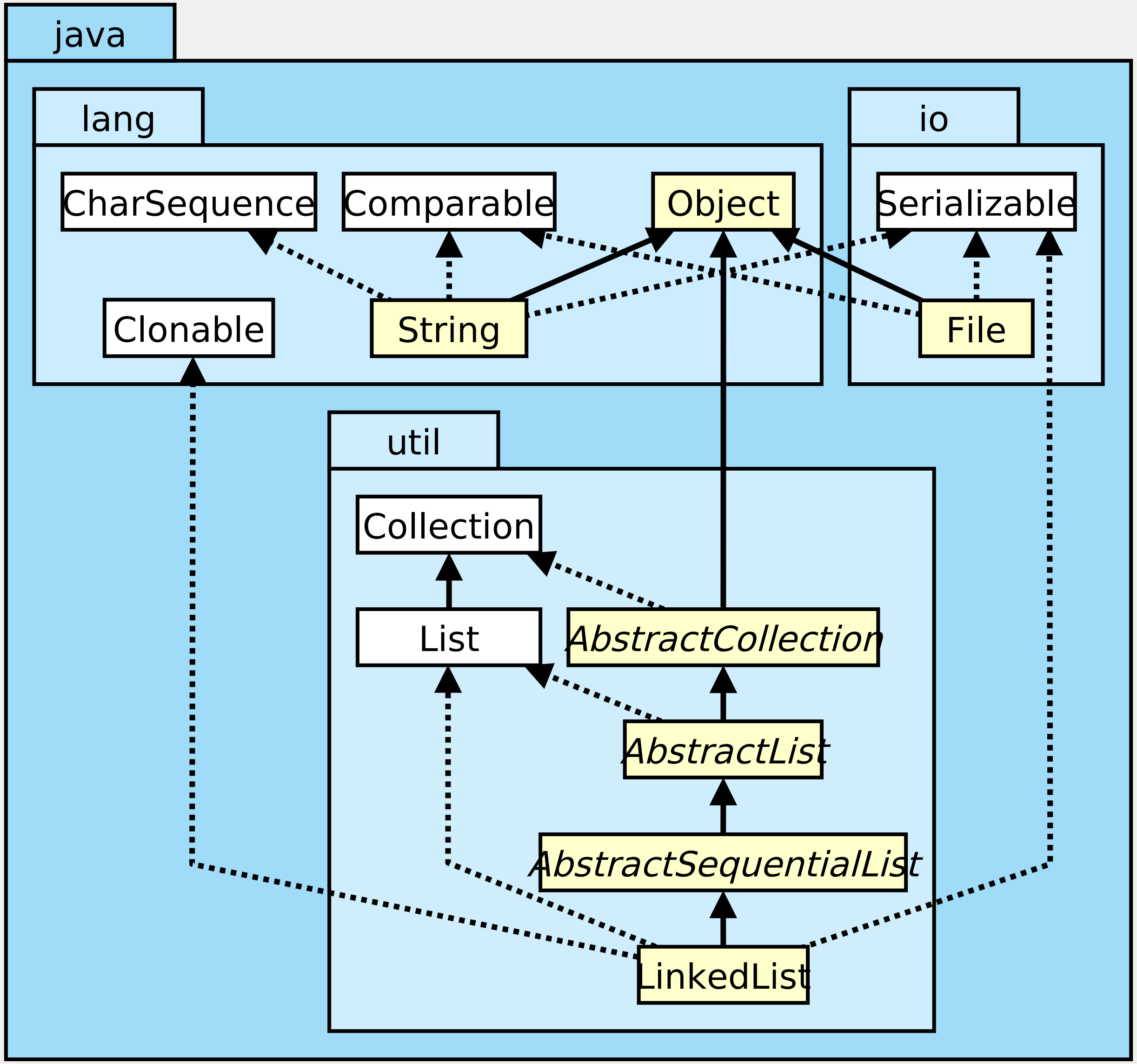
Clustered Level Planarity

Michael Forster, Christian Bachmaier

University of Passau

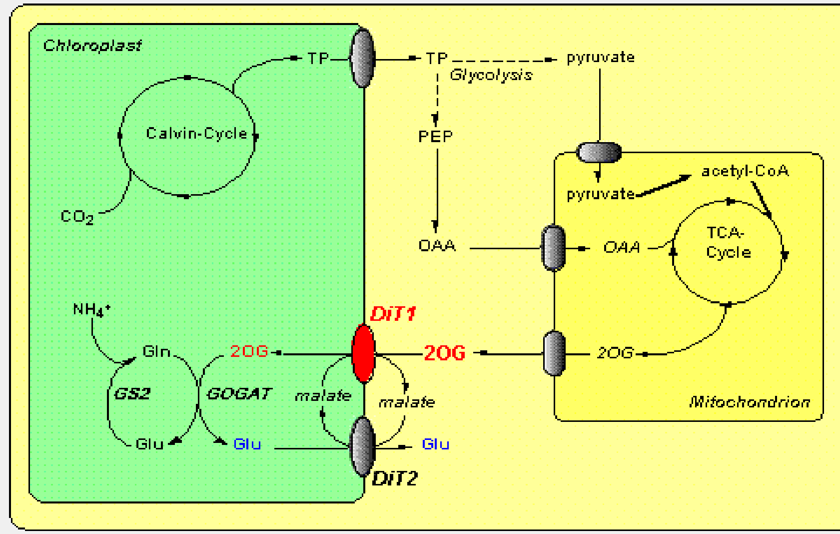
UML Diagrams

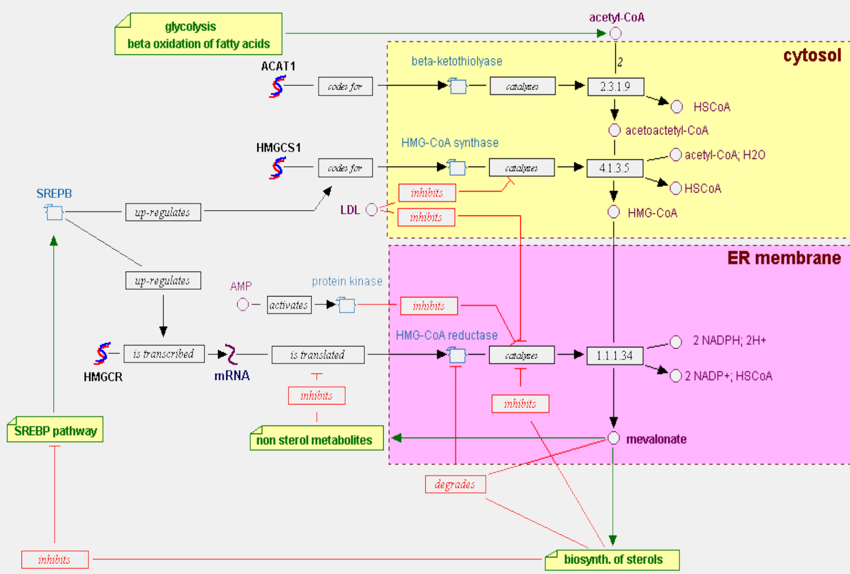
- Vertices
  - Classes
  - Interfaces
- Directed edges
  - Inheritance
  - Association
- Recursive clustering
  - Packages
  - Arbitrary depth

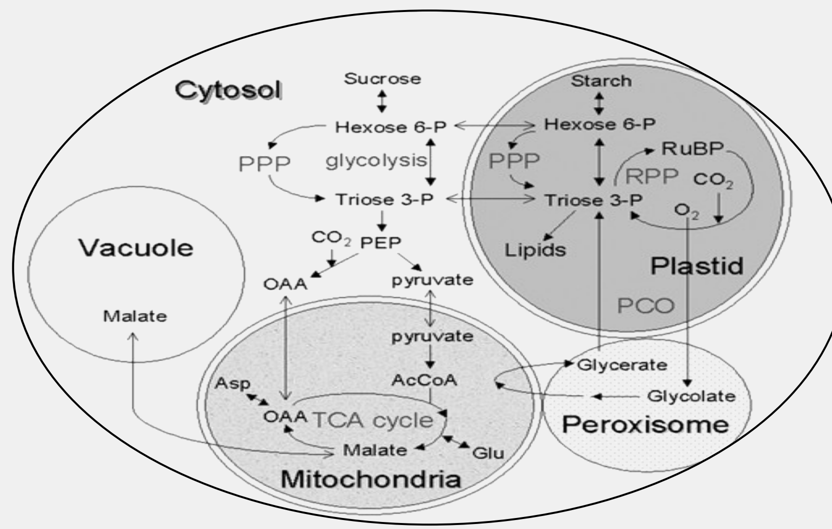


Biochemical Pathways

- Reaction networks
  - Substances / reactions
  - Directed (hyper) graphs
- Compartments
  - Separate cell regions
  - Nested







Overview

- Introduction
- Definition and concepts
- Level planarity testing
  - Algorithm of Jünger, Leipert, Mutzel (1998)
  - PQ-trees
- Clustered level planarity testing
  - Differences to level planarity
  - Extension of the algorithm
- Summary

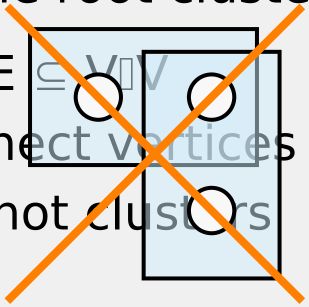
Definitions and Concepts

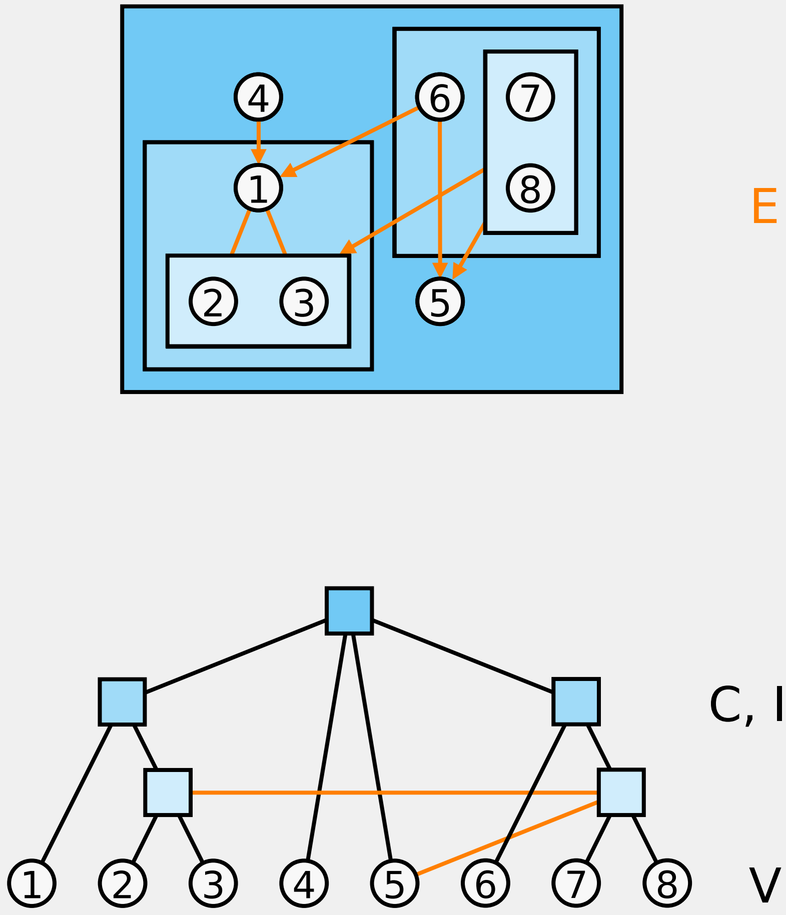
Clustered (Level) Graphs

Clustered (Level) Planarity

Clustered Graphs

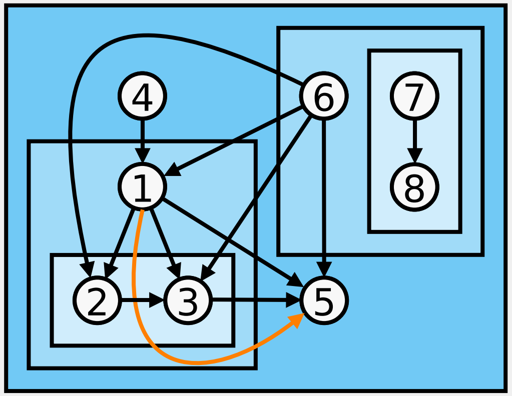
- Clustered graph: $G = (V, E, C, I)$
- Vertices: V
- Clusters: C
  - Contain vertices
  - Or other clusters
- Inclusion relation: I
  - $T = (V \cup C, I)$ is a tree
  - No cluster overlap
  - Single root cluster
- Edges: $E \subseteq V \times V$
  - Connect vertices
  - But not clusters





Clustered Planarity (c-Planarity)

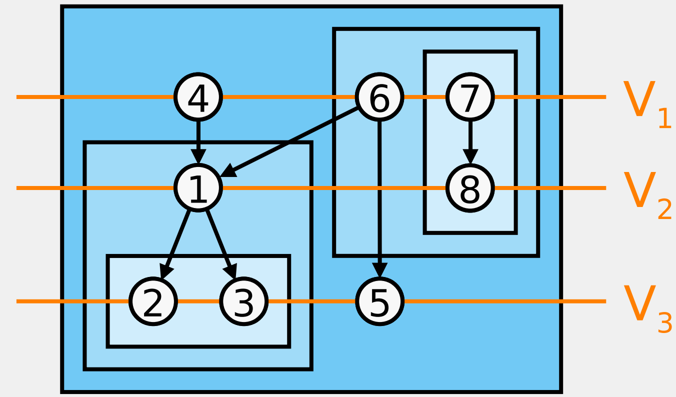
- Is G planar?
- Can G be drawn without
  - edge crossings
  - edge/clusters crossings?
- Complexity
  - O(|V|) for c-connected graphs [Feng, Cohen, Eades, 1995]
  - Open in the general case

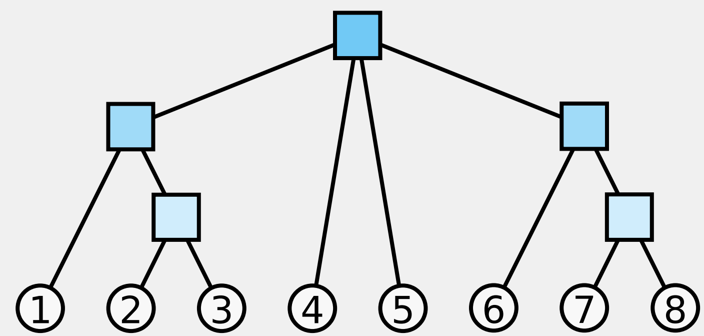


One of the TOP 10 open problems in graph drawing!

Clustered Level Graphs

- Clustered level graph:
  $G=(V_1 \cup V_2 \cup \ldots \cup V_k, E, C, I)$
  - Clustered graph: (V, E, C, I)
  - Leveling: $V = V_1 \cup V_2 \cup \ldots \cup V_k$
- Drawing conventions
  - Clusters
    - Connected regions
    - No overlaps
    - Recursively nested
    - Simple: convex polygons, rectangles
  - Levels
    - Horizontal lines
  - Edges
    - Strictly downwards
    - Horizontal edges: see previous talk
- Question: is G planar?





Overview

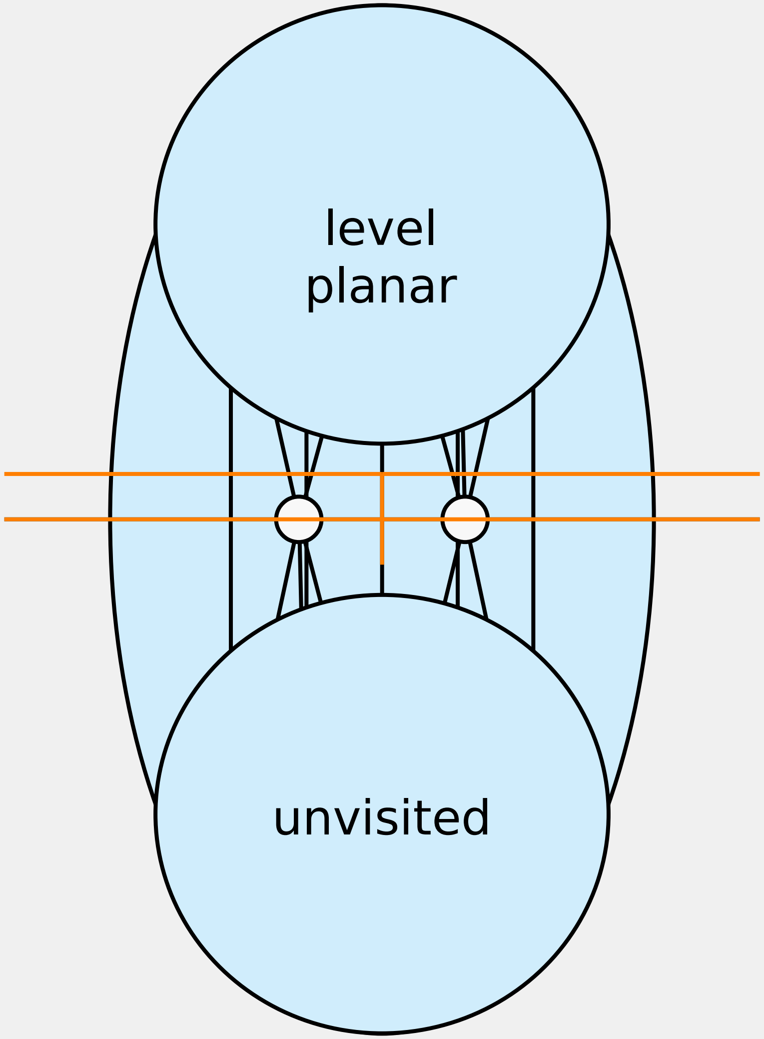
	Class of graphs	Complexity
Clustered Planarity	arbitrary	open
	c-connected	$\Theta(\lvert V \rvert)$
Level Planarity	arbitrary	$\Theta(\lvert V \rvert)$

Level Planarity Testing

Algorithm of Jünger, Leipert, Mutzel (1998)

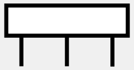
Level Planarity Testing

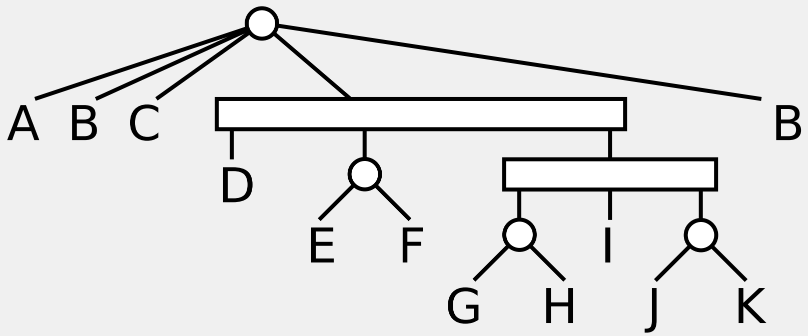
- Similar to planarity testing (vertex addition method, LEC)
- “Sweep-line”
  - Traverse the graph level by level
  - Store the “admissible” edge permutations
  - For each vertex: REDUCE
    - Check planarity
    - Update permutation set
  - For each vertex: REPLACE
    - Insert outgoing edges
  - Next level
- Precondition: proper, single source
- Running time: O(|V|)
- Data structure: PQ-trees



Storing the permutations

- PQ-trees [Booth, Lueker, 1976]
  - Leaves represent edges (“virtual vertices”)
  - P-nodes:
    - Represent cut vertices
    - Arbitrary permutations of children
  - Q-nodes:
    - Represent biconnected components
    - Only “reversion”





PQ-trees: Operations

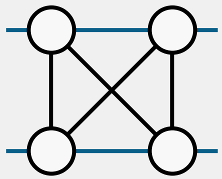
- REDUCE(T, S):
  - Restricts the permutation set:
    Leaves S must be consecutive
  - Failure is possible → graph is not planar
  - Traverse tree bottom-up
    - At each node: local updates
    - Apply one of 11 “Templates”
  - Afterwards: Reduced leaves correspond to a subtree of T
- REPLACE(T, S, S'):
  - Substitute leaves S with leaves S'
  - Leaves S must already be consecutive

Clustered Level Planarity Testing

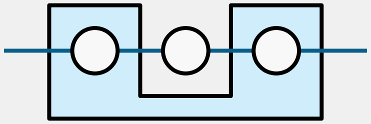
Needed Extensions

Clustered Level Planarity Restrictions

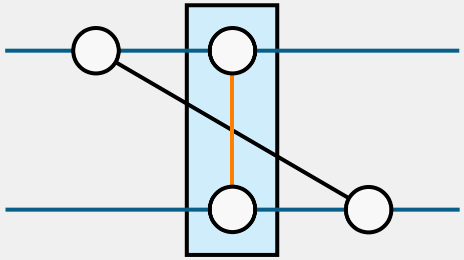
- Edge/Edge-Restriction



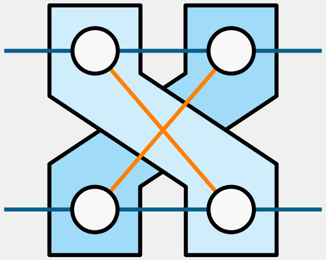
- Cluster/Level-Restriction



- Edge/Cluster-Restriction



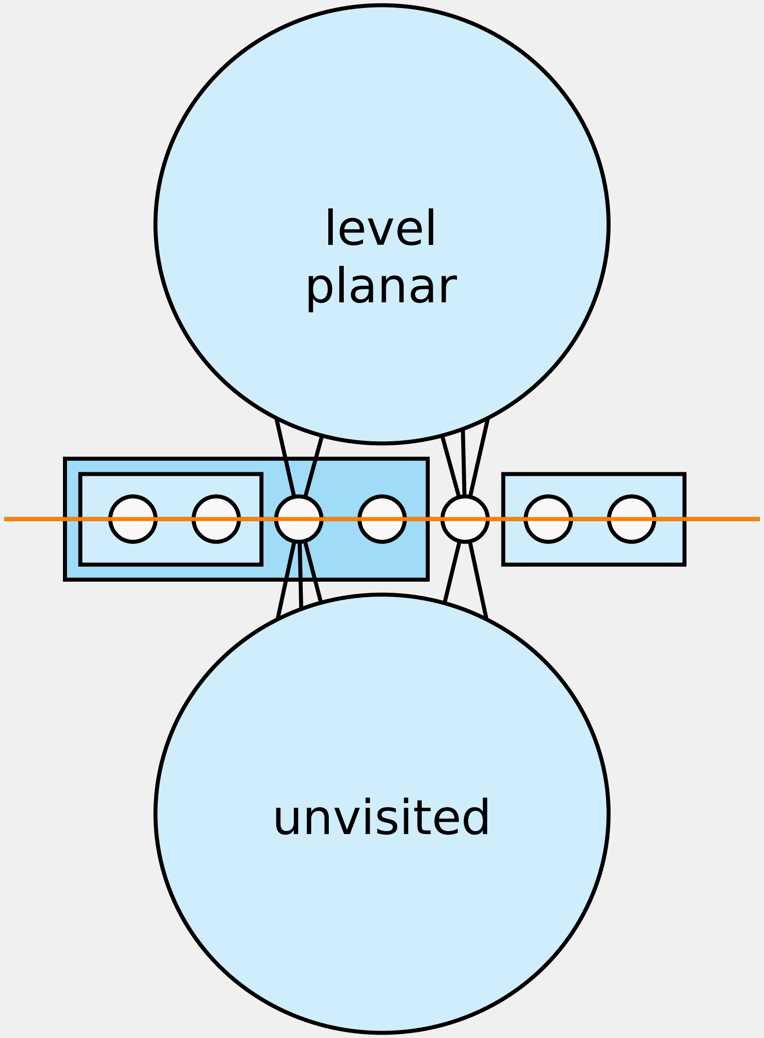
- Cluster/Cluster-Restriction



- Property of the graph: level connectivity
  - At least one edge between consecutive levels
  - Weak form of cluster connectivity
  - Similar to cluster planarity

Clustered Level Planarity: Algorithm

- "Sweep-line"
  - Traverse the graph level by level
  - Store the "admissible" edge permutations
  - For each vertex: REDUCE
    - Check planarity
    - Update permutation set
  - REDUCE-CLUSTERS
  - For each vertex: REPLACE
    - Insert outgoing edges
  - Next level

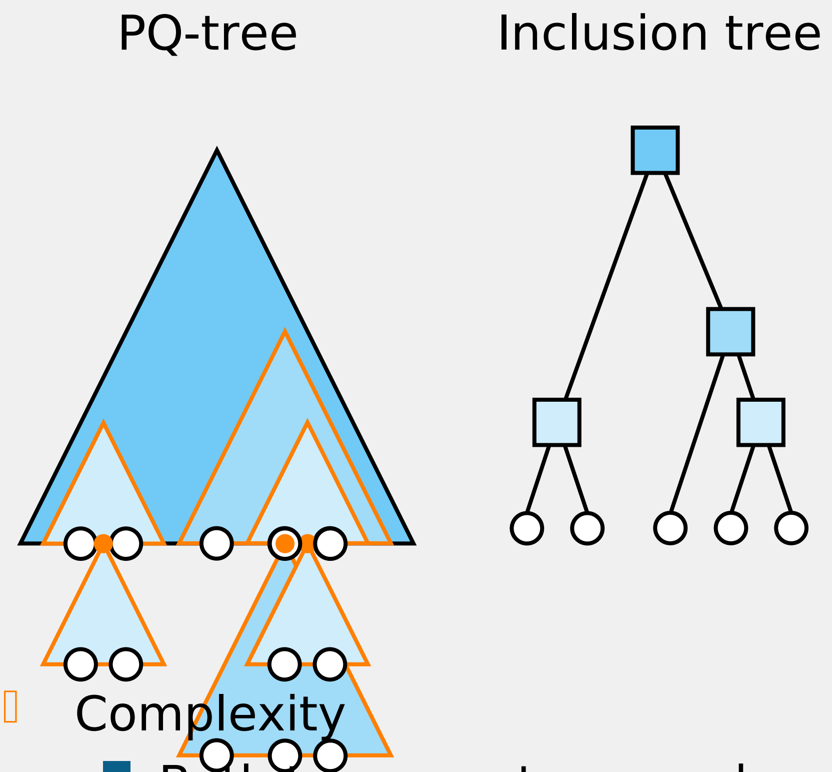


REDUCE-CLUSTERS

- Given: PQ-tree containing admissible permutations
- Wanted: Restricted permutation set
  - Contents of a cluster must be consecutive
  - Invalid permutations are removed
- Idea
  - Semantics of the PQ-tree operation REDUCE are the same
  - Reuse REDUCE
- Naive Algorithm
  - For each cluster c on the current level
    - REDUCE(T, {vertices contained in c})
  - Running time: $O(|V| \cdot |C|)$
- Improvement
  - Single traversal of the PQ-tree
  - Running time: $O(|V|)$

REDUCE-CLUSTERS

- Compare clusters and PQ-tree
  - Every cluster corresponds to a subtree of the PQ-tree
  - This is only true after the reduction
- Nested clusters
  - Child in the inclusion tree
  - Corresponding PQ-subtrees are nested
- Idea
  - Reuse the inner PQ-subtree for the outer PQ-subtree
  - Simultaneously traverse trees bottom up



- Complexity
  - Both trees are traversed once per level
  - Both are of size O(|V|)
  - Overall running time O(k • |V|)

Summary
Future Work

Results and Open Problems

Summary / Future Work

- Cluster-Level-Planarity
  - Test and Embedding
    - $O(k \cdot |V|)$ running time
    - Restricted graph class
  - Combinable with other advanced level planarity concepts
- Open
  - Test for arbitrary clustered level graphs
  - Linear time algorithm

Similar situation as with c-planarity

Questions?

Thank you for your attention!