

# X-graphs of Y-graphs visualization and interaction

Vladimir Batagelj  
Franz J. Brandenburg  
Walter Didimo  
Giuseppe Liotta  
Maurizio Patrignani

Graph Drawing with Applications to Bioinformatics and Social Sciences  
Dagstuhl Seminar N 08191  
04.05.2008-09.05.2008

# Goal

- Identify subgraphs (clusters) that are responsible for the visual complexity of the drawing
- Represent clusters in a succinct way
  - matrices, confluent drawings, etc.
- The reduced graph has a simple structure that can be effectively conveyed to the user



# The model: X-graphs of Y-graphs

Input:

- $\mathbf{G} = (V, E)$

Output:

- A family of disjoint subsets  $V_1, V_2, \dots, V_n$  (clusters) of  $V$ , so that:
  1.  $G(V_i)$  belongs to a specific class (**Y-class**)
  2. The *reduced graph*, i.e., the graph obtained by collapsing each  $V_i$  into a single node, belongs to a specific class (**X-class**)

# Specific Y-classes

- a) Connected graphs with bounded/specified number of nodes
- b) Trees, paths, and cycles
- c) Dense graphs, i.e., with clustering coefficient  $2|E_i| / |V_i|(|V_i|-1)$  greater than a certain threshold
- d) Cliques
- e) Complete bipartite graphs
- f) K-connected graphs
- g) K-cores
- h) Strongly connected digraphs
- i) Stars (subgraphs induced by a vertex and its neighbors)

# Specific X-classes

- a) General graphs
- b) Connected graphs with bounded/fixed number of nodes
- c) Planar graphs
- d) Trees
- e) Paths
- f) Cycles
- g) Directed acyclic graphs
- h) “Low-density” graphs

# Design framework

(X,Y)-decomposition



Visualization of the (X,Y)-graph

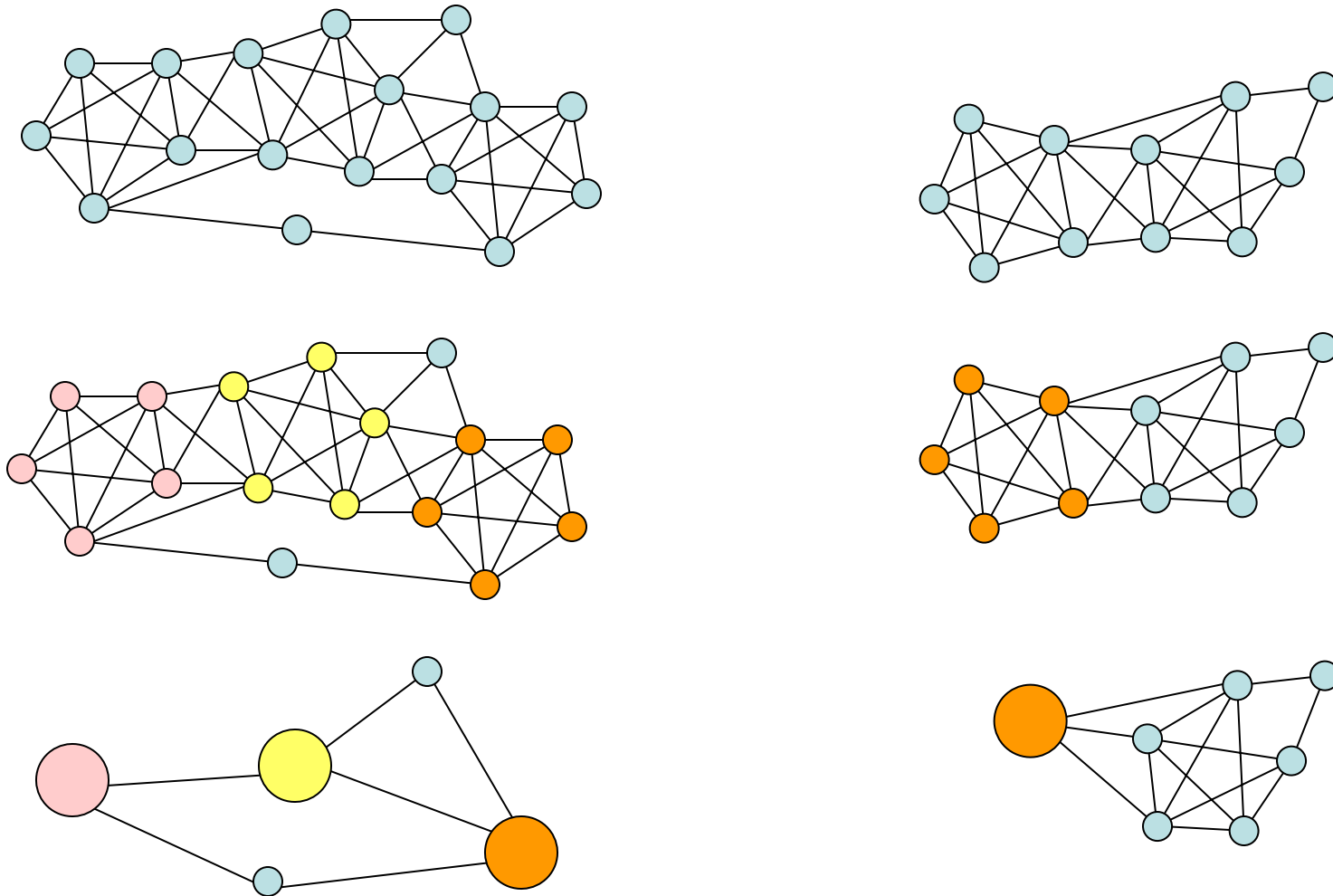


User interaction primitives

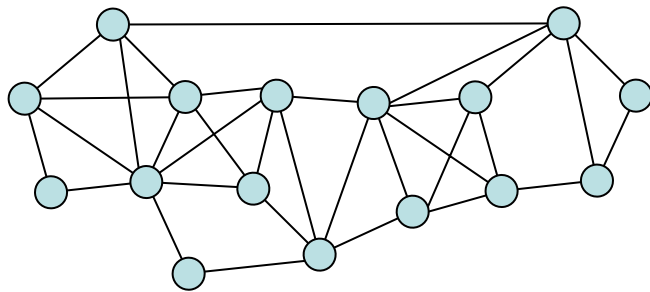
# $(X, Y)$ -decomposition

- a) Existence of solutions
- b) Complexity
- c) Heuristics
- d) Approximation algorithms
- e) FPT-algorithms

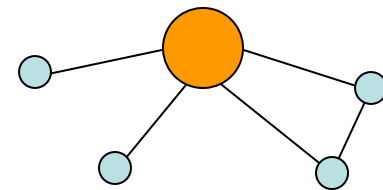
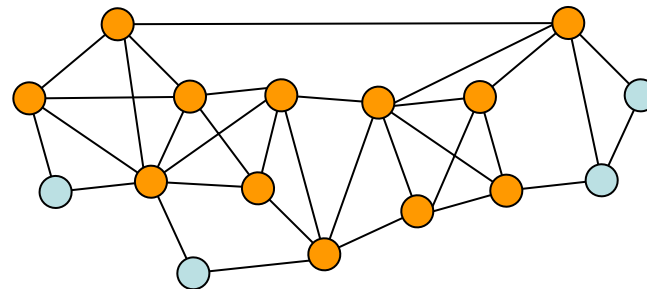
# An example: (planar, $K_5$ )-graphs



# An example: (planar,k-core)-graphs



3-core



X-class/ Y-class	Bounded number of vertices	Tree	3-Cycle	Cycle	Non-trivial path	Single k- clique	Single k- star	General graph	Planar
Bounded number of vertices	$O(1)$	$O(n)$		$O(n)?$					
Path		NP- comple te [GD'98]			NP-complete [Le,Le,Mueller 03]				
Cycle									
Bounded path		Polyno mial [GD'98]							
Bounded cycle									
Triangle								NP- complete [GJ'79 (GT11)]	
Clique	NP- hard		NP-Hard [GD'97]	Polynomial if the input graph has diameter > 3 [GD'97]					NP-hard [K-JCT-91]
K-Clique									K=3? K=5?
K-core (max k)									$O(m + n \log(n))$

# (planar, k-core)-graphs

## **Theorem:**

- Any graph is a (planar, k-core)-graph for some  $k$
- It is possible to determine the highest value of  $k$  such that  $G(V, E)$  is a (planar, k-core)-graph in  $O(m + n \log(n))$  time

# Algorithm for (planar,k-core)-graphs

- Compute the core number of each vertex ( $O(m)$ )
- Rearrange the adjacency lists based on decreasing core numbers
- Perform a binary search combined with planarity testing on the reduced graph

# (bounded, tree)-graphs

## **Theorem:**

- There is an  $O(n)$  algorithm to recognize  $(X, Y)$ -graph, for  $X$ =bounded number of nodes and  $Y$ =tree

## **Proof:**

- Based on the bounded tree-width of the graph
- Analogous to a proof in [Schreiber, Skodinis, GD'98]

# (bounded, clique)-graphs

## **Theorem:**

- It is NP-hard to recognize  $(X, Y)$ -graphs, where  $X$ =bounded number of nodes and  $Y$ =clique

## **Proof:**

- Similar to the NP-hardness proof of partition into cliques [Garey, Johnson, '79]

# Design framework

(X,Y)-decomposition



Visualization of the (X,Y)-graph



User interaction primitives

# Visualization of the $(X, Y)$ -graph

- Draw the  $X$ -graph with your favorite algorithm
- Display the  $Y$ -graphs
  - Matrices
  - Icons
  - Ordinary (node-link) graphs
  - ...

# Design framework

(X,Y)-decomposition



Visualization of the (X,Y)-graph

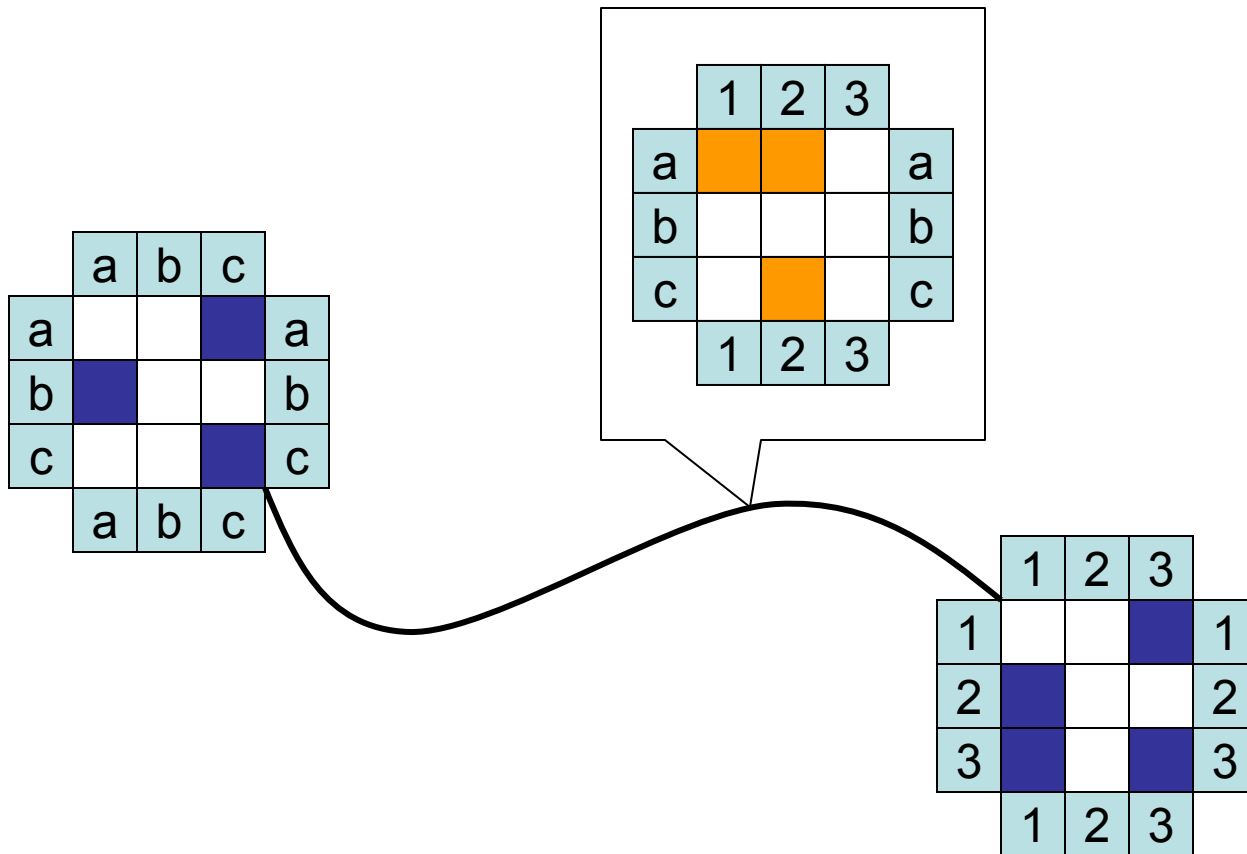


User interaction primitives

# Interaction primitives

- click on clusters to collapse/expand them
- click on edges between clusters to explore the connections
  - a matrix, associated with each edge, shows the connections between vertices of the two clusters (NodeLinkTrix)

# NodeLinkTrix



# ConfluenTrix

