

To update the binary tree according to a new vertex $v_k$

- In the binary tree, we need the $y(v_k)$ and the $x$ differences from $v_k$ to its covered neighbour $w_{p+1}$ and to its “end” neighbours $w_p$ and $w_q$.
- Compute $y(v_k)$ with (2), and $\Delta_x(v_k, w_p)$ with (3).
- $\Delta_x(v_k, w_q) = \Delta_x(w_p, w_q) - \Delta_x(v_k, w_p)$, and $\Delta_x(v_k, v_{p+1}) = \Delta_x(w_p, w_{p+1}) - \Delta_x(v_k, w_p)$.
Outline

Planar Graphs: Background

The Canonical Order of a Planar Graph

Straight-line Drawing using a Canonical Order

Geometric Representations using Canonical Orders
Intersection Representations of Graphs

Definition
For a collection $\mathcal{S}$ of sets $S_1, \ldots, S_n$, the *intersection graph* $G(\mathcal{S})$ of $\mathcal{S}$ has vertex set $\mathcal{S}$ and edge set

$$\{ S_i S_j : i, j \in \{1, \ldots, n\}, i \neq j, \text{ and } S_i \cap S_j \neq \emptyset \}.$$ 

We call $\mathcal{S}$ an *intersection representation* of $G(\mathcal{S})$.

http://upload.wikimedia.org/wikipedia/commons/e/e9/Intersection_graph.gif
**Intersection Representations of Graphs**

**Definition**
For a collection $\mathcal{S}$ of sets $S_1, \ldots, S_n$, the *intersection graph* $G(\mathcal{S})$ of $\mathcal{S}$ has vertex set $\mathcal{S}$ and edge set

$$\{S_i S_j : i, j \in \{1, \ldots, n\}, i \neq j, \text{ and } S_i \cap S_j \neq \emptyset\}.$$  

We call $\mathcal{S}$ an *intersection representation* of $G(\mathcal{S})$.

http://upload.wikimedia.org/wikipedia/commons/e/e9/Intersection_graph.gif

Does every graph have an intersection representation?
Contact Representations of Graphs

A collection of interiorly disjoint objects \( S = \{ S_1, \ldots, S_n \} \) is called a contact representation of its intersection graph \( G(S) \).

- Some object-types: circles, line segments, triangles, rectangles, ...
- What about the domain? 2D, 3D, higher dimension, non-orientable?
- ...

Is the intersection graph of a contact representation always planar?
Contact Representations of Graphs

A collection of interiorly disjoint objects $\mathcal{S} = \{S_1, \ldots, S_n\}$ is called a contact representation of its intersection graph $G(\mathcal{S})$.

- Some object-types: circles, line segments, triangles, rectangles, ...
- What about the domain? 2D, 3D, higher dimension, non-orientable?
- ...

Is the intersection graph of a contact representation always planar? No. Not even for planar object-types!
Contact Representations of Graphs

A collection of interiorly disjoint objects \( S = \{ S_1, \ldots, S_n \} \) is called a contact representation of its intersection graph \( G(S) \).

- Some object-types: circles, line segments, triangles, rectangles, ...
- What about the domain? 2D, 3D, higher dimension, non-orientable?

Is the intersection graph of a contact representation always planar? No. Not even for planar object-types!

Which object-types can be used to represent all planar graphs?
Planar Graphs

- Contact Disk [Koebe 1936]
- Contact Triangles and T-shapes [de Fraysseix, Ossona de Mendez, Rosenstiehl 1994]
- Side Contact of 3D Boxes [Thomassen 1986]
- and many more!
Planar Graphs

- Contact Disk [Koebe 1936]
- Contact Triangles and T-shapes [de Fraysseix, Ossona de Mendez, Rosenstiehl 1994]
- Side Contact of 3D Boxes [Thomassen 1986]
- and many more!
Triangulating for representations

Goal: Prove that all planar graphs have a intersection/contact representation by some object-type $\mathcal{T}$.

- If we are given a plane graph, there are many ways to triangulate it – by adding edges or vertices. Recall, our previous triangulation picture:
Triangulating for representations

Goal: Prove that all planar graphs have a intersection/contact representation by some object-type $\mathcal{T}$.

- If we are given a plane graph, there are many ways to triangulate it – by adding edges or vertices. Recall, our previous triangulation picture:

- What is best for our goal?
Triangulating for representations

Goal: Prove that all planar graphs have a intersection/contact representation by some object-type $\mathcal{T}$.

- If we are given a plane graph, there are many ways to triangulate it – by adding edges or vertices. Recall, our previous triangulation picture:

- What is best for our goal? Adding vertices.

Lemma

*For any given object-type $\mathcal{T}$, if every planar triangulation has an intersection representation using $\mathcal{T}$-type objects, then every planar graph also can be represented using $\mathcal{T}$-type objects.*
Lemma
For any given object-type $T$, if every planar triangulation has an intersection representation using $T$-type objects, then every planar graph also can be represented using $T$-type objects.

Proof
Lemma

For any given object-type $T$, if every planar triangulation has an intersection representation using $T$-type objects, then every planar graph also can be represented using $T$-type objects.

Proof

- Start with a planar graph $G$ and triangulate $G$ to get $G'$ by adding one dummy vertex for each face.
Lemma
For any given object-type $T$, if every planar triangulation has an intersection representation using $T$-type objects, then every planar graph also can be represented using $T$-type objects.

Proof

- Start with a planar graph $G$ and triangulate $G$ to get $G'$ by adding one dummy vertex for each face.
- Now, we have a $T$-type intersection representation $R$ of $G'$. 

The more general property we are exploiting is the fact that intersection classes of graphs are hereditary, i.e., closed under the taking of induced subgraphs.
Lemma

For any given object-type $T$, if every planar triangulation has an intersection representation using $T$-type objects, then every planar graph also can be represented using $T$-type objects.

Proof

- Start with a planar graph $G$ and triangulate $G$ to get $G'$ by adding one dummy vertex for each face.
- Now, we have a $T$-type intersection representation $R$ of $G'$. 
- Remove the objects corresponding to dummy objects from $R$ and now we have $R'$ which represents precisely $G$. □
Intersection Representations of Planar Graphs

Lemma
For any given object-type $T$, if every planar triangulation has an intersection representation using $T$-type objects, then every planar graph also can be represented using $T$-type objects.

Proof

- Start with a planar graph $G$ and triangulate $G$ to get $G'$ by adding one dummy vertex for each face.
- Now, we have a $T$-type intersection representation $R$ of $G'$.
- Remove the objects corresponding to dummy objects from $R$ and now we have $R'$ which represents precisely $G$. □
Lemma

For any given object-type \( T \), if every planar triangulation has an intersection representation using \( T \)-type objects, then every planar graph also can be represented using \( T \)-type objects.

Proof

- Start with a planar graph \( G \) and triangulate \( G \) to get \( G' \) by adding one dummy vertex for each face.
- Now, we have a \( T \)-type intersection representation \( R \) of \( G' \).
- Remove the objects corresponding to dummy objects from \( R \) and now we have \( R' \) which represents precisely \( G \).

The more general property we are exploiting is the fact that intersection classes of graphs are **hereditary**, i.e., closed under the taking of induced subgraphs.
T-contact and Triangle-contact Representations

Example Representations:

Idea: Use the canonical order. Notice any interesting invariant about the two representations?
T-contact and Triangle-contact Representations

Example Representations:

![Example Representations](image)

Idea: Use the canonical order. Notice any interesting invariant about the two representations? Did something change??
Example Representations:

Idea: Use the canonical order. Notice any interesting invariant about the two representations? Did something change??

Observations:
T-contact and Triangle-contact Representations

Example Representations:

Idea: Use the canonical order. Notice any interesting invariant about the two representations? Did something change??

Observations:

- The base triangle or T-shape is precisely its position in the canonical order.
T-contact and Triangle-contact Representations

Example Representations:

Idea: Use the canonical order. Notice any interesting invariant about the two representations? Did something change??

Observations:

- The base triangle or T-shape is precisely its position in the canonical order.
- The highest point is precisely the base of its cover neighbour from above.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation. Note: we also get a T-contact representation.
T-contact and Triangle-contact Systems

Using the canonical order, we can generate a right-triangle contact representation. Note: we also get a T-contact representation.
Schnyder Realizers

- partition of the internal edges into three spanning trees
- every vertex has out-degree exactly one in $T_1$, $T_2$ and $T_3$
- vertex rule: order of edges: entering $T_1$, leaving $T_2$, entering $T_3$, leaving $T_1$, entering $T_2$, leaving $T_3$. 
3 edge-disjoint spanning trees $T_1$, $T_2$, $T_3$ cover $G$.

$T_1$, $T_2$, $T_3$ rooted at external vertices of $G$. 
Schnyder Realizers, Canonical Orders, and Representations
Exercises

1. Canonical Orders:
   1.1 Can you describe a special canonical order to build precisely the maximal outerplane graphs (i.e., outerplane inner triangulations)? (hint: how many neighbours can \( v_i \) have in \( G_i \)?)
   1.2 Can you describe a variation on the canonical order to build precisely the maximal bipartite plane graphs (i.e., every face has 4 vertices)?

2. Contact Representations:
   2.1 Show that every maximal outerplane graphs has a contact representation by: (i) rectangles; (ii) squares.
   2.2 Show that every maximal bipartite plane graph has a contact representation by: (i) rectangles; (ii) vertical and horizontal line segments.
   2.3 Show that there is a planar graph which does not have a contact representation by line segments. Note: here we do not restrict the slopes on the line segments in any way. Hint: how many edges can there be in the intersection graph of such a contact representation?