Monotone Drawings of k-Inner Planar Graphs

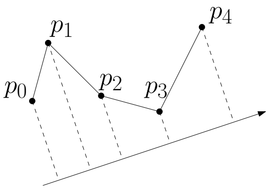
Anargyros Oikonomou   Antonios Symvonis

National Technical University of Athens, Greece

27 September 2018

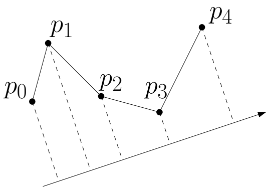
Monotone Drawings

- A path $P = \{p_0, p_1, \ldots, p_n\}$ is monotone if there exists a line l such that the projections of the vertices of P appear on l in the same order as on P.

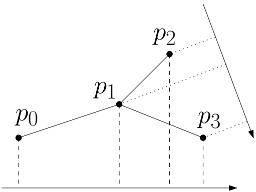


Monotone Drawings

- A path $P = \{p_0, p_1, \ldots, p_n\}$ is monotone if there exists a line l such that the projections of the vertices of P appear on l in the same order as on P.

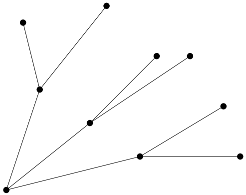


- A straight-line drawing Γ of a graph G is *monotone*, if a monotone path connects every pair of vertices.



Monotone Drawings

- A monotone drawing Γ of a tree T rooted at r is *near-convex* monotone, if for any pair of consecutive edges incident to a vertex, with the exception of a single pair of consecutive edges incident to r, form a convex angle.



What we Know so Far For Planar Monotone Drawings

What we Know so Far For Planar Monotone Drawings

- 3-connected planar graphs on $(2n-5) \times (2n-5)$

  Schnyder drawing

  [*He and He - ESA 2015*]

What we Know so Far For Planar Monotone Drawings

- 3-connected planar graphs on $(2n-5) \times (2n-5)$
  Schnyder drawing
  [*He and He - ESA 2015*]

- Optimal drawings for trees on $12n \times 12n$.
  There exists a tree that requires at least $\frac{n}{9} \times \frac{n}{9}$
  [*He and He - SIDMA 31(3)*]

What we Know so Far For Planar Monotone Drawings

- 3-connected planar graphs on $(2n-5) \times (2n-5)$

  Schnyder drawing
  [*He and He - ESA 2015*]

- Optimal drawings for trees on $12n \times 12n$.

  There exists a tree that requires at least $\frac{n}{9} \times \frac{n}{9}$
  [*He and He - SIDMA 31(3)*]

- Planar graphs on $O\left(n^2\right) \times O\left(n\right)$.

  It changes the embedding of the graph.
  [*Hossain and Rahman - TCS 607*]

What we Know so Far For Planar Monotone Drawings

- 3-connected planar graphs on $(2n-5) \times (2n-5)$

  Schnyder drawing

  [*He and He - ESA 2015*]

- Optimal drawings for trees on $12n \times 12n$.

  There exists a tree that requires at least $\frac{n}{9} \times \frac{n}{9}$

  [*He and He - SIDMA 31(3)*]

- Planar graphs on $O\left(n^2\right) \times O\left(n\right)$.

  It changes the embedding of the graph.

  [*Hossain and Rahman - TCS 607*]

- Trees on $n \times n$.

  [*Oikonomou and Symvonis - GD 2017*]

What we Know so Far For Planar Monotone Drawings

- Trees on $\left\lfloor \frac{3}{4}(n+2) \right\rfloor \times \left\lfloor \frac{3}{4}(n+2) \right\rfloor$.

    It changes the embedding and layout of the tree
    [*Oikonomou and Symvonis - arXiv*]

Class of k-Inner Planar Graphs

Bridges the gap between Outerplanar and Planar graphs.

Class of k-Inner Planar Graphs

Bridges the gap between Outerplanar and Planar graphs.

Definition (k-Inner Planar Graphs)

A k-inner planar graph is a planar graph that has a plane drawing with at most k internal vertices i.e., vertices that do not lie on the boundary of the outer face of its drawing.

Outerplanar graphs are 0-inner planar graphs.

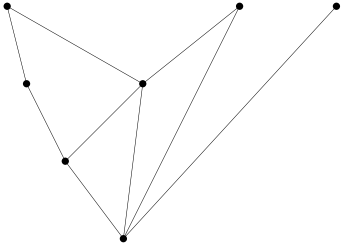
Class of k-Inner Planar Graphs

Bridges the gap between Outerplanar and Planar graphs.

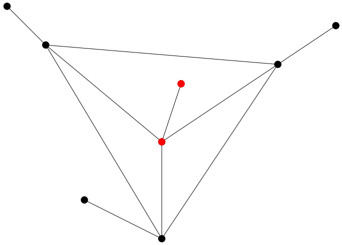
Definition (k-Inner Planar Graphs)

A k-inner planar graph is a planar graph that has a plane drawing with at most k internal vertices i.e., vertices that do not lie on the boundary of the outer face of its drawing.

Outerplanar graphs are 0-inner planar graphs.



0-inner planar graph



2-inner planar graph

Our Results

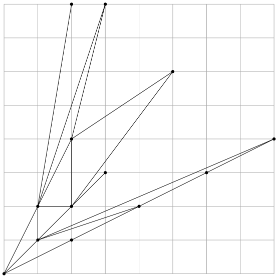
- k-inner planar graphs on $2(k+1)n \times 2(k+1)n$

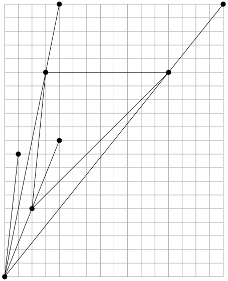
Our Results

- k-inner planar graphs on $2(k+1)n \times 2(k+1)n$
- outerplanar graphs on $n \times n$

Our Results

- k-inner planar graphs on $2(k+1)n \times 2(k+1)n$
- outerplanar graphs on $n \times n$





Intuition Behind the Algorithm

We have a planar graph G and we want a planar monotone drawing of G:

Intuition Behind the Algorithm

We have a planar graph G and we want a planar monotone drawing of G:

- Draw a monotone drawing of a spanning tree T of G. We use Good ST.

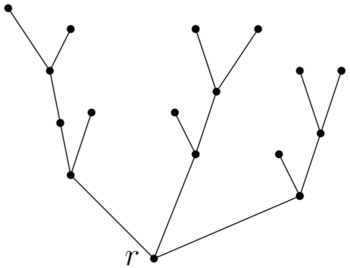
Intuition Behind the Algorithm

We have a planar graph G and we want a planar monotone drawing of G:

- Draw a monotone drawing of a spanning tree T of G. We use Good ST.
- Insert the remaining non-tree edges so that the drawing remain planar.

Good Spanning Trees

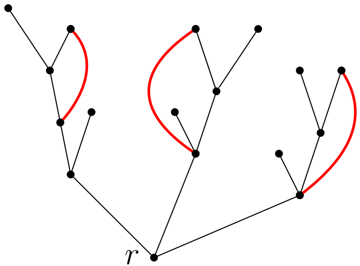
A spanning tree T is called a good spanning tree of a graph G if:



Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

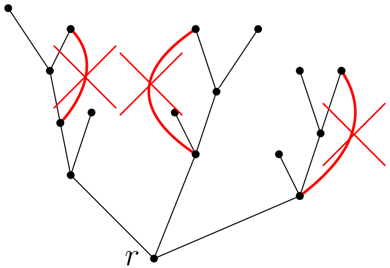
1. There is no non-tree edge (u, v) where both u and v lie in the same path from the root to a leaf.



Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

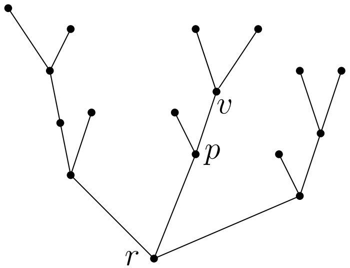
1. There is no non-tree edge (u, v) where both u and v lie in the same path from the root to a leaf.



Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

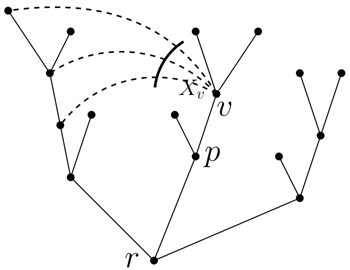
2. The edges incident to a vertex v, excluding the vertex of v to its father p can be divided into three sets X_u, Y_u and Z_u that appear clockwise in this order after edge (p, u) where:



Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

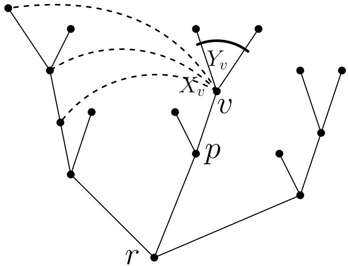
2. The edges incident to a vertex v, excluding the vertex of v to its father p can be divided into three sets X_u, Y_u and Z_u that appear clockwise in this order after edge (p, u) where:



Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

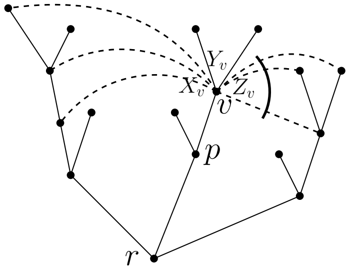
2. The edges incident to a vertex v, excluding the vertex of v to its father p can be divided into three sets X_u, Y_u and Z_u that appear clockwise in this order after edge (p, u) where:



Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

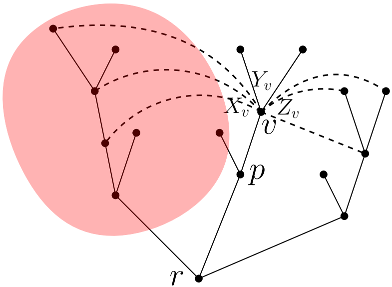
2. The edges incident to a vertex v, excluding the vertex of v to its father p can be divided into three sets X_u, Y_u and Z_u that appear clockwise in this order after edge (p, u) where:



Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

2. The edges incident to a vertex v, excluding the vertex of v to its father p can be divided into three sets X_u, Y_u and Z_u that appear clockwise in this order after edge (p, u) where:
   - A. X_v is a set of non-tree edges that terminate "left" of v.

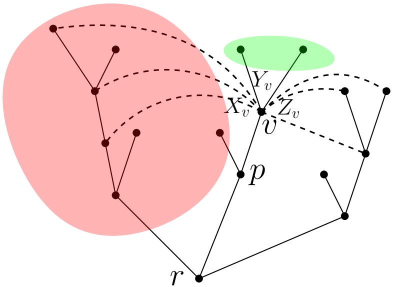


Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

2. The edges incident to a vertex v, excluding the vertex of v to its father p can be divided into three sets X_u, Y_u and Z_u that appear clockwise in this order after edge (p, u) where:

   B. Y_v is a set of tree edges that are children of v.

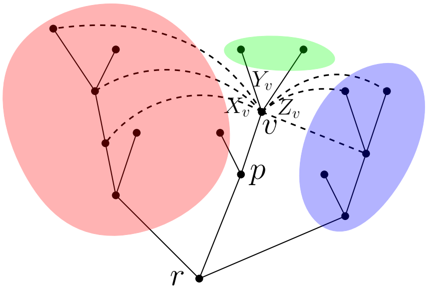


Good Spanning Trees

A spanning tree T is called a good spanning tree of a graph G if:

2. The edges incident to a vertex v, excluding the vertex of v to its father p can be divided into three sets X_u, Y_u and Z_u that appear clockwise in this order after edge (p, u) where:

    c. Z_v is a set of non-tree edges that terminate "right" of v.



Good Spanning Trees

Theorem (Hossain and Rahman)

Let G be a connected planar graph of n vertices. Then G has a planar embedding G_ϕ that contains a good spanning tree.

Good Spanning Trees

Theorem (Hossain and Rahman)

Let G be a connected planar graph of n vertices. Then G has a planar embedding G_ϕ that contains a good spanning tree.

- Good spanning trees can lead to monotone drawings.

Intuition Behind the Algorithm

There are two important things we have to note:

Intuition Behind the Algorithm

There are two important things we have to note:

- If we keep the drawing near-convex monotone, then some non-tree edges are untroublesome.

Intuition Behind the Algorithm

There are two important things we have to note:

- If we keep the drawing near-convex monotone, then some non-tree edges are untroublesome.
- If we insert the non-tree edges in a proper order, the size of the drawing can be kept small.

Modified Tree Drawing Algorithm

Theorem (Oikonomou and Symvonis)

We can produce a monotone drawing of a rooted n-vertex tree T where:

1. *the root r is drawn at* $(0, 0)$*,*

Modified Tree Drawing Algorithm

Theorem (Oikonomou and Symvonis)

We can produce a monotone drawing of a rooted n-vertex tree T where:

1. *the root r is drawn at $(0, 0)$,*
2. *the drawing is near-convex monotone,*

Modified Tree Drawing Algorithm

Theorem (Oikonomou and Symvonis)

We can produce a monotone drawing of a rooted n-vertex tree T where:

1. *the root r is drawn at $(0, 0)$,*
2. *the drawing is near-convex monotone,*
3. *the drawing is contained in the second octant,*

Modified Tree Drawing Algorithm

Theorem (Oikonomou and Symvonis)

We can produce a monotone drawing of a rooted n-vertex tree T where:

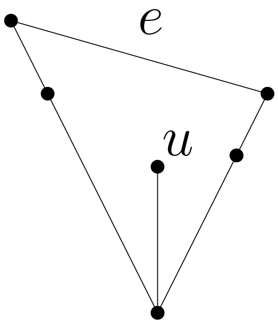
1. *the root r is drawn at $(0, 0)$,*
2. *the drawing is near-convex monotone,*
3. *the drawing is contained in the second octant,*
4. *it fits in a $2n \times 2n$ grid.*

Some Terminology

Consider an embedded plane graph G and a good spanning tree T.

Definition

We say that a non-tree edge e of G *covers* vertex u if u lies in the inner face delimited by the simple cycle formed by tree-edges of T and e.

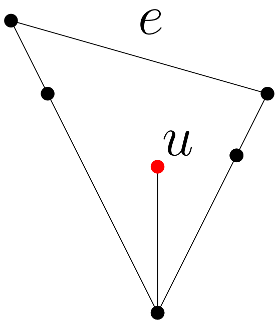


Some Terminology

Consider an embedded plane graph G and a good spanning tree T.

Definition

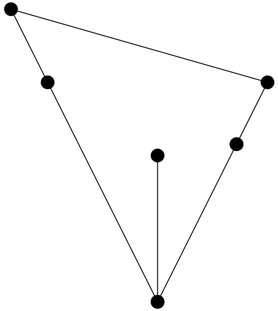
We say that a non-tree edge e of G *covers* vertex u if u lies in the inner face delimited by the simple cycle formed by tree-edges of T and e.



Some Terminology

Definition

A non-tree edge e is called a *leader edge* if:

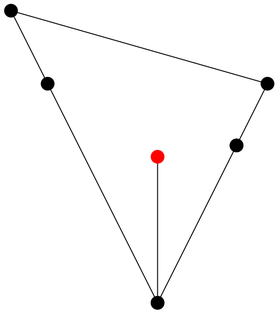


Some Terminology

Definition

A non-tree edge e is called a *leader edge* if:

- e covers at least one vertex

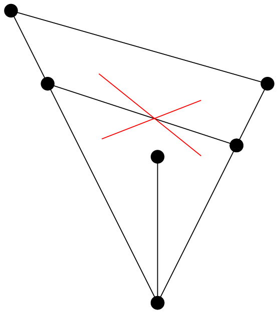


Some Terminology

Definition

A non-tree edge e is called a *leader edge* if:

- e covers at least one vertex
- There does not exist another non-tree edge e' that:
  - Covers the same set of vertices as e
  - e' is inside the simple cycle induced by T and e



Some Terminology

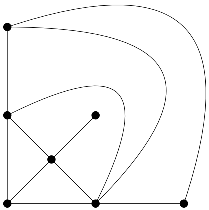
Lemma

Given a k-inner planar graph G, we can get an embedding G_ϕ with a good spanning tree T where there exist at most k leader edges.

- Follows from the construction of GST given by Hossain and Rahman.

Planarity of the Drawing

Consider an k-inner planar graph G with a good spanning tree T.

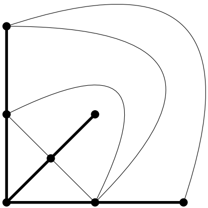


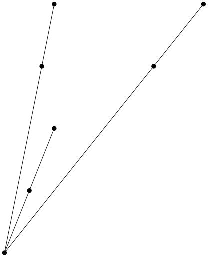
Planarity of the Drawing

Consider an k-inner planar graph G with a good spanning tree T.

Lemma

Let Γ_T a non-strictly slope-disjoint and near-convex monotone drawing of T. We consider two drawings:





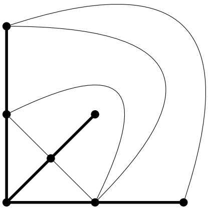
Planarity of the Drawing

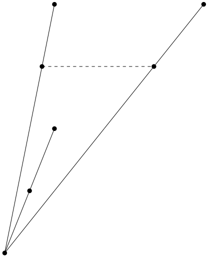
Consider an k-inner planar graph G with a good spanning tree T.

Lemma

Let Γ_T a non-strictly slope-disjoint and near-convex monotone drawing of T. We consider two drawings:

- Γ_L *the drawing produced if we add all leader edges to T to Γ_T*





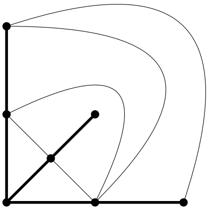
Planarity of the Drawing

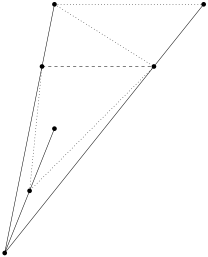
Consider an k-inner planar graph G with a good spanning tree T.

Lemma

Let Γ_T a non-strictly slope-disjoint and near-convex monotone drawing of T. We consider two drawings:

- *Γ_L the drawing produced if we add all leader edges to T to Γ_T*
- *Γ the drawing produced if we add all non-tree edges to Γ_T.*





Planarity of the Drawing

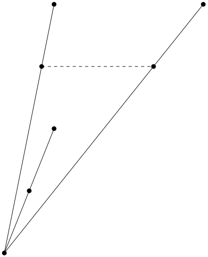
Consider an k-inner planar graph G with a good spanning tree T.

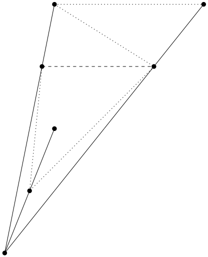
Lemma

Let Γ_T a non-strictly slope-disjoint and near-convex monotone drawing of T. We consider two drawings:

- *Γ_L the drawing produced if we add all leader edges to T to Γ_T*
- *Γ the drawing produced if we add all non-tree edges to Γ_T.*

Γ_L is planar iff Γ is planar.





Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.

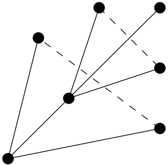
Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

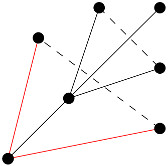
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

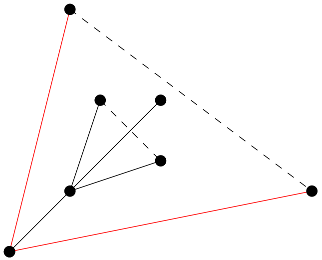
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

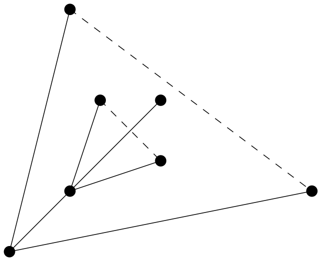
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

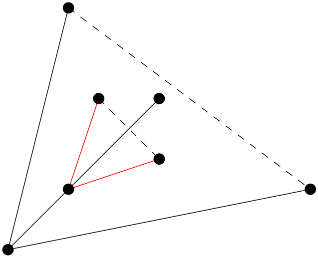
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

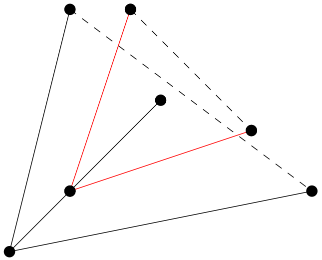
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

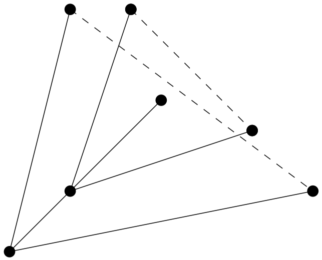
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

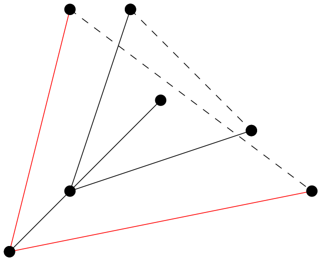
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

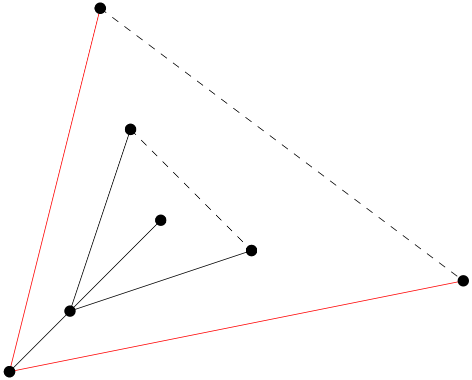
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

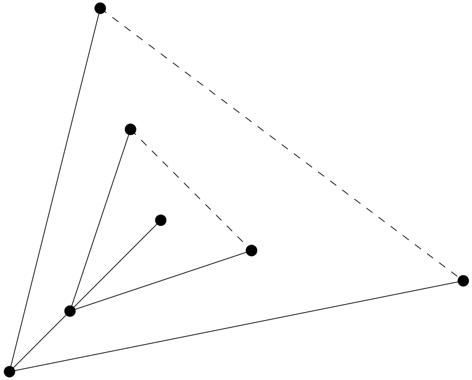
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

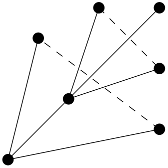
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

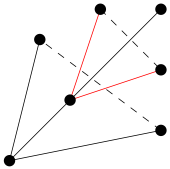
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

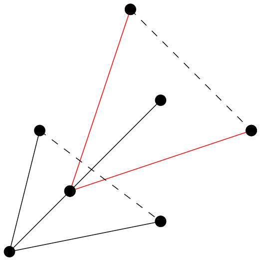
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

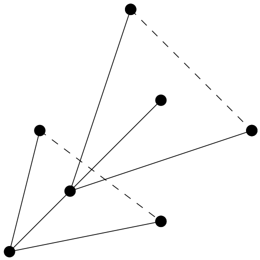
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

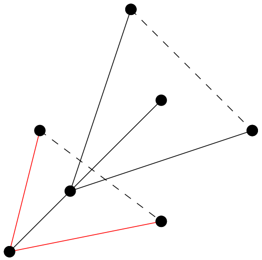
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

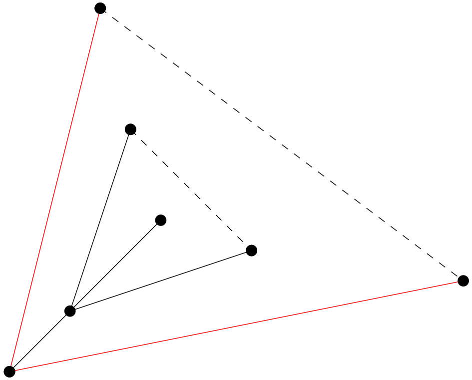
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

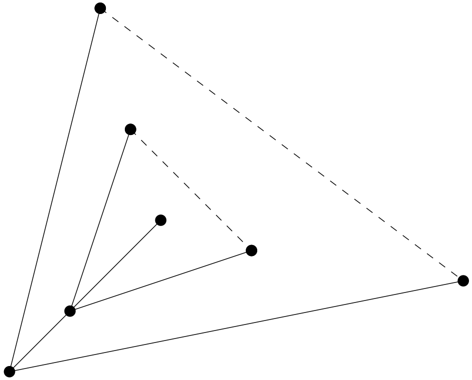
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

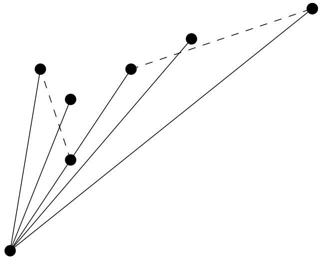
Case I: Nested Leader Edges



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

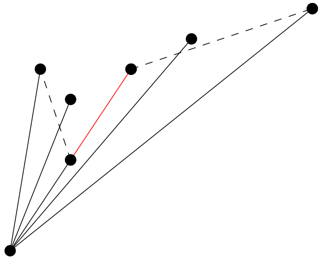
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

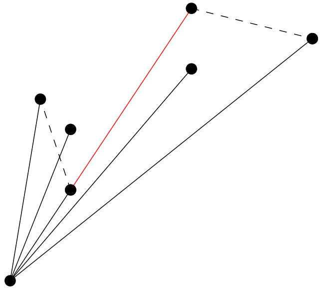
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

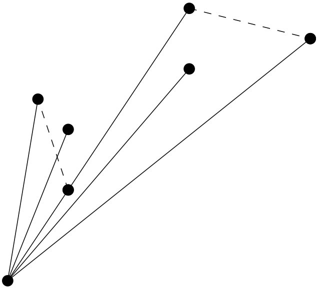
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

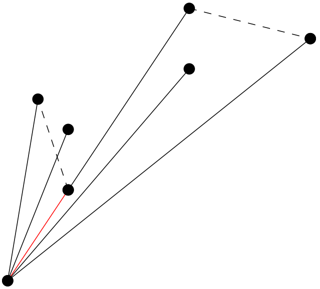
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

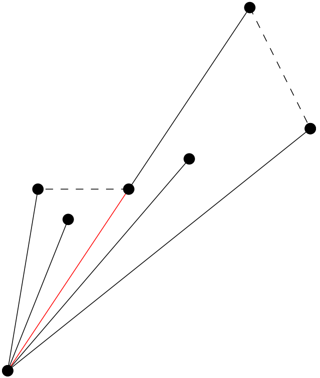
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

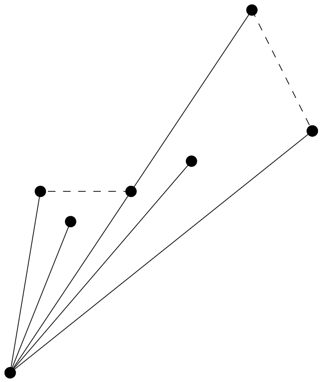
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

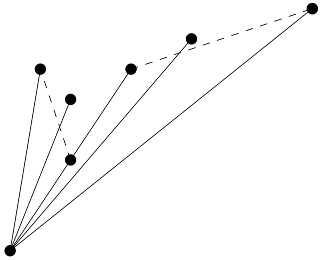
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

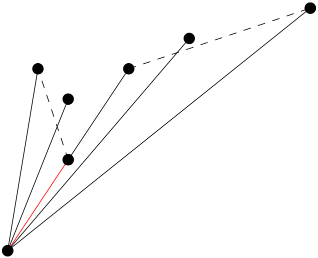
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

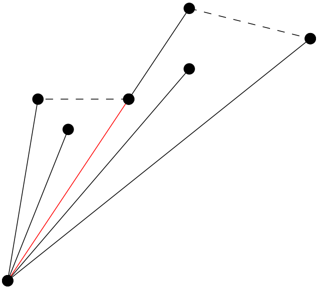
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

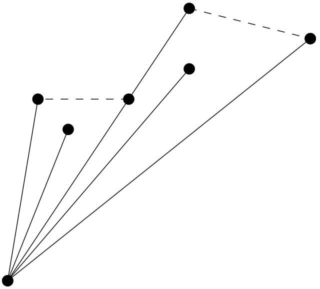
Case II: Ancestor



Examination of Leader Edges

- We wish to deal with each distinct leader edge only once.
- Two problematic cases.

Case II: Ancestor



Examination Order of Leader Edges

Lemma

There exists an ordering of the leader edges, such that if they are inserted into the drawings in that order they need to be examined exactly once.

How:

Examination Order of Leader Edges

Lemma

There exists an ordering of the leader edges, such that if they are inserted into the drawings in that order they need to be examined exactly once.

How:

- We create a DAG with the leader edges as nodes

Examination Order of Leader Edges

Lemma

There exists an ordering of the leader edges, such that if they are inserted into the drawings in that order they need to be examined exactly once.

How:

- We create a DAG with the leader edges as nodes
- Dependencies between them are the directed edges

Examination Order of Leader Edges

Lemma

There exists an ordering of the leader edges, such that if they are inserted into the drawings in that order they need to be examined exactly once.

How:

- We create a DAG with the leader edges as nodes
- Dependencies between them are the directed edges
- In each step we visit a leader edge with no dependencies.

Monotone Graph Drawing Algorithms

Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

Monotone Graph Drawing Algorithms

Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

- Calculate an embedding G_ϕ that contains a good spanning tree T

Monotone Graph Drawing Algorithms

Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

- Calculate an embedding G_ϕ that contains a good spanning tree T
- Draw T according to the modified Monotone Tree Drawing Algorithm

Monotone Graph Drawing Algorithms

Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

- Calculate an embedding G_ϕ that contains a good spanning tree T
- Draw T according to the modified Monotone Tree Drawing Algorithm
- Calculate the dependencies DAG G_L between leader edges of G_ϕ

Monotone Graph Drawing Algorithms

Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

- Calculate an embedding G_ϕ that contains a good spanning tree T
- Draw T according to the modified Monotone Tree Drawing Algorithm
- Calculate the dependencies DAG G_L between leader edges of G_ϕ
- **While** G_L is not empty **do**

Monotone Graph Drawing Algorithms

Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

- Calculate an embedding G_ϕ that contains a good spanning tree T
- Draw T according to the modified Monotone Tree Drawing Algorithm
- Calculate the dependencies DAG G_L between leader edges of G_ϕ
- **While** G_L is not empty **do**
  - Find a leader edge e in G_L with no dependencies

Monotone Graph Drawing Algorithms

Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

- Calculate an embedding G_ϕ that contains a good spanning tree T
- Draw T according to the modified Monotone Tree Drawing Algorithm
- Calculate the dependencies DAG G_L between leader edges of G_ϕ
- **While** G_L is not empty **do**
  - Find a leader edge e in G_L with no dependencies
  - Elongate the appropriate edges so that the drawing is planar after inserting e

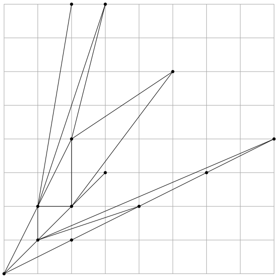
Monotone Graph Drawing Algorithms

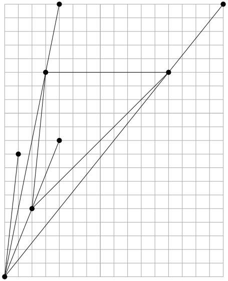
Input: A k-Inner Planar Graph G
Output: A planar monotone drawing of G on $2(k+1)n \times 2(k+1)n$

- Calculate an embedding G_ϕ that contains a good spanning tree T
- Draw T according to the modified Monotone Tree Drawing Algorithm
- Calculate the dependencies DAG G_L between leader edges of G_ϕ
- **While** G_L is not empty **do**
  - Find a leader edge e in G_L with no dependencies
  - Elongate the appropriate edges so that the drawing is planar after inserting e
  - Remove e from G_L

Our Results

- k-inner planar graphs on $2(k+1)n \times 2(k+1)n$
- outerplanar graphs on $n \times n$





Conclusion

- We introduced the class of k-Inner Planar Graphs and provided an algorithm that incorporate k into its output quality.

Conclusion

- We introduced the class of k-Inner Planar Graphs and provided an algorithm that incorporate k into its output quality.
- Can we apply to other graph Drawing Problems?

Conclusion

- We introduced the class of k-Inner Planar Graphs and provided an algorithm that incorporate k into its output quality.
- Can we apply to other graph Drawing Problems?

Thank You for Your Attention