Straightening Drawings of Clustered Hierarchical Graphs

Sergey Bereg[1], **Markus Völker**[2], Alexander Wolff[3], Yuanyi Zhang[1]

[1] Department of Computer Science, University of Texas at Dallas, U.S.A.

[2] Fakultät für Informatik, Universität Karlsruhe, Germany

[3] Faculteit Wiskunde en Informatica, Technische Universiteit Eindhoven, the Netherlands

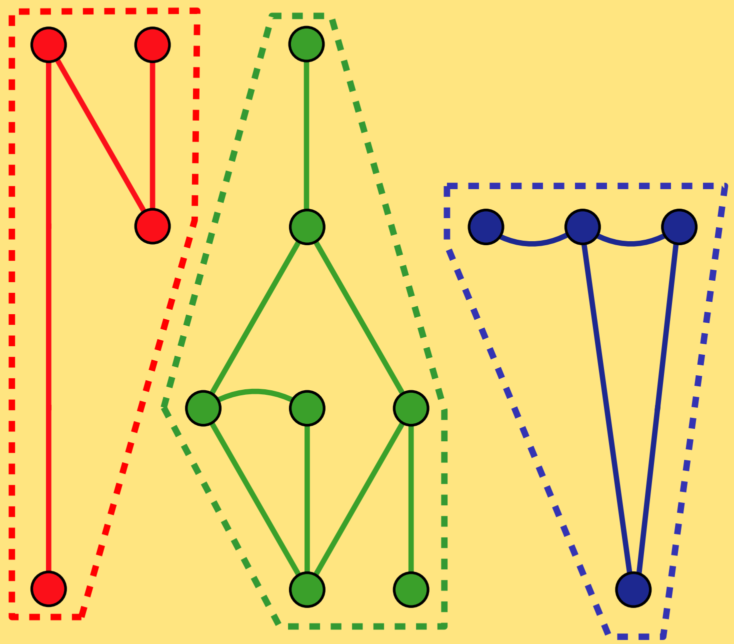
Introduction

Clustered Graphs

Definition

A *clustered graph* $\mathcal{C} = (G, T)$ consists of

- an undirected graph $G = (V, E)$
- a partition of the vertex set V into clusters



Structural Information

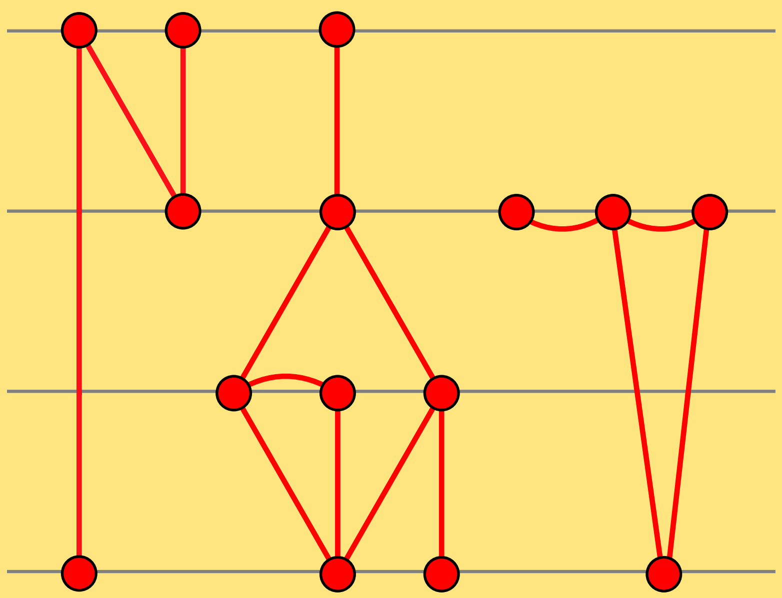
- vertices in the same cluster are interpreted as being similar
- vertices in different clusters are interpreted as being different

Hierarchical Graphs

Definition

A *hierarchical* graph $\mathcal{L} = (G, \lambda)$ is given by

- an undirected graph $G = (V, E)$
- an assignment $\lambda : V \to \{1, \ldots, k\}$ of the vertices to horizontal layers

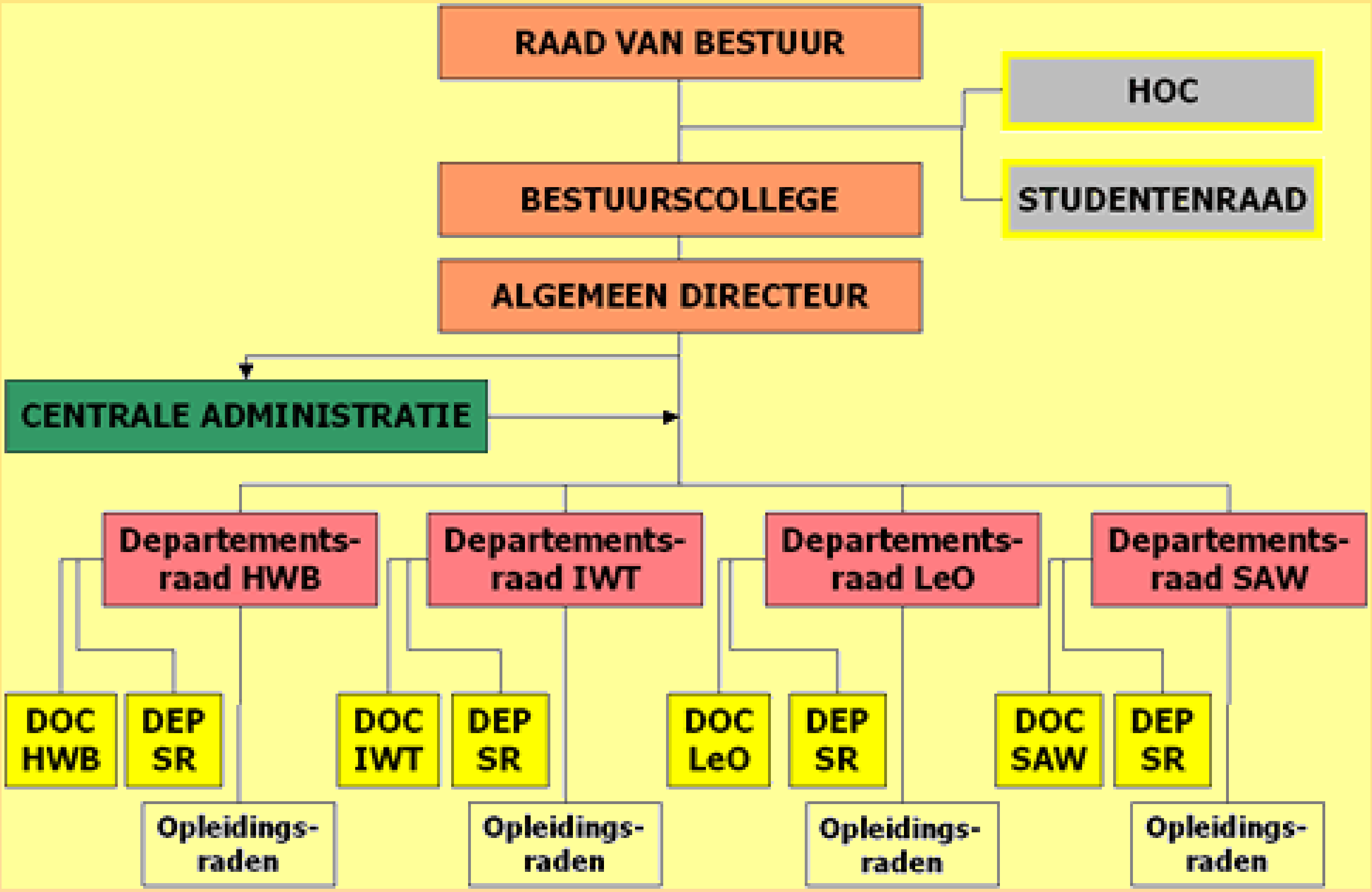


Structural Information

- the vertex set V is partitioned by the rank of the vertices
- the rank of a vertex reflects its importance in relation to vertices of lower or higher rank

Hierarchical Graphs

Example - Organigrams



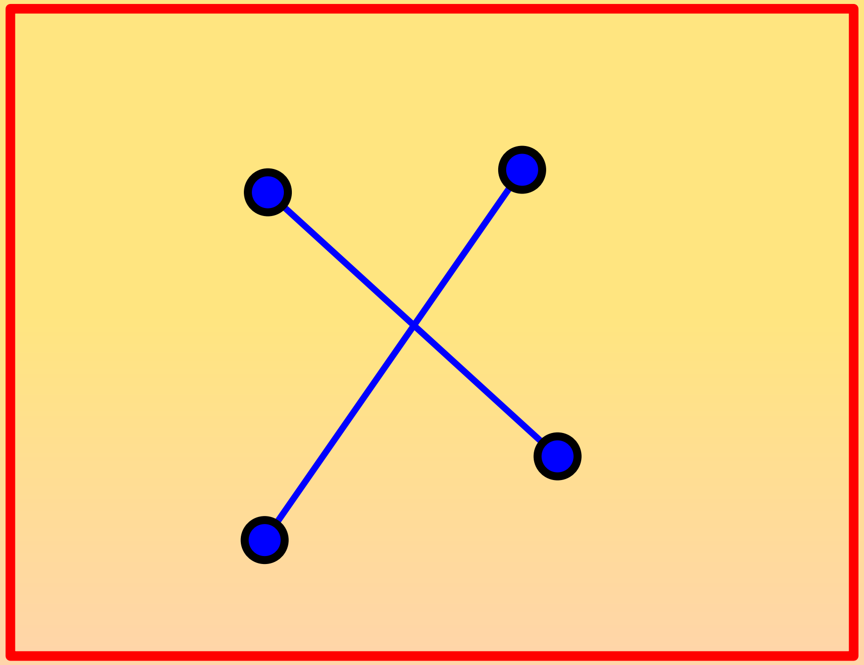
organigram of Hogeschool Limburg

Compound Planar Graphs

Definition

A graph is *compound planar (c-planar)*, if it admits a drawing

- without edge-crossings
- without edge-region-crossings

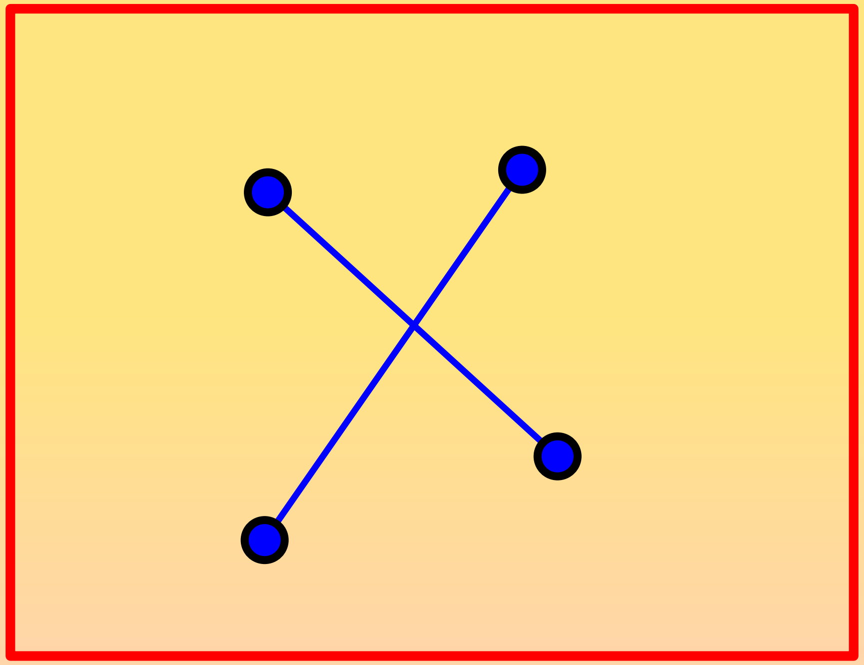


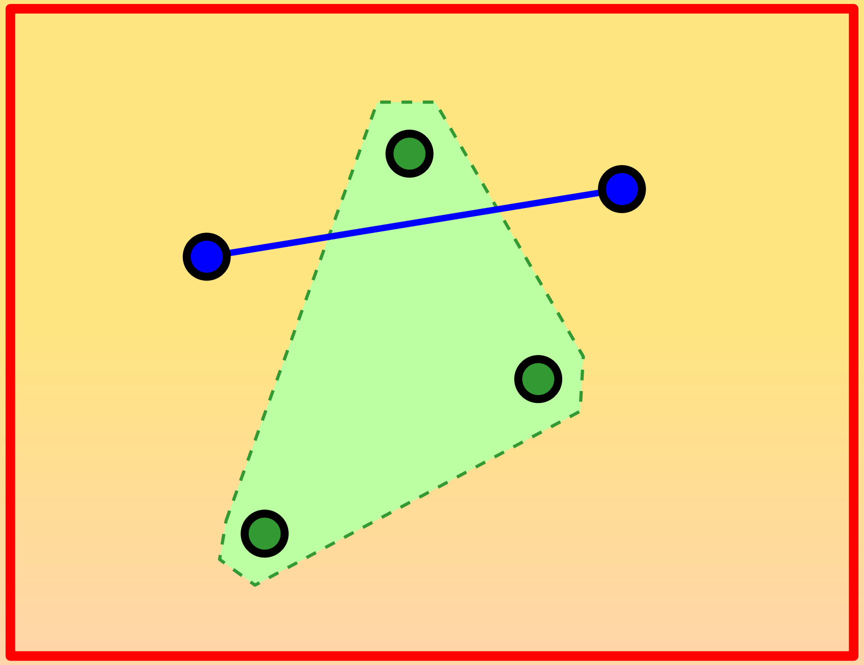
Compound Planar Graphs

Definition

A graph is *compound planar (c-planar)*, if it admits a drawing

- without edge-crossings
- without edge-region-crossings *(region = convex hull of a cluster)*

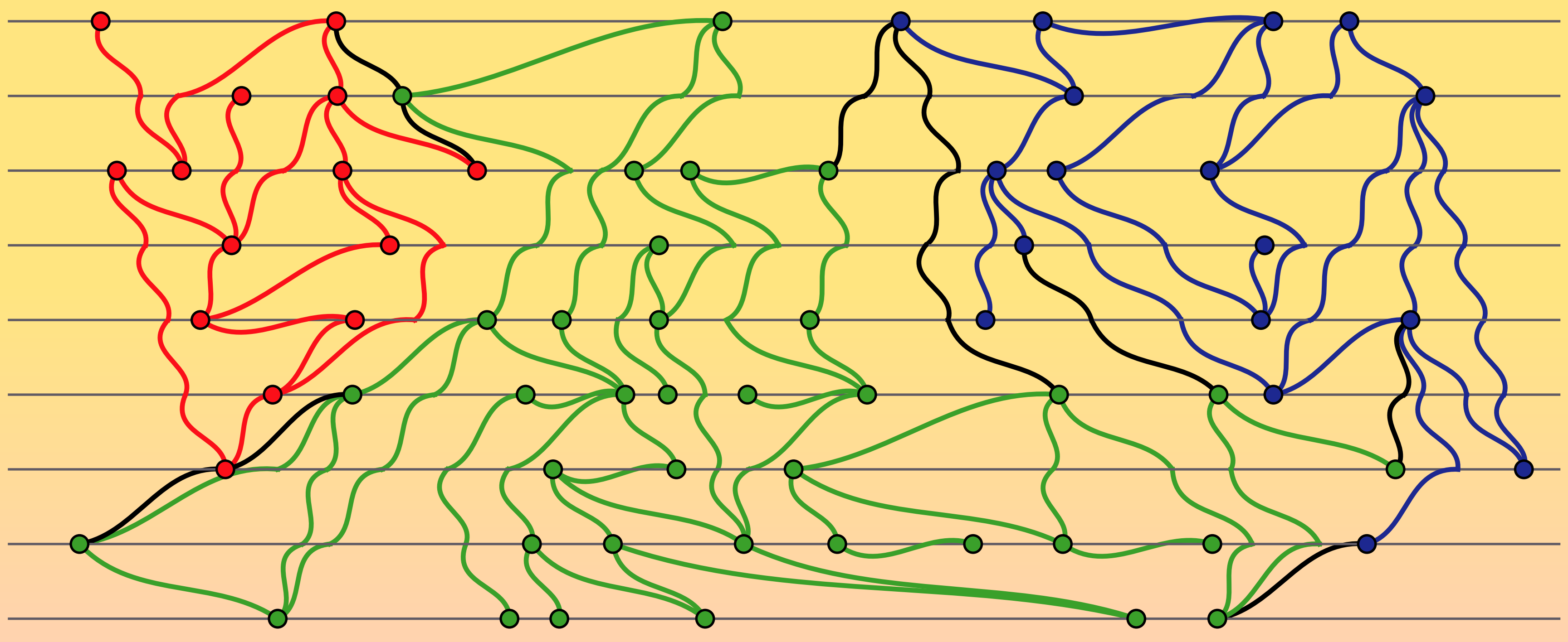




Problem Definition

Input

- embedded c-planar graph $G(V, E)$
- disjoint clusters $C_1 \cup \ldots \cup C_m = V$
- layers $\lambda : V \rightarrow \{1, 2, \ldots, k\}$

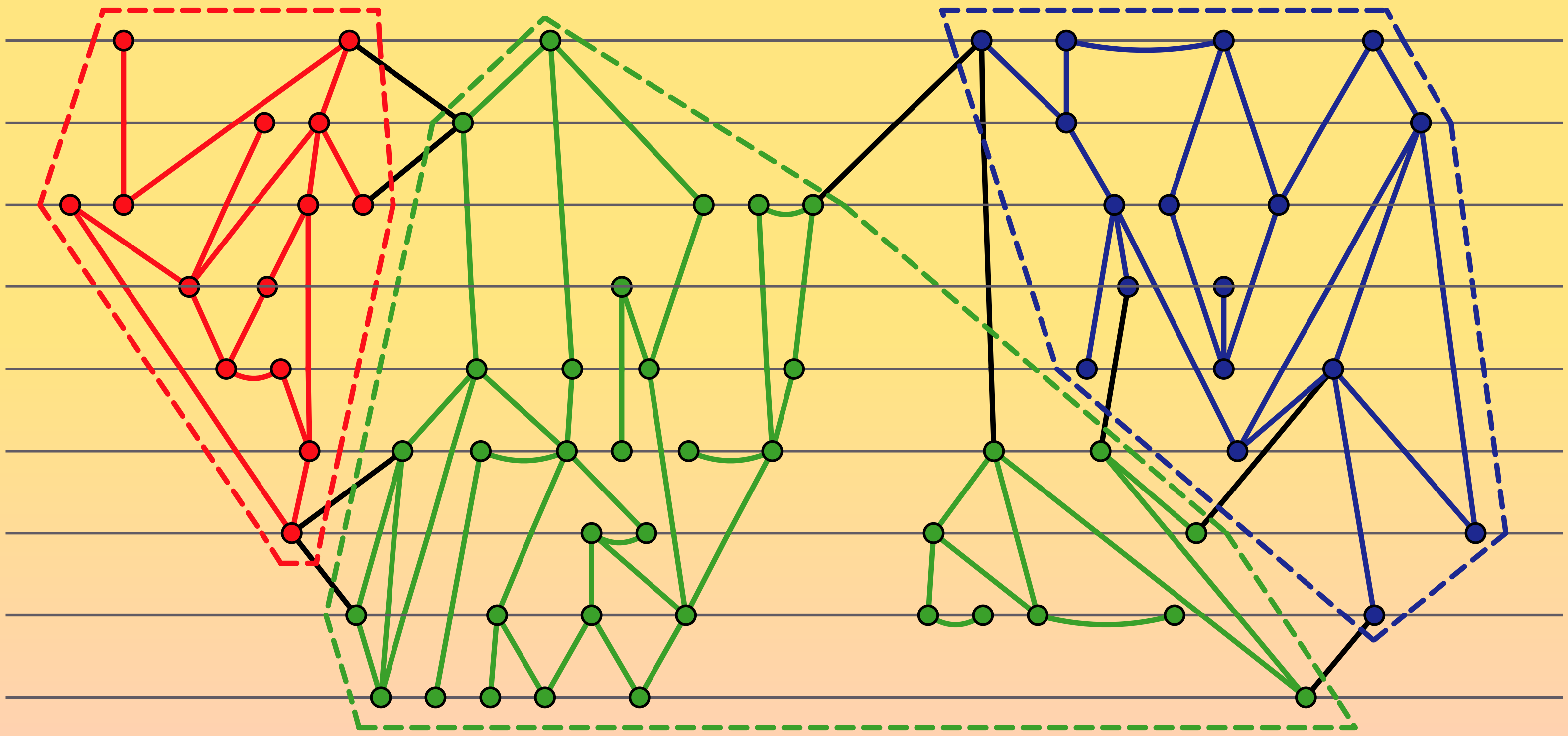


Problem Definition

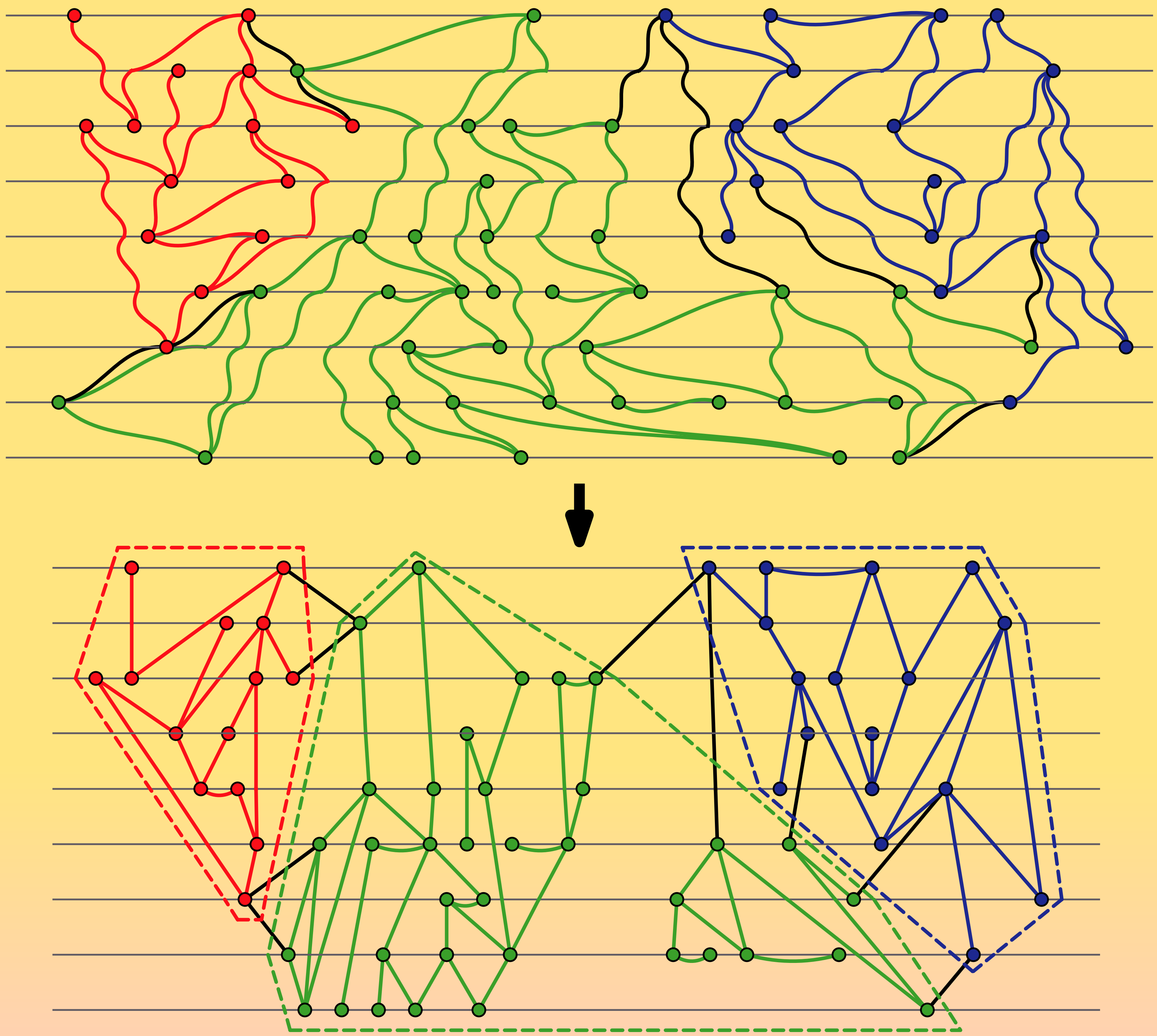
Output

Drawing of G such that

- edges are straight-line segments,
- clusters lie in disjoint convex regions,
- no edge intersects a cluster boundary twice.



Problem Definition



Related Work

Eades, Feng, Lin, Nagamochi (2005)

- *input:* compound planar graph G
- *output:* drawing of G with
  - straight edges
  - convex cluster regions
- time complexity: $O(n)$
- disadvantage: places each vertex at a unique layer
  $\Rightarrow k \times k$ square grid will be drawn on k^2 layers

For further references to related work please refer to our paper.

Overview of our work

Our aim: Producing vertical compact drawings

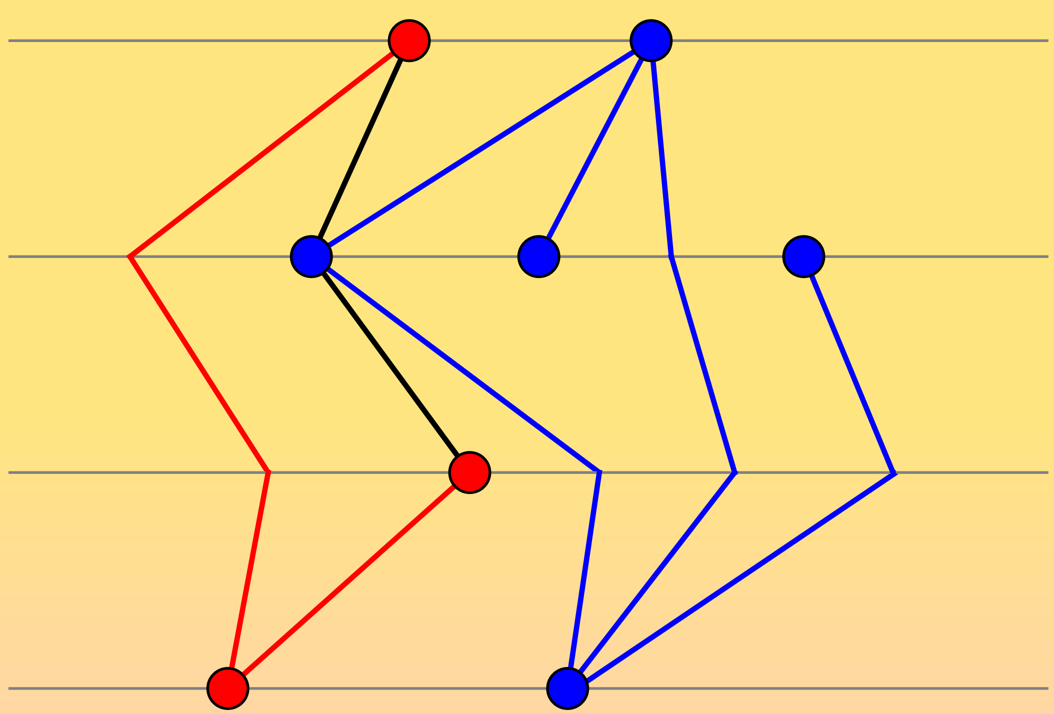
- Two fast algorithms
  - run in $O(n^2)$ and $O(n)$ time, resp.,
  - have certain preconditions.
- LP formulation
  - *always* finds a drawing if one exists,
  - produces nicer results due to global optimization,
  - slower.

LP Formulation

LP formulation: *variables*

We add one variable to our LP formulation for the x-coordinate of each

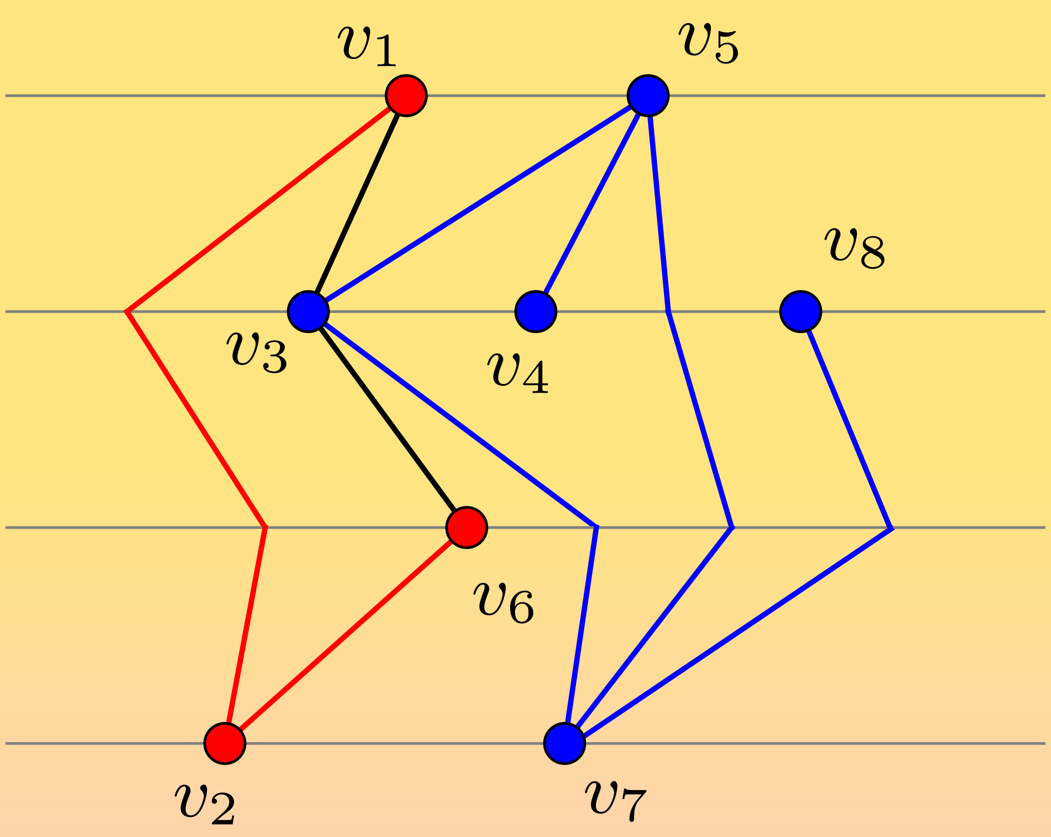
- vertex
- edge-level-crossing



LP formulation: *variables*

We add one variable to our LP formulation for the x-coordinate of each

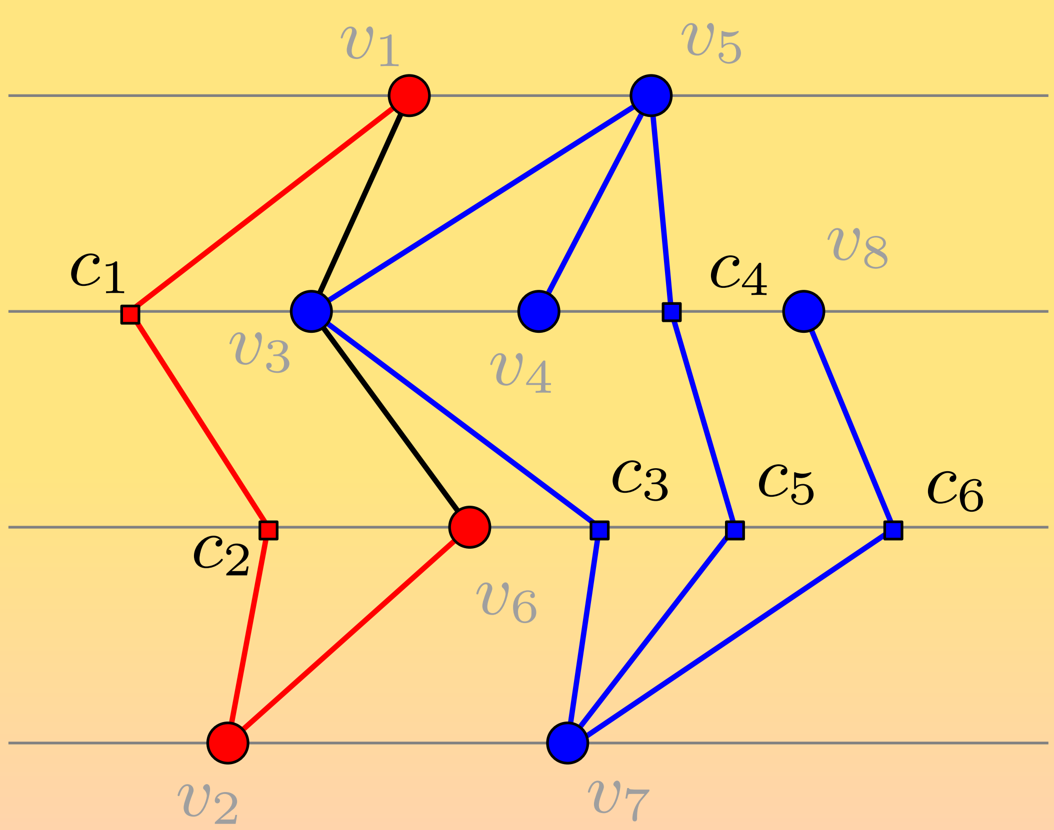
- vertex
- edge-level-crossing



LP formulation: *variables*

We add one variable to our LP formulation for the x-coordinate of each

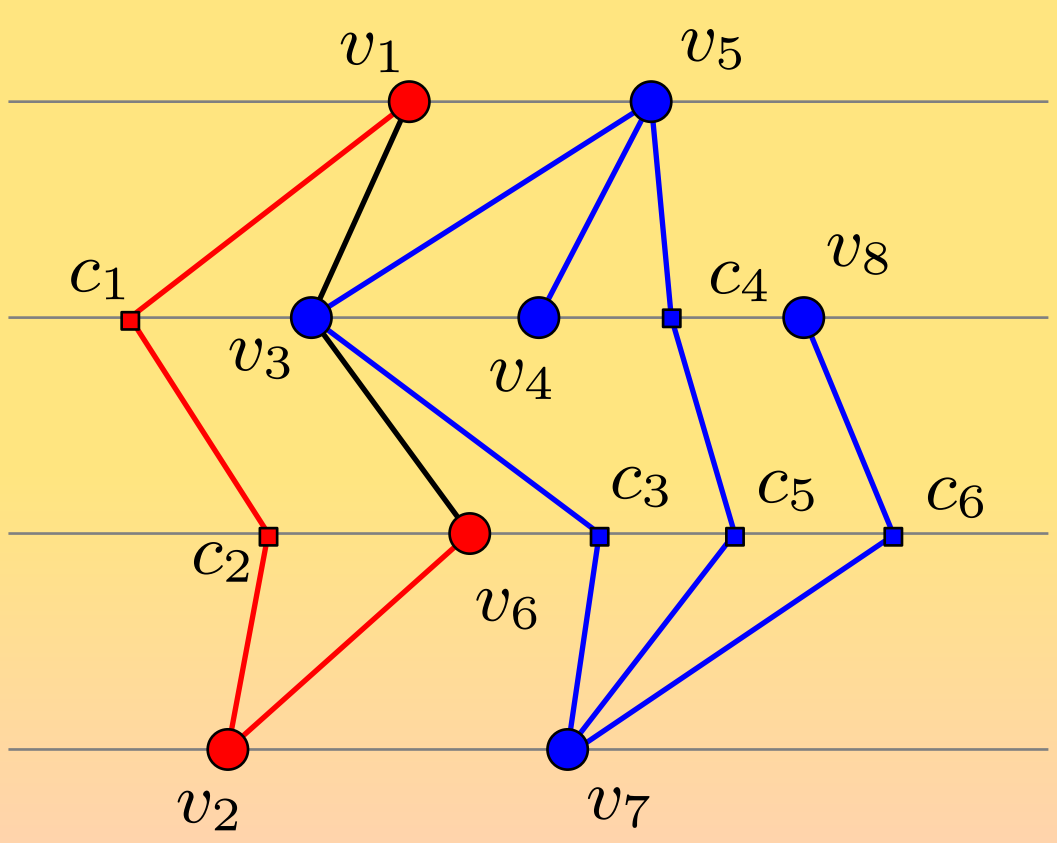
- vertex
- edge-level-crossing



LP formulation: *variables*

We add one variable to our LP formulation for the x-coordinate of each

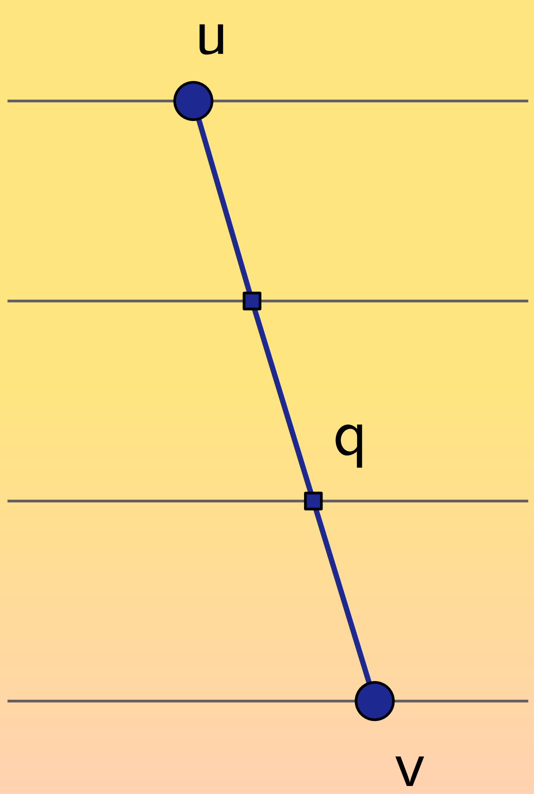
- vertex $\Rightarrow$ $O(n)$ variables
- edge-level-crossing $\Rightarrow$ $O(n)$ variables



LP formulation: *constraints*

We want ...

- straight line edges $\Rightarrow O(n)$ constraints
- preservation of the original embedding
- minimum distances between vertices and edges
- disjoint convex hulls



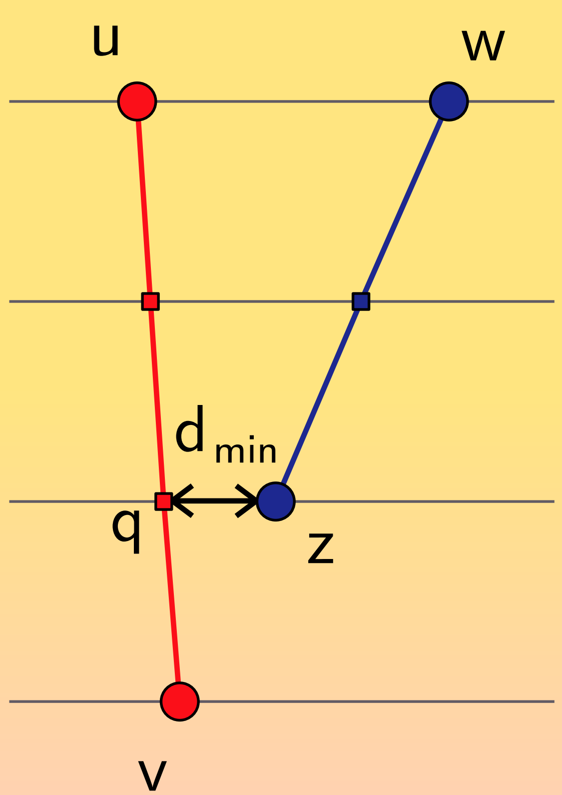
For each edge $(u, v) \in E$ and each crossing q of (u, v) with a layer add constraint:

$$\boxed{\mathrm{RelPos}(q,u,v) = \begin{vmatrix} q_x & \lambda(q) & 1 \\ u_x & \lambda(u) & 1 \\ v_x & \lambda(v) & 1 \end{vmatrix} \stackrel{!}{=} 0}$$

LP formulation: *constraints*

We want . . .

- straight line edges $\Rightarrow O(n)$ constraints
- preservation of the original embedding $\Rightarrow O(n)$ constraints
- minimum distances between vertices and edges $\Rightarrow O(n)$ constraints
- disjoint convex hulls



For each vertex w to the right of a vertex u add constraint:

$$\boxed{u_x + d_{min} \leq w_x}$$

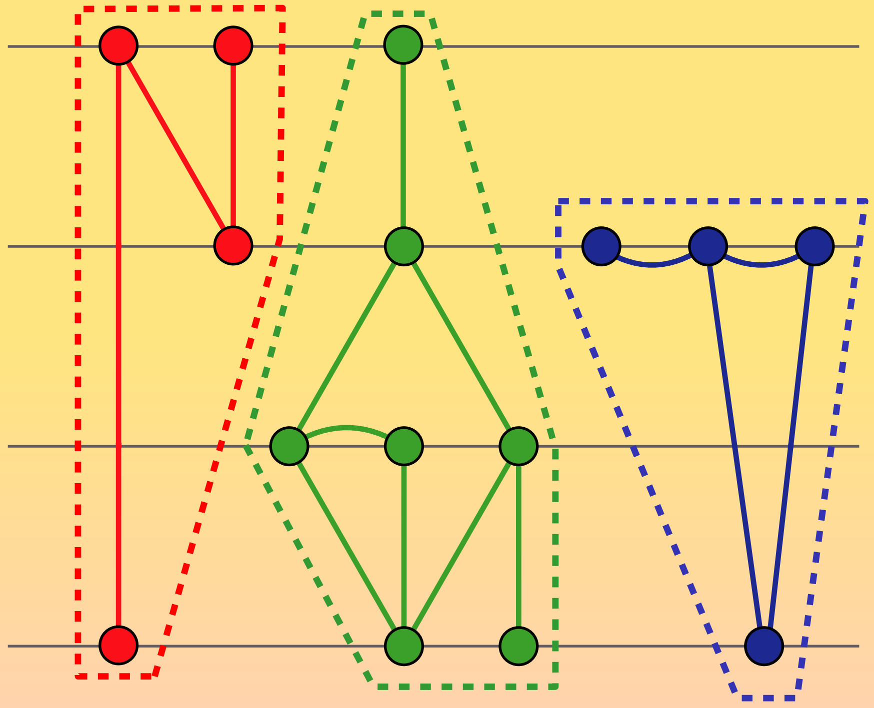
For each vertex z to the right of an edge-layer crossing q add constraint:

$$\boxed{q_x + d_{min} \leq z_x}$$

LP formulation: *constraints*

We want . . .

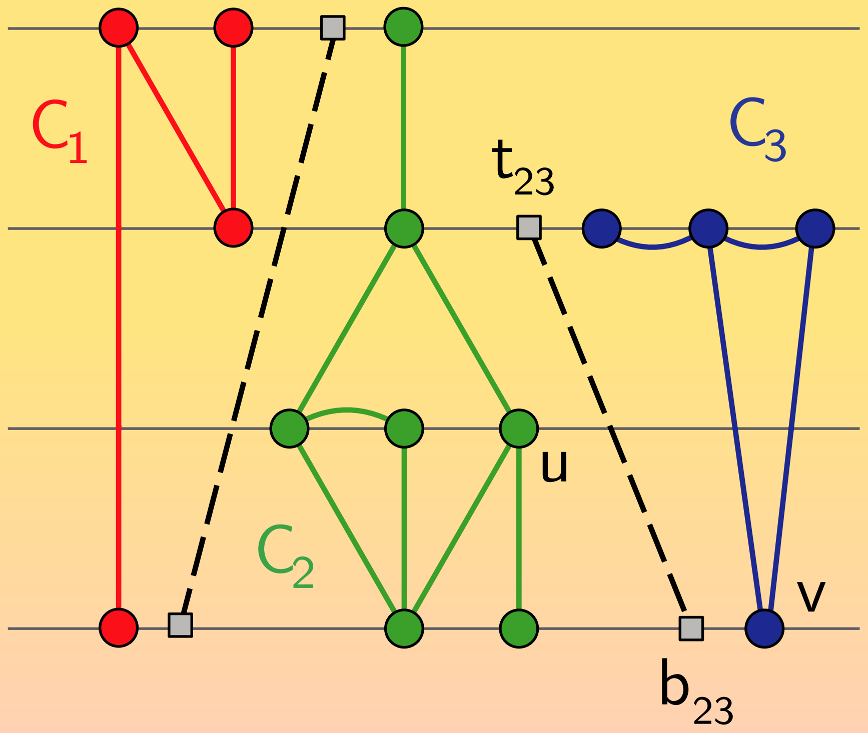
- straight line edges $\Rightarrow O(n)$ constraints
- preservation of the original embedding $\Rightarrow O(n)$ constraints
- minimum distances between vertices and edges $\Rightarrow O(n)$ constraints
- disjoint convex hulls



LP formulation: *constraints*

We want ...

- straight line edges $\Rightarrow O(n)$ constraints
- preservation of the original embedding $\Rightarrow O(n)$ constraints
- minimum distances between vertices and edges $\Rightarrow O(n)$ constraints
- disjoint convex hulls $\Rightarrow O(n)$ constraints



- add separating line between adjoining pairs of clusters
- maintain position in relation to the separating line

$$\mathrm{RelPos}(u, b_{23}, t_{23}) > 0$$
$$\mathrm{RelPos}(v, b_{23}, t_{23}) < 0$$

LP formulation: *constraints*

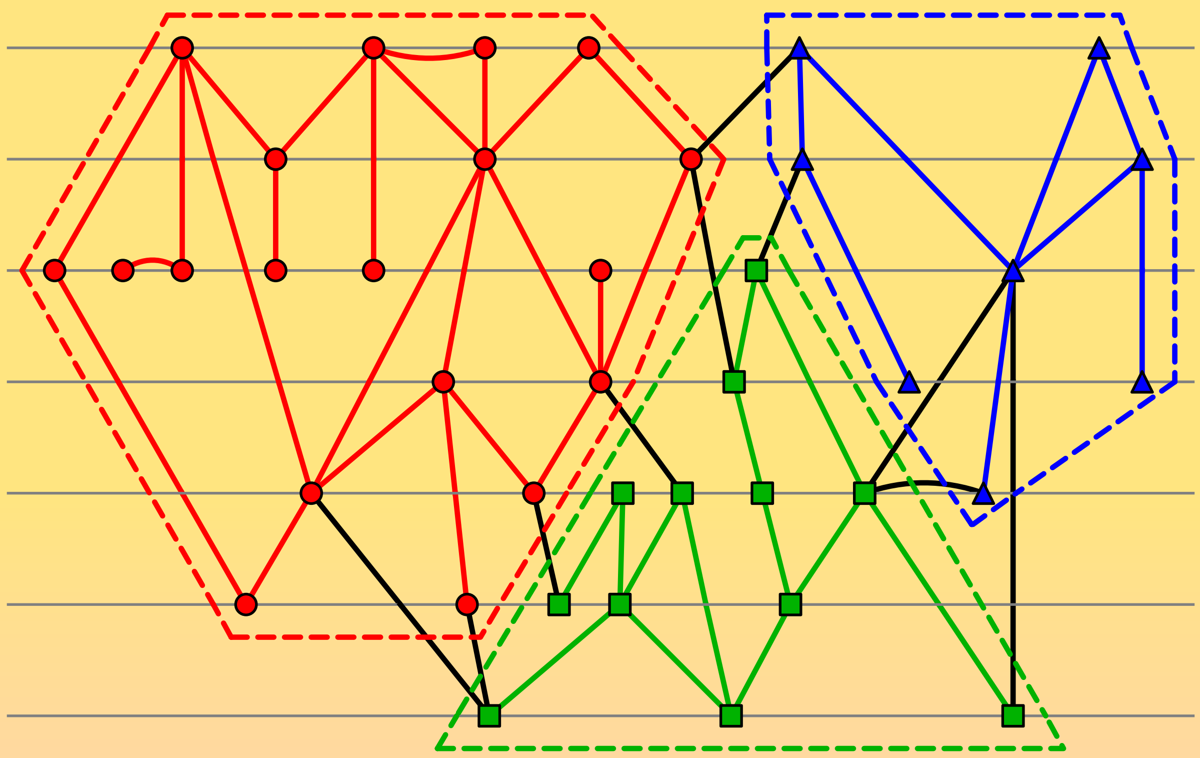
We want . . .

- straight line edges $\Rightarrow O(n)$ constraints
- preservation of the original embedding $\Rightarrow O(n)$ constraints
- minimum distances between vertices and edges $\Rightarrow O(n)$ constraints
- disjoint convex hulls $\Rightarrow O(n)$ constraints

Lemma. Our LP uses $O(n)$ variables and $O(n)$ constraints.

LP formulation: *objective function*

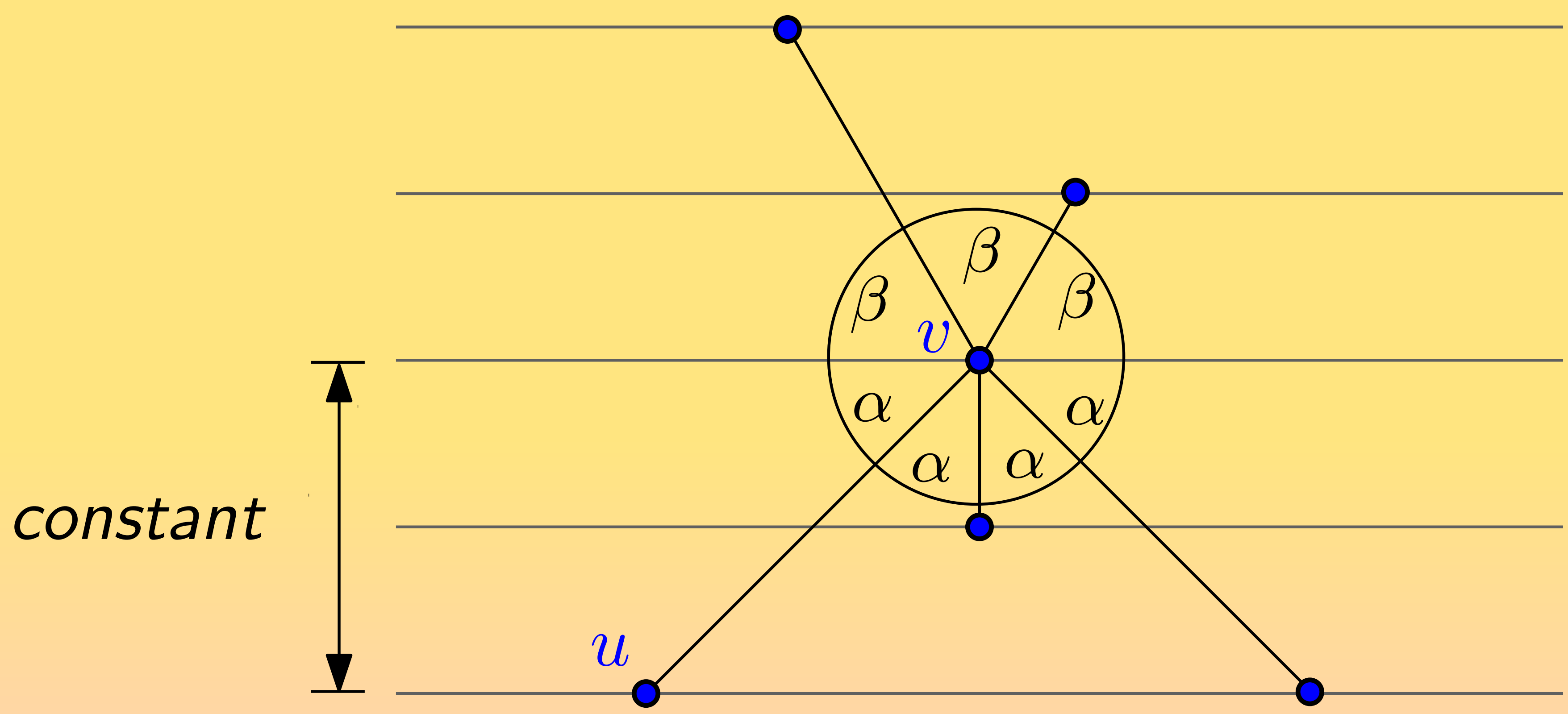
- many optimization criteria possible *(angles, width, . . .)*
- optimization for a good angular resolution works very well
- question: *How to optimize angles using* **linear** *constraints?*



optimize for *"nice"* angles

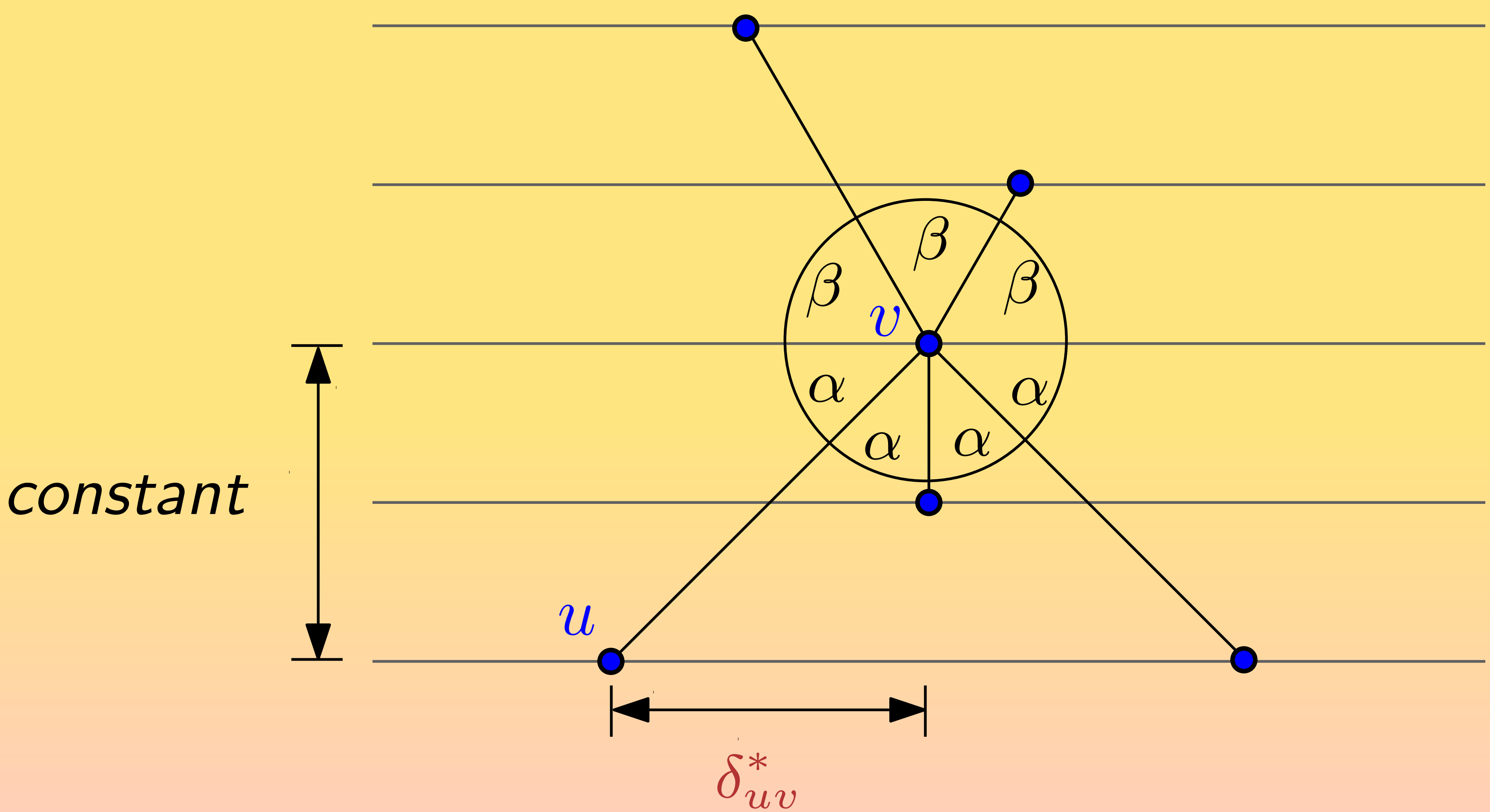
LP formulation: *objective function*

- uniformly distribute the 180° angular space above and below each vertex
- for each vertex the optimal relative positions of all adjacent vertices can be precomputed using trigonometric functions



LP formulation: *objective function*

- now we can compute an optimal x-offset δ^*_{uv} between u and v
- the actual offset δ_{uv} is given by $x_u - x_v$
- the absolute difference μ_{uv} of δ_{uv} and δ^*_{uv} can expressed as follows:



LP formulation: *objective function*

- now we can compute an optimal x-offset δ^*_{uv} between u and v
- the actual offset δ_{uv} is given by $x_u - x_v$
- the absolute difference μ_{uv} of δ_{uv} and δ^*_{uv} can expressed as follows:

$$\boxed{\begin{aligned} \mu_{uv} &\geq +\delta^*_{uv} - \delta_{uv} \\ \mu_{uv} &\geq -\delta^*_{uv} + \delta_{uv} \end{aligned}}$$

- our objective function minimizes these deviations μ_{uv} from the optimum

$$\boxed{\text{minimize} \sum_{\{u,v\} \in E} (\mu_{uv} + \mu_{vu})}$$

Recursive Algorithm

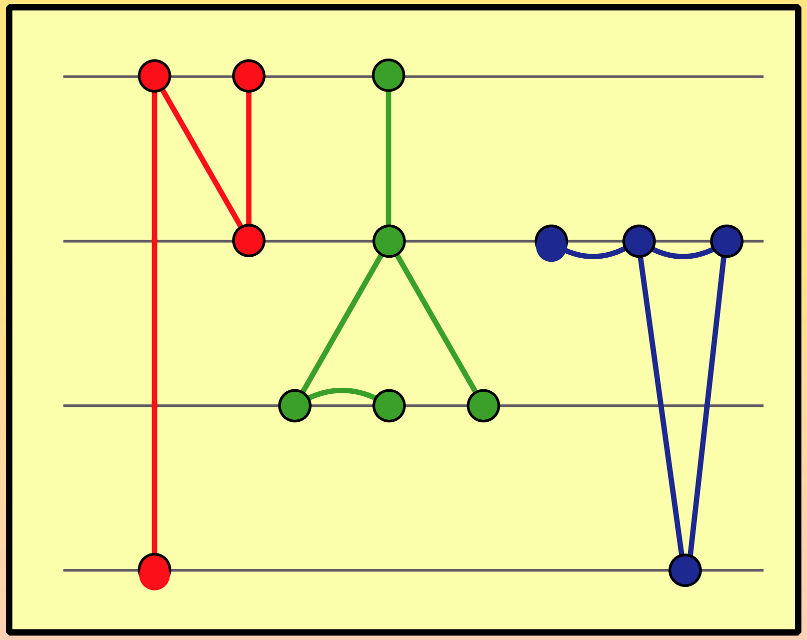
Recursive Algorithm: *Precondition*

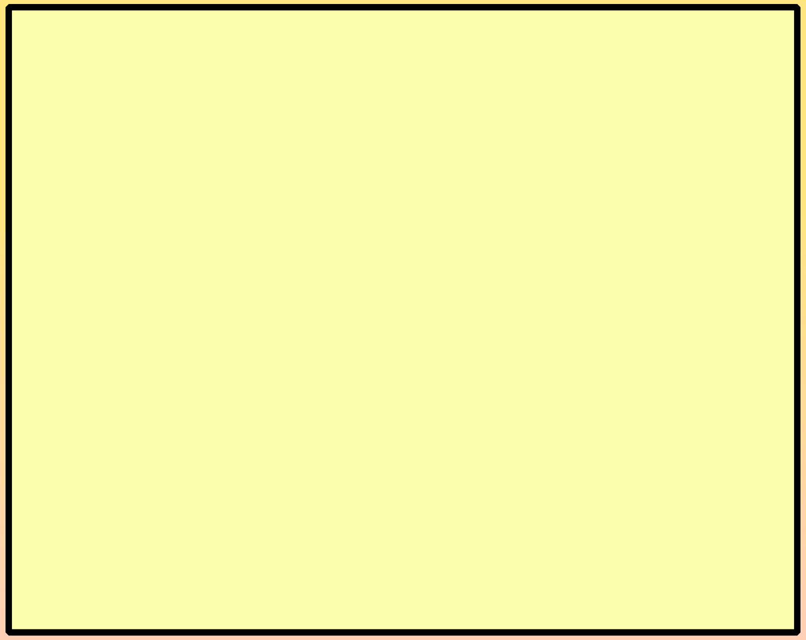
Let $G = (V, E)$ be the graph that we want to draw.

Define the *cluster adjacency graph* F as the directed graph...

- whose vertices correspond to clusters in G
- that has a directed edge between the cluster vertices C and C' if there is a level i on which a vertex of C or an edge connected to a vertex of C lies to the left of a vertex or edge of C'

G





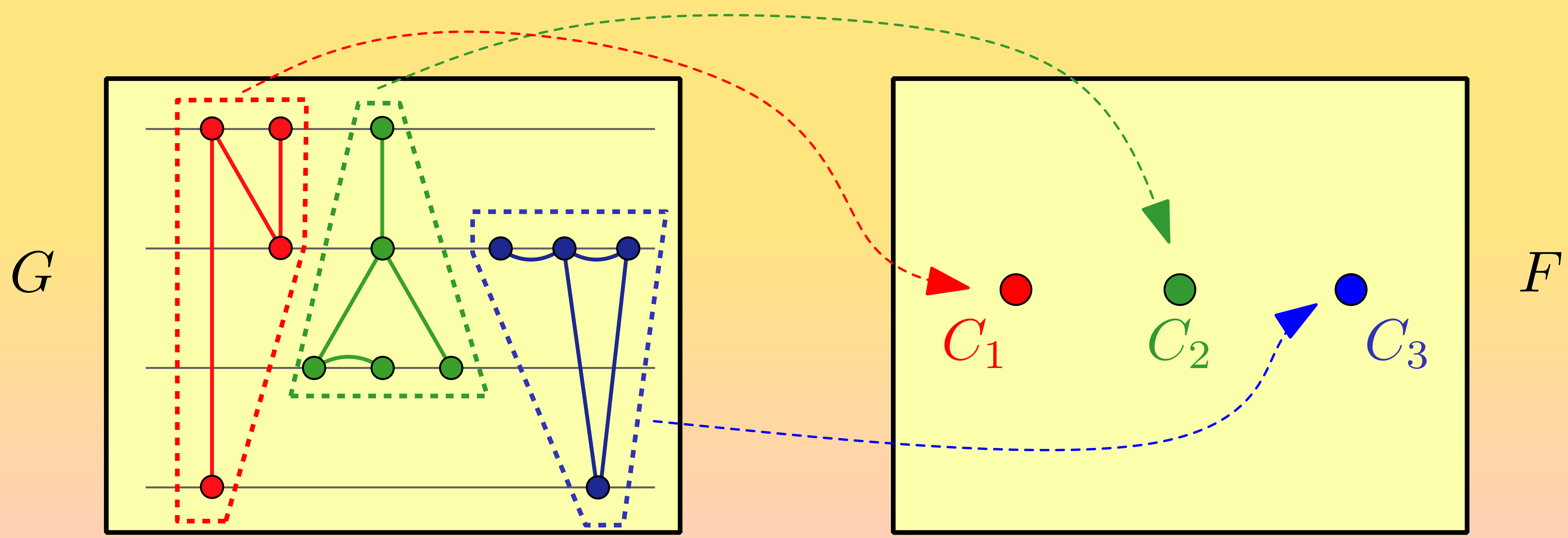
F

Recursive Algorithm: *Precondition*

Let $G = (V, E)$ be the graph that we want to draw.

Define the *cluster adjacency graph* F as the directed graph...

- whose vertices correspond to clusters in G
- that has a directed edge between the cluster vertices C and C' if there is a level i on which a vertex of C or an edge connected to a vertex of C lies to the left of a vertex or edge of C'

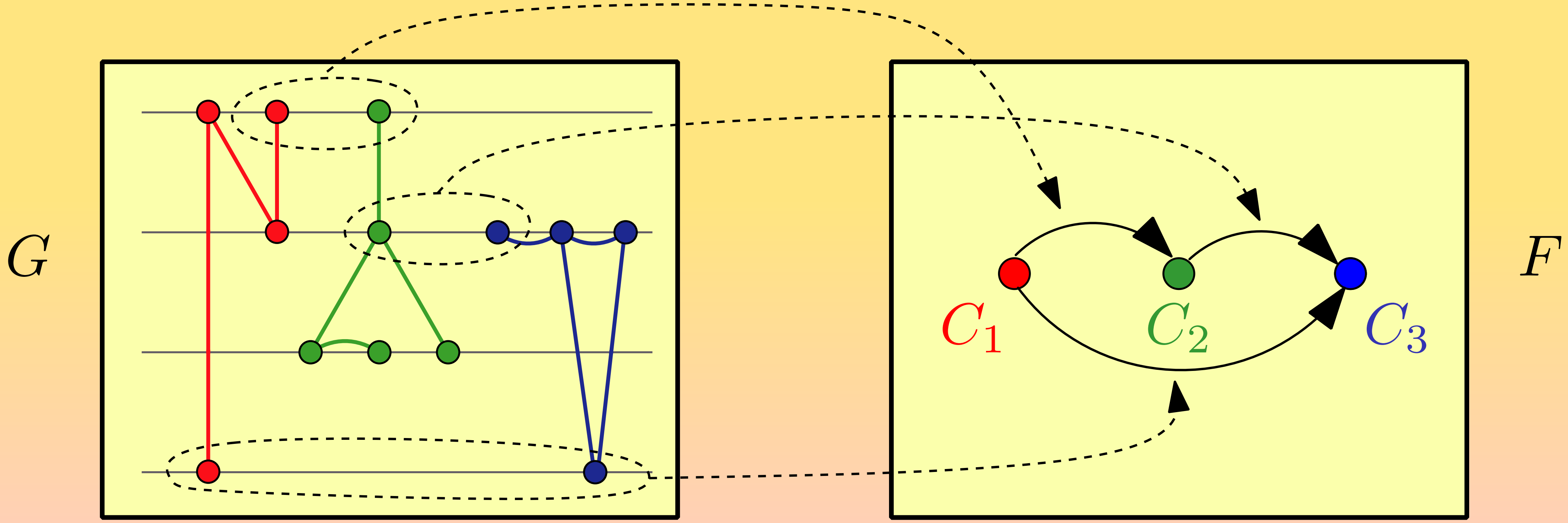


Recursive Algorithm: *Precondition*

Let $G = (V, E)$ be the graph that we want to draw.

Define the *cluster adjacency graph* F as the directed graph...

- whose vertices correspond to clusters in G
- that has a directed edge between the cluster vertices C and C' if there is a level i on which a vertex of C or an edge connected to a vertex of C lies to the left of a vertex or edge of C'

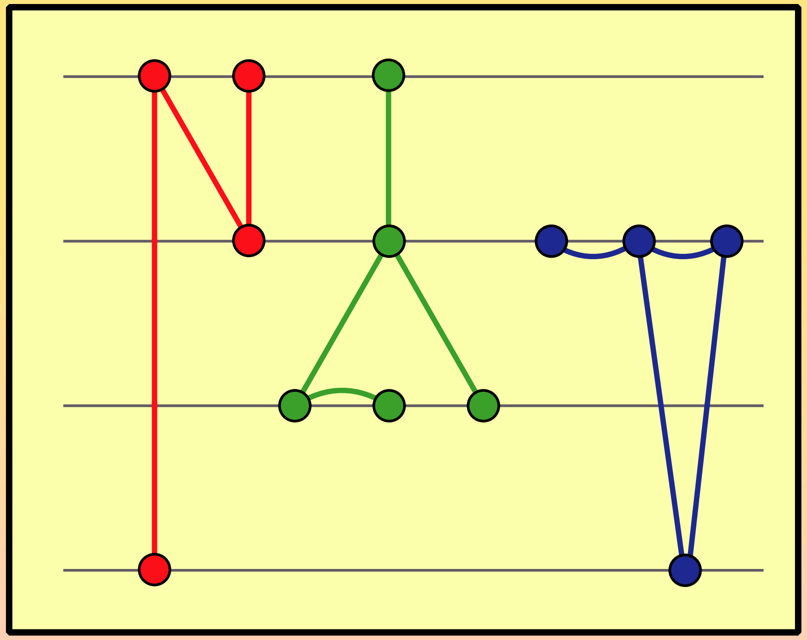


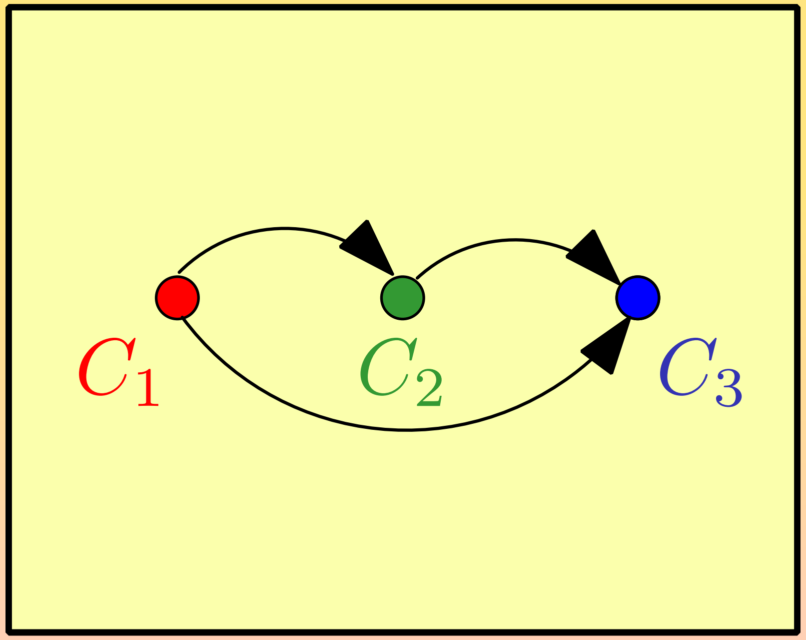
Recursive Algorithm: *Precondition*

Let $G = (V, E)$ be the graph that we want to draw.

Lemma. The *Recursive Algorithm* can be used to draw G if the *cluster adjacency graph* F is acyclic.

G

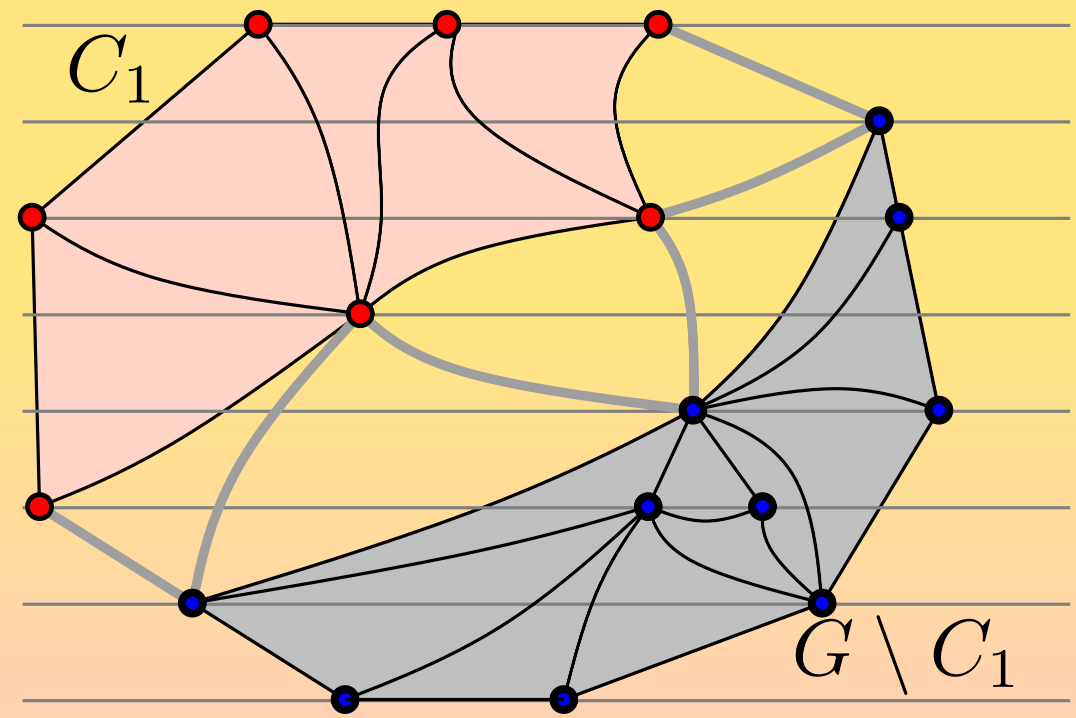




F

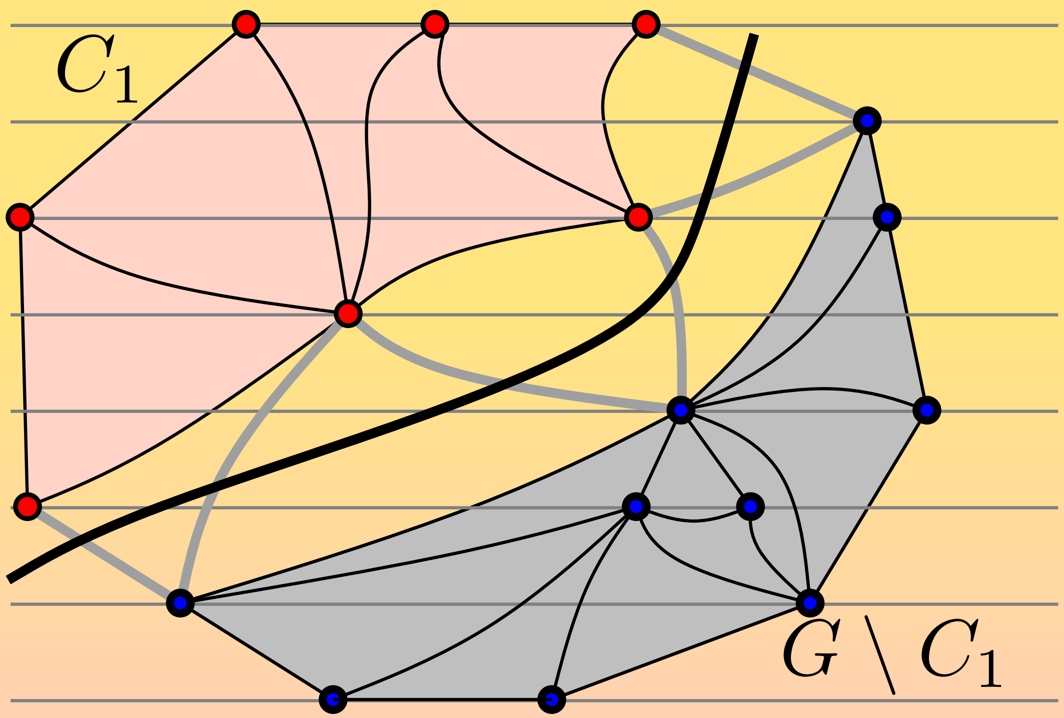
Recursive Algorithm: *Main Concepts*

- Triangulate G in $O(n)$ time.



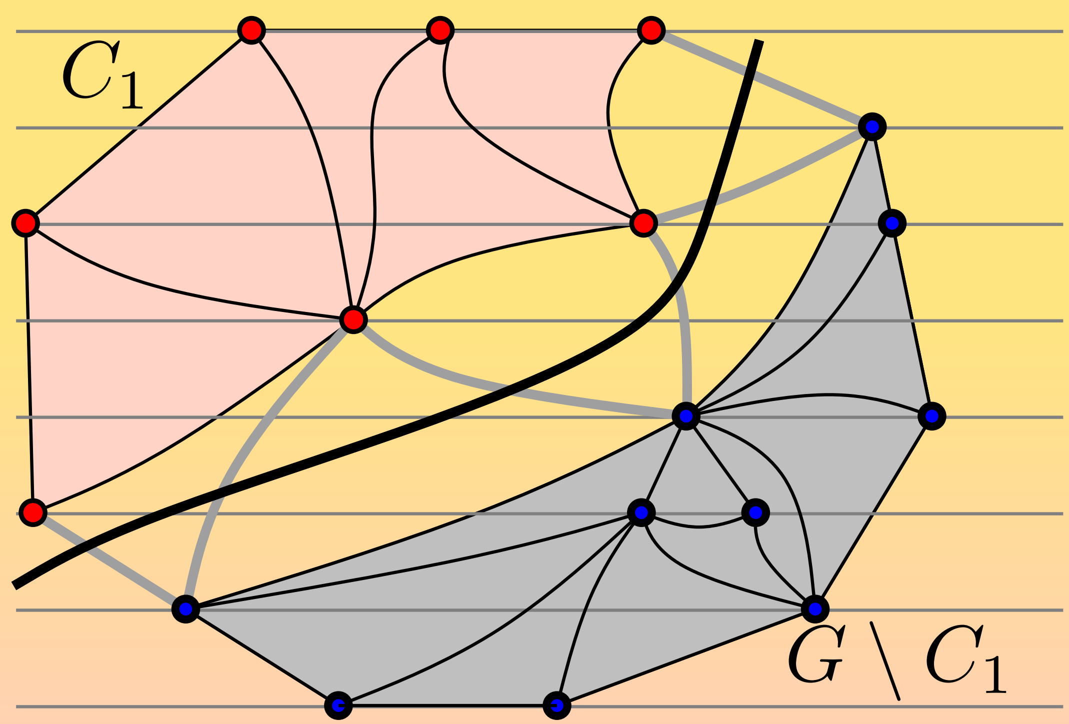
Recursive Algorithm: *Main Concepts*

- Triangulate G in $O(n)$ time.
- C_1 = first cluster of cluster adjacency graph F (in topological order).
- Split G into graphs G_1 and G_2 induced by the vertex sets V_1 of C_1 and $V_2 = V \setminus V_1$.



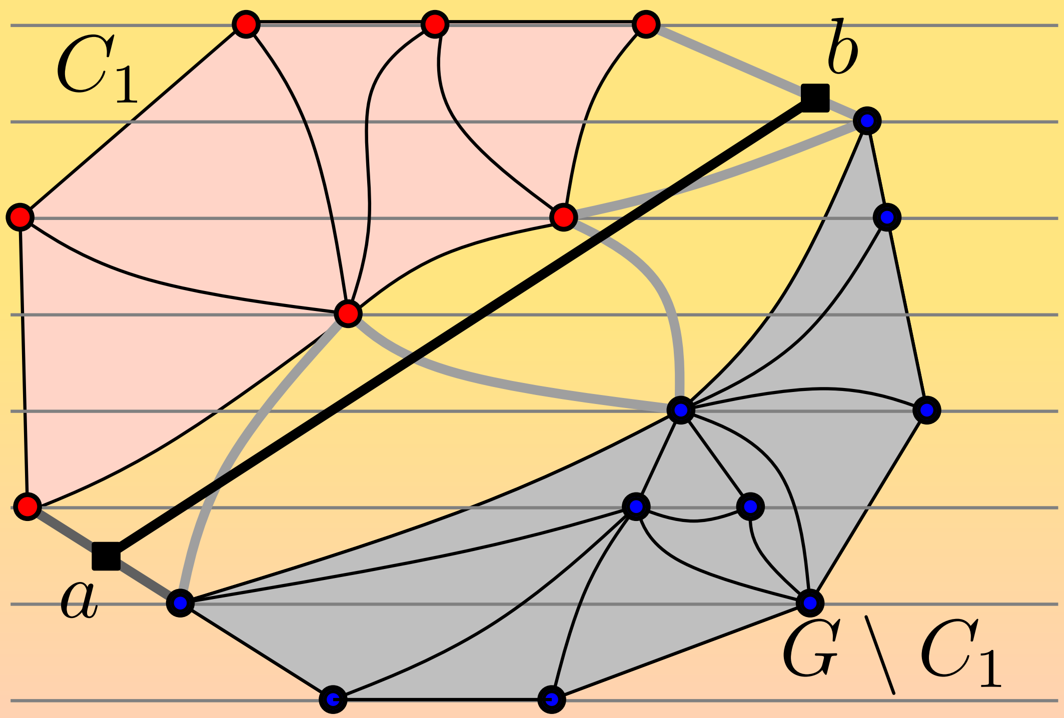
Recursive Algorithm: *Main Concepts*

- The cut occurs through the *gray edges* between C_1 and $G \setminus C_1$ and can be computed in linear time.



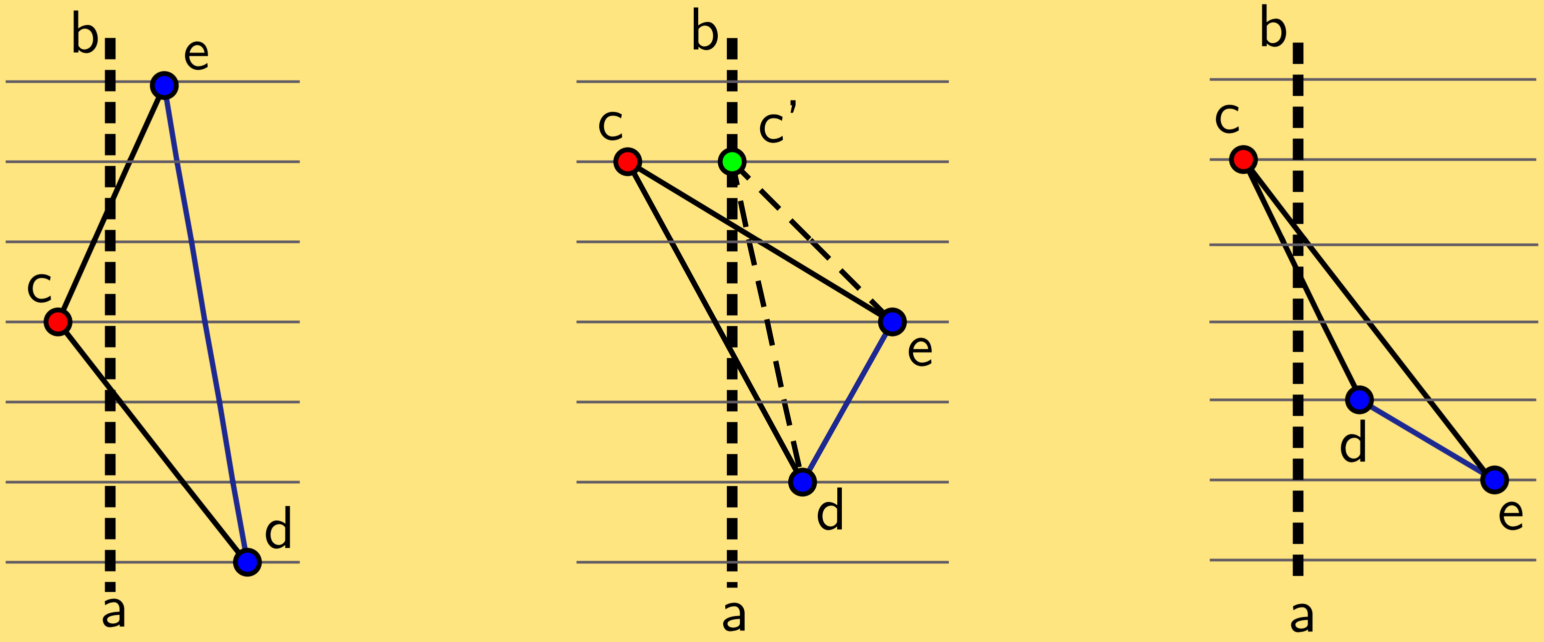
Recursive Algorithm: *Main Concepts*

- The cut occurs through the *gray edges* between C_1 and $G \setminus C_1$ and can be computed in linear time.
- Split the drawing of G by a straight line ab.
- Draw C_1 in linear time using the algorithm of *Eades at al.*
- Treat $G \setminus C_1$ the same way recursively.

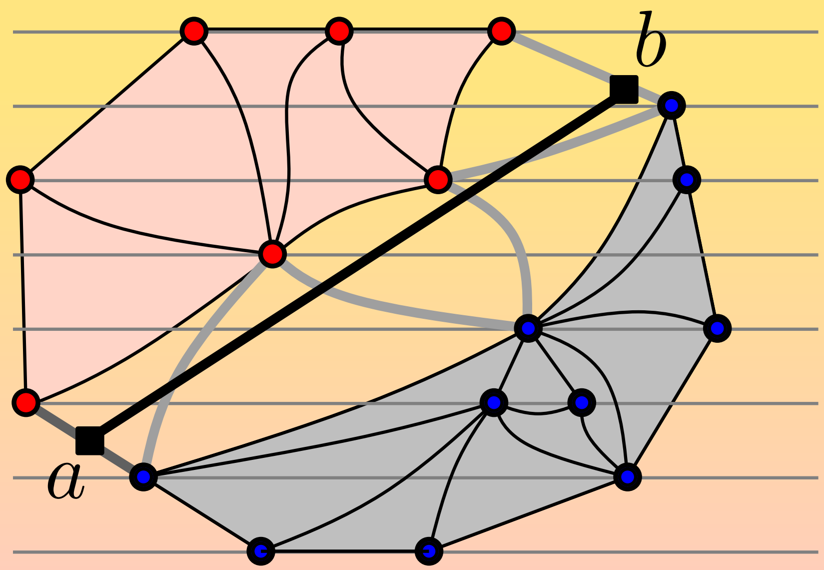


Recursive Algorithm: *Main Concepts*

What happens to the gray edges?

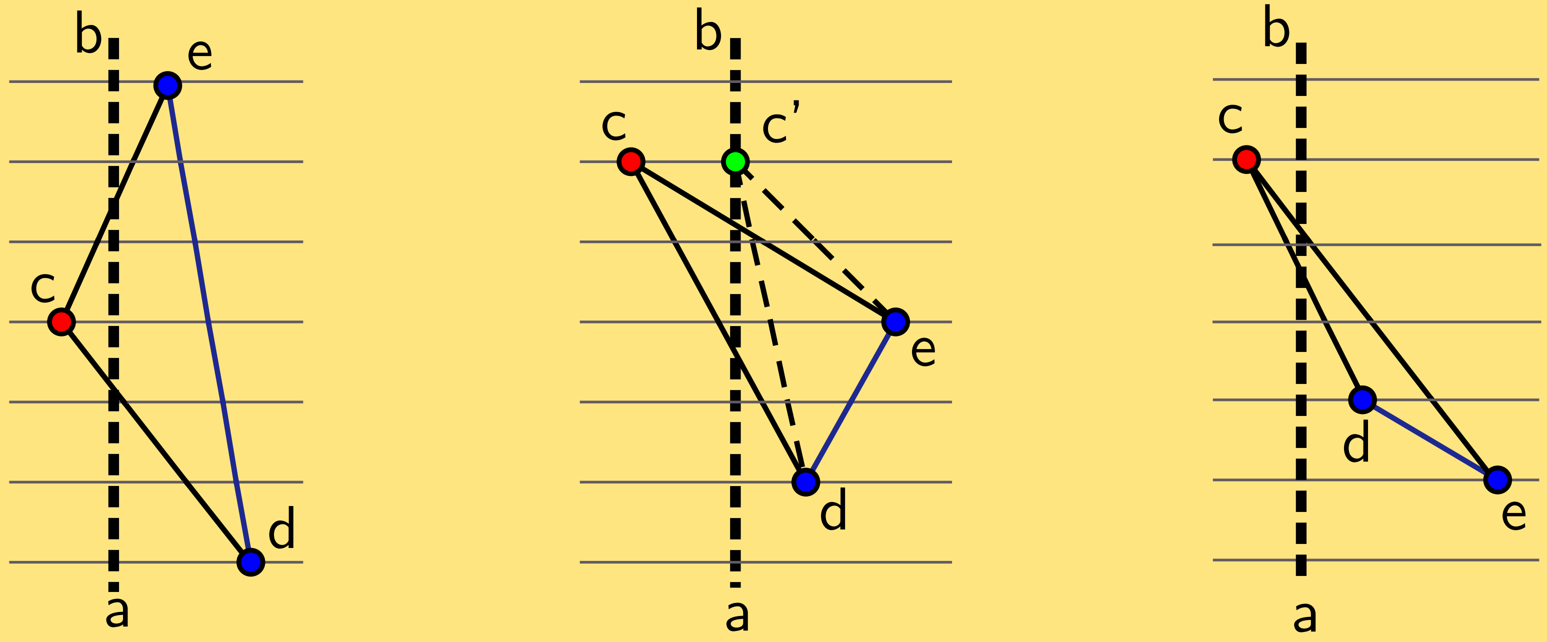


The three types how a face can be split



Recursive Algorithm: *Main Concepts*

What happens to the gray edges?



The three types how a face can be split

> **Theorem.** If the cluster adjacency graph is acyclic, then a straight-line drawing with convex cluster regions can be computed in $O(n^2)$ time.

Separating-Path Algorithm

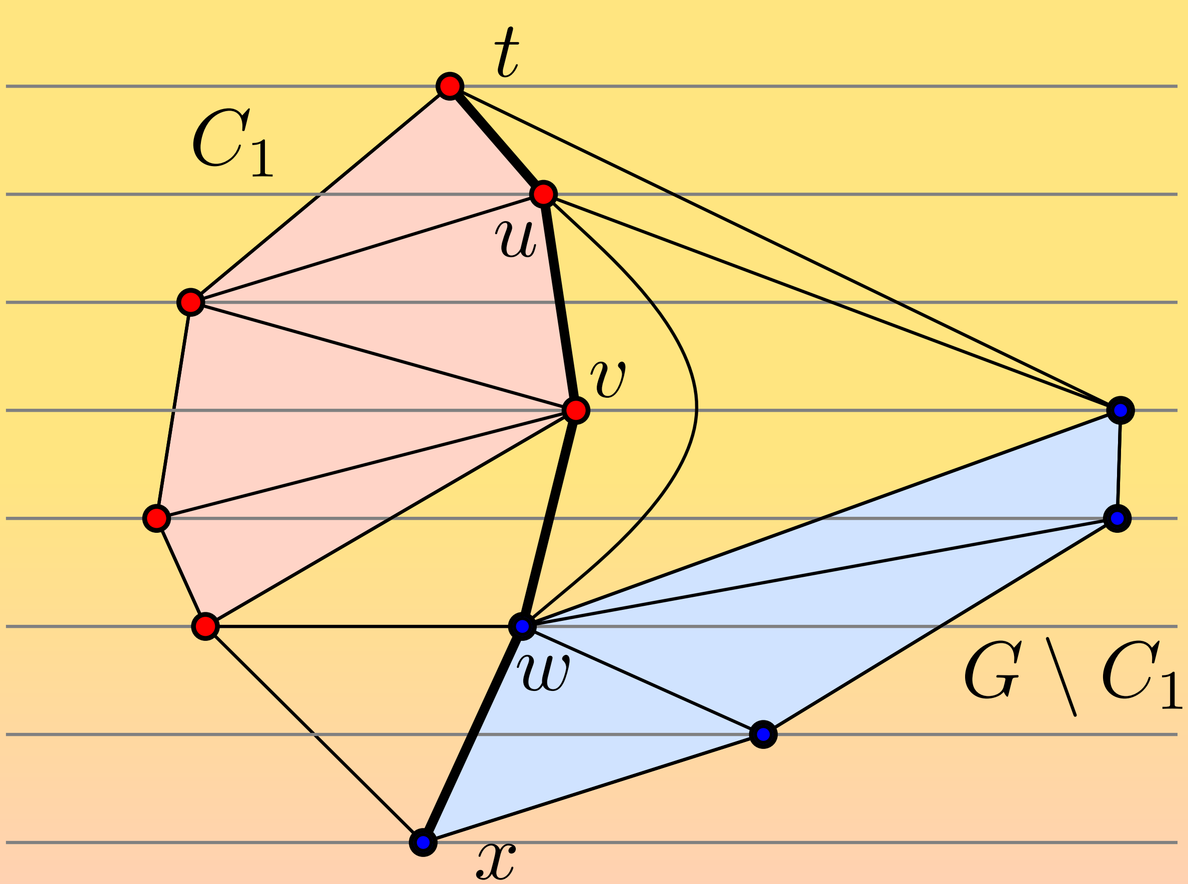
Separating-Path Algorithm

Monotone Separating Paths

Let $G = (V, E)$ be a clustered hierarchical graph.

A path Π in G is a *monotone separating path* if ...

- Π is a path between two vertices on the boundary of G,
- Π is y-monotone, and
- $G \setminus \Pi$ has two connected components G_1 and G_2 whose vertices are in different clusters



Separating-Path Algorithm

Finding a monotone separating path

In the following suppose that G has only two clusters.

Definition. An edge (u, v) is called *separating* if it separates the clusters on all layers that it spans.

separating edge

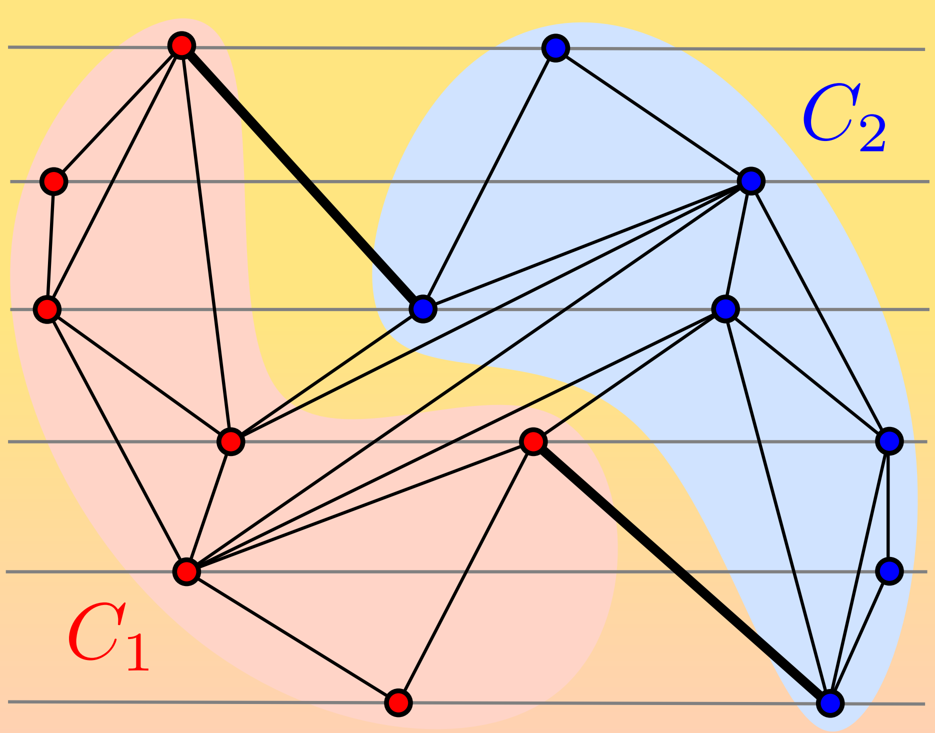
not a separating edge

Separating-Path Algorithm

Finding a monotone separating path

In the following suppose that G has only two clusters.

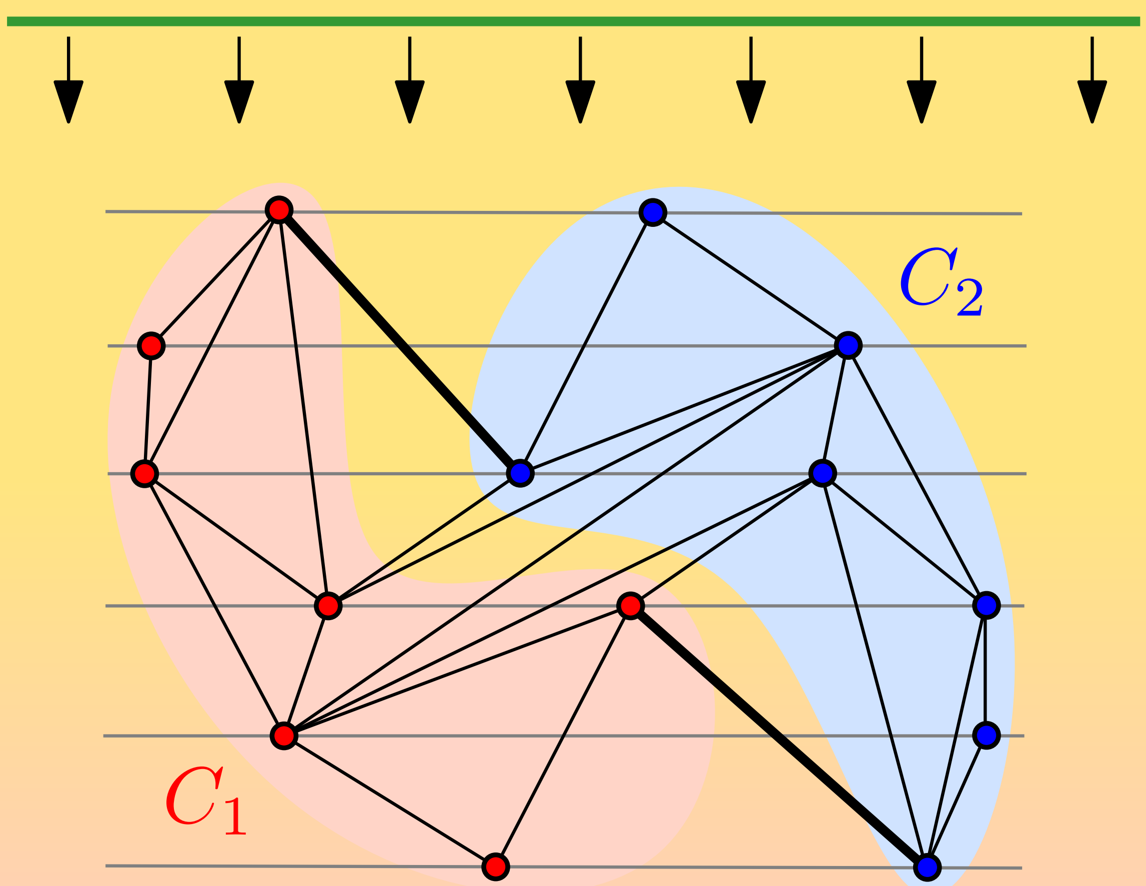
Definition. The two edges on the boundary of G whose endpoints are in different clusters are called $gates$.



Separating-Path Algorithm

Finding a monotone separating path

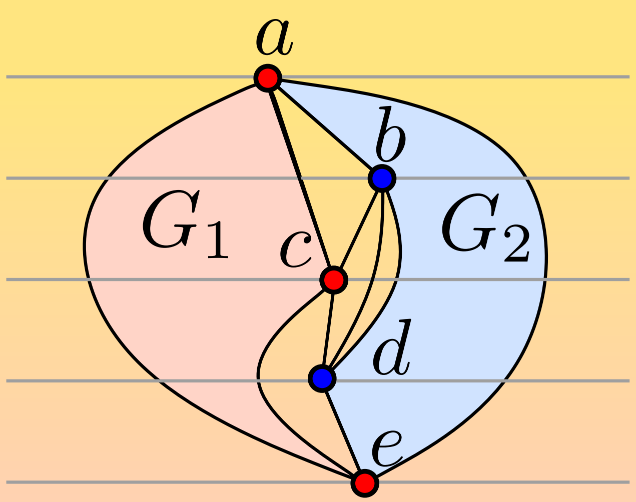
- A monotone separating path connects one endpoint of the first gate with one endpoint of the second gate using only separating edges.
- It can be found in $O(n)$ time by a line sweep.



Separating-Path Algorithm

Drawing of the graph

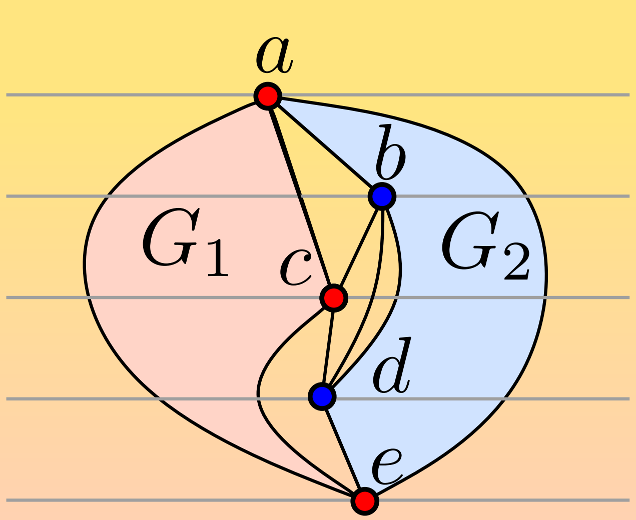
1. Compute *left path* and *right path* using shortcuts.
2. Draw *left path* and *right path* using parallel line segments.
3. Compute drawings of G_1 and G_2 using the algorithm of *Eades et al.*
4. Place the drawings of G_1 and G_2 at distance ξ from each other.
5. Place the remaining vertices on two arcs using distance δ.

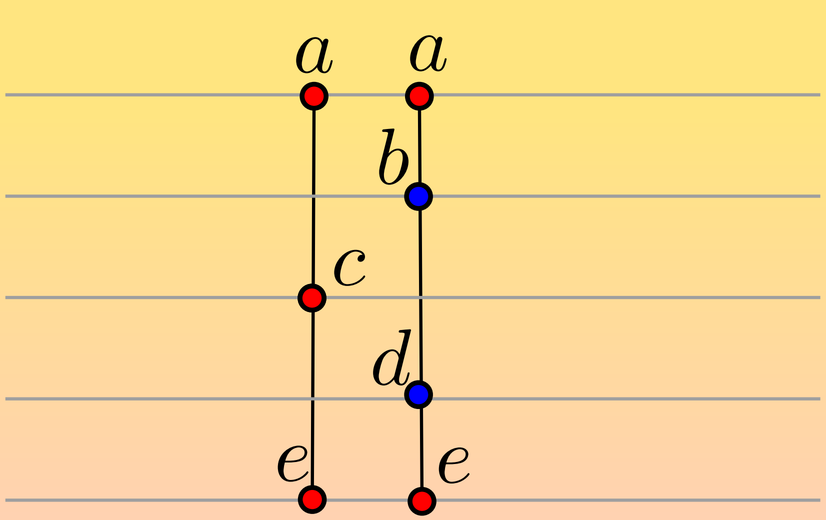


Separating-Path Algorithm

Drawing of the graph

1. Compute *left path* and *right path* using shortcuts.
2. Draw *left path* and *right path* using parallel line segments.
3. Compute drawings of G_1 and G_2 using the algorithm of *Eades et al.*
4. Place the drawings of G_1 and G_2 at distance ξ from each other.
5. Place the remaining vertices on two arcs using distance δ.

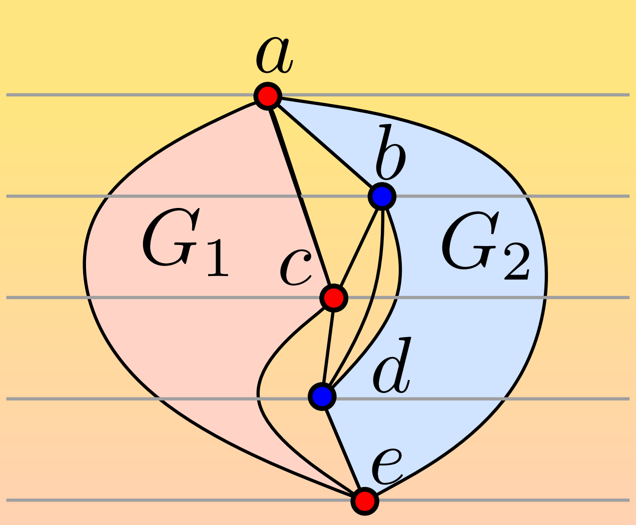


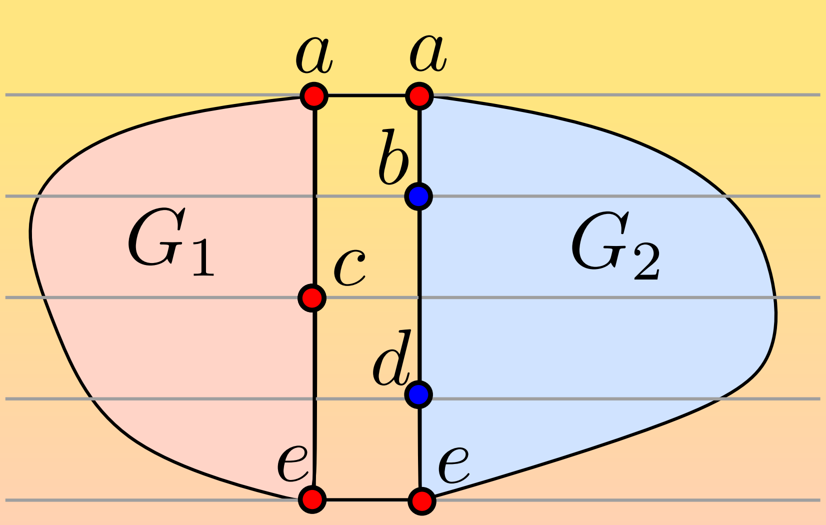


Separating-Path Algorithm

Drawing of the graph

1. Compute *left path* and *right path* using shortcuts.
2. Draw *left path* and *right path* using parallel line segments.
3. Compute drawings of G_1 and G_2 using the algorithm of *Eades et al.*
4. Place the drawings of G_1 and G_2 at distance ξ from each other.
5. Place the remaining vertices on two arcs using distance δ.

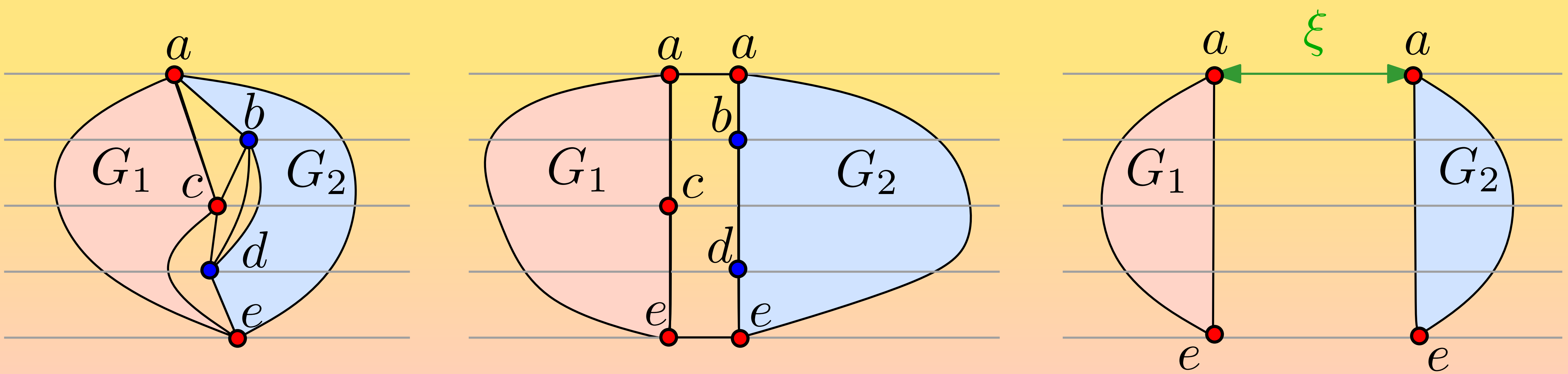




Separating-Path Algorithm

Drawing of the graph

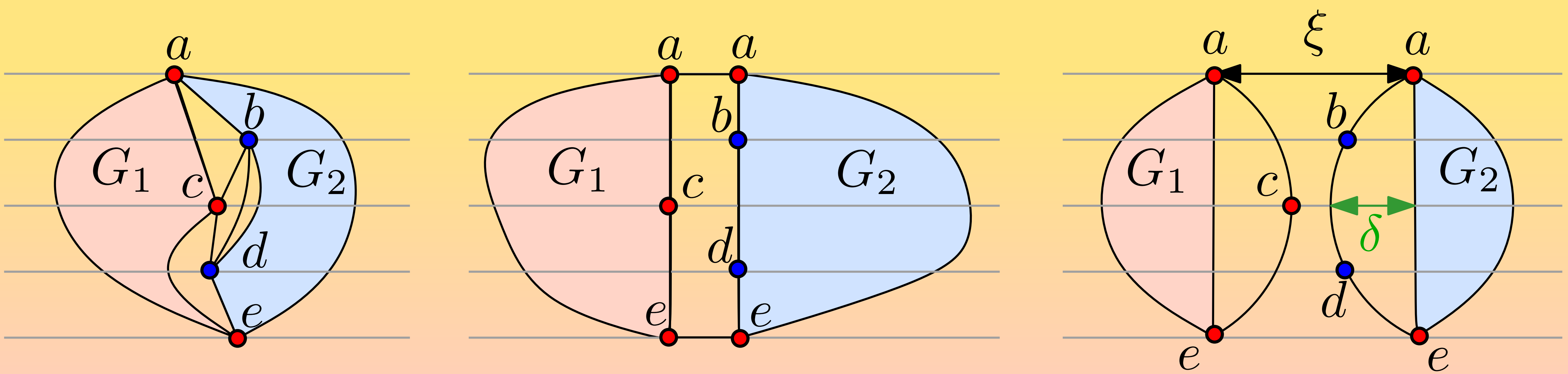
1. Compute *left path* and *right path* using shortcuts.
2. Draw *left path* and *right path* using parallel line segments.
3. Compute drawings of G_1 and G_2 using the algorithm of *Eades et al.*
4. Place the drawings of G_1 and G_2 at distance ξ from each other.
5. Place the remaining vertices on two arcs using distance δ.



Separating-Path Algorithm

Drawing of the graph

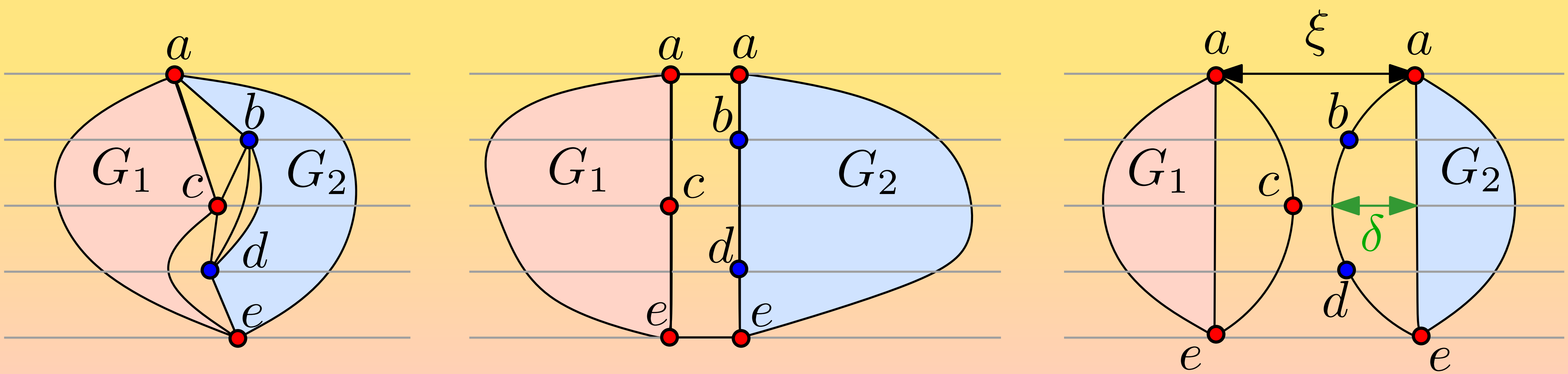
1. Compute *left path* and *right path* using shortcuts.
2. Draw *left path* and *right path* using parallel line segments.
3. Compute drawings of G_1 and G_2 using the algorithm of *Eades et al.*
4. Place the drawings of G_1 and G_2 at distance ξ from each other.
5. Place the remaining vertices on two arcs using distance δ.



Separating-Path Algorithm

Drawing of the graph

> **Theorem.** Given a c-planar clustered hierarchical graph G with two clusters and a monotone separating path, a straight-line drawing of G with convex cluster regions can be computed in linear time.



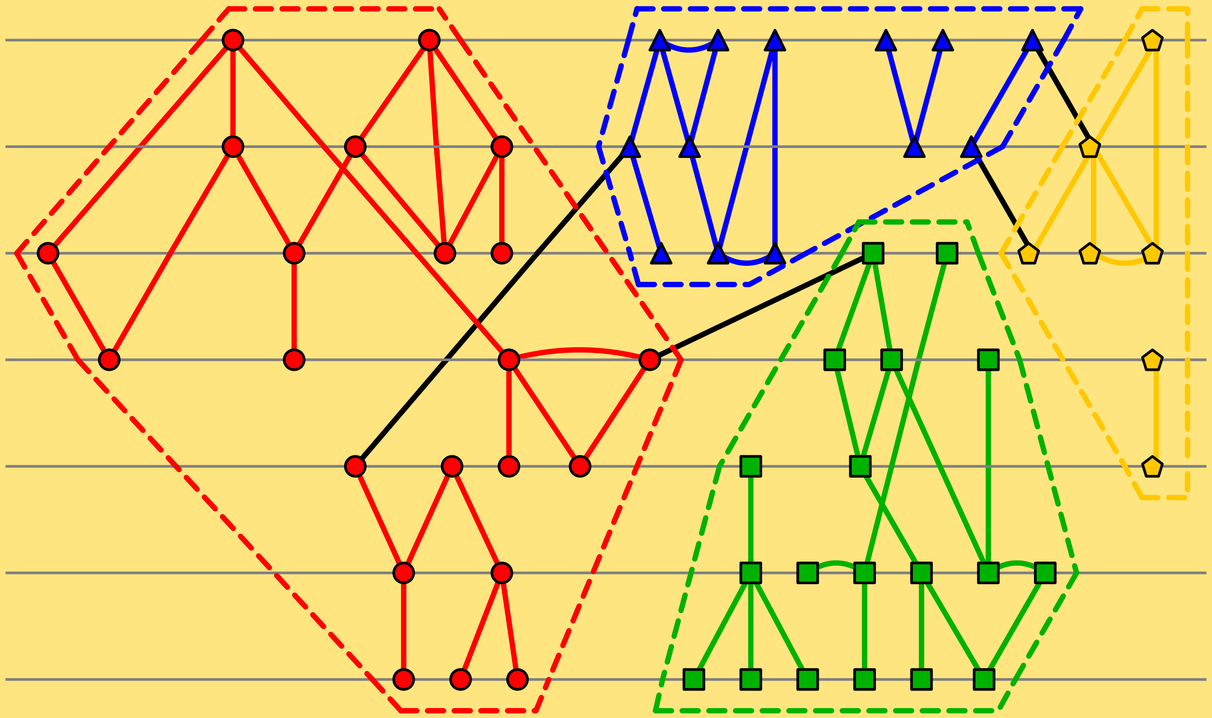
Conclusion

Conclusion

Three new methods to produce drawings of *clustered hierarchical graphs* with straight-line edges and non-intersecting cluster regions:

- Recursive Algorithm
  - works only if cluster adjacency graph is acyclic,
  - runs in $O(n^2)$ time.
- Separating Path Algorithm
  - only applicable if a monotone separating path exists,
  - runs in $O(n)$ time.
- Linear Programming Formulation
  - slowest of our methods (roughly $O(n^{3.5})$ time),
  - needs only $O(n)$ variables and $O(n)$ constraints,
  - produces nicest results due to global optimization.

Thank you for your attention!



Do you have any questions?

Check out our Java applet at: http://i11www.ira.uka.de/clusteredgraph