Grouse: Feature-Based, Steerable Graph Hierarchy Exploration

Daniel Archambault[1] Tamara Munzner[1]
David Auber[2]

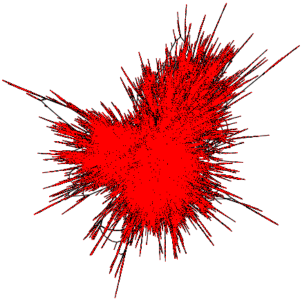
[1]Department of Computer Science, Imager Laboratory
University of British Columbia, Canada

[2]Departement des Informatiques, LaBRI
Université de Bordeaux I, France

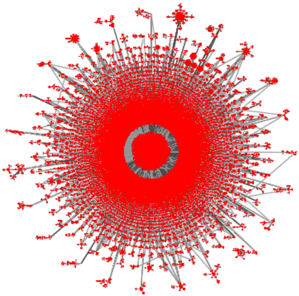
EUROVIS 2007
May 23 - 25
Norrköping, Sweden

Outline

Layout Has High Computational Cost



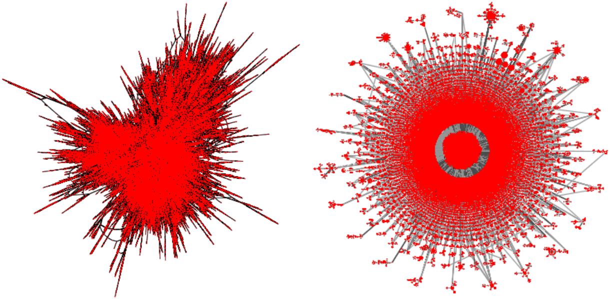
(a) FM^3: 11 min



(b) SPF: 30 min

- Generating full layout has high computational cost

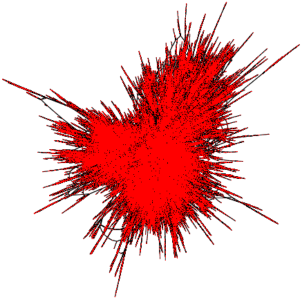
Layout Has High Computational Cost



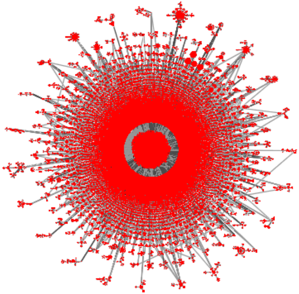
(a) FM^3: 11 min   (b) SPF: 30 min

- Generating full layout has high computational cost
  - most approaches have quadratic running times

Layout Has High Computational Cost



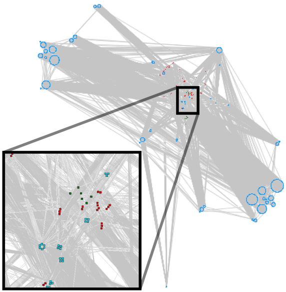
(a) FM^3: 11 min



(b) SPF: 30 min

- Generating full layout has high computational cost
  - most approaches have quadratic running times
- Delays exploration

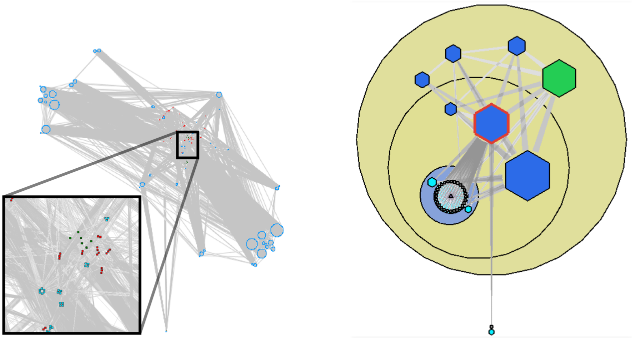
Overwhelming Visual Complexity



(a) TopoLayout

- All nodes and edges drawn: occlusion

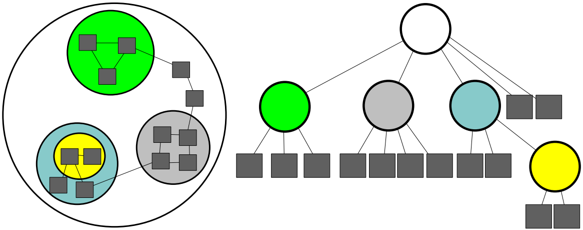
Overwhelming Visual Complexity



(a) TopoLayout (b) Grouse

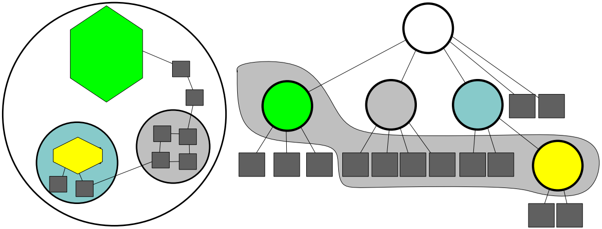
- All nodes and edges drawn: occlusion
- Group subgraphs into a **metanode** to simplify drawing

Multilevel Hierarchy for Abstraction



- A **multilevel hierarchy**: recursive grouping of metanodes
  - **leaves** (squares) are nodes of the input graph
  - **metanodes** (circles) are internal nodes of the hierarchy

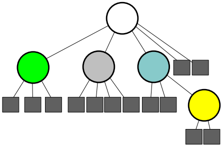
Multilevel Hierarchy for Abstraction: Cut



- A **cut** defines which nodes are visible or hidden
  - nodes on and above the cut are visible in the graph view

Contribution: Steerable, Feature-Based Exploration

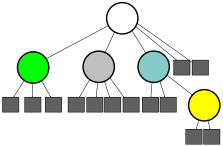
Graph Without Layout +

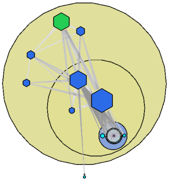


Input

Contribution: Steerable, Feature-Based Exploration

Graph Without Layout +





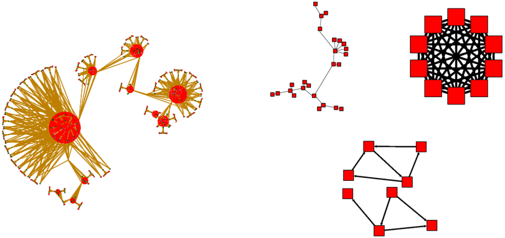
Input

Output

- Advantages
  - exploration can begin immediately
  - uses a **feature-based** hierarchy

Feature-Based Approaches

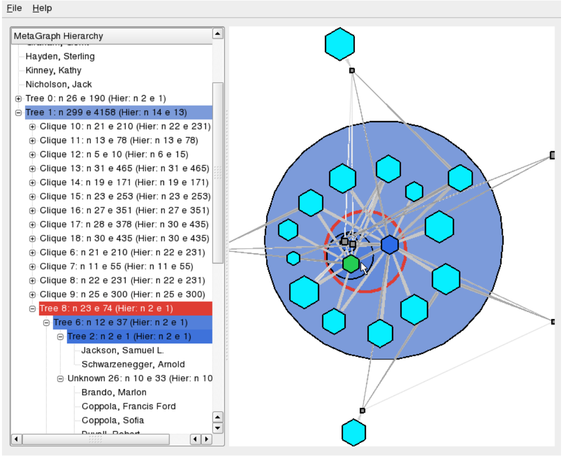
- Layout highlights features of interest in graph



(a) TopoLayout (b) Topological Features

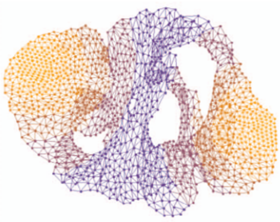
- Grouse uses topology for feature-based hierarchy
  - based on TopoLayout (Archambault *et al.*, 2007)

Video

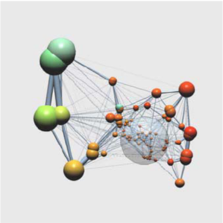

File Help
MetaGraph Hierarchy
Hayden, Sterling
Kinney, Kathy
Nicholson, Jack
Tree 0: n 26 e 190 (Hier: n 2 e 1)
Tree 1: n 299 e 4158 (Hier: n 14 e 13)
Clique 10: n 21 e 210 (Hier: n 22 e 231)
Clique 11: n 13 e 78 (Hier: n 13 e 78)
Clique 12: n 5 e 10 (Hier: n 6 e 15)
Clique 13: n 31 e 465 (Hier: n 31 e 465)
Clique 14: n 19 e 171 (Hier: n 19 e 171)
Clique 15: n 23 e 253 (Hier: n 23 e 253)
Clique 16: n 27 e 351 (Hier: n 27 e 351)
Clique 17: n 28 e 378 (Hier: n 30 e 435)
Clique 18: n 30 e 435 (Hier: n 30 e 435)
Clique 6: n 21 e 210 (Hier: n 22 e 231)
Clique 7: n 11 e 55 (Hier: n 11 e 55)
Clique 8: n 22 e 231 (Hier: n 22 e 231)
Clique 9: n 25 e 300 (Hier: n 25 e 300)
Tree 8: n 23 e 74 (Hier: n 2 e 1)
Tree 6: n 12 e 37 (Hier: n 2 e 1)
Tree 2: n 2 e 1 (Hier: n 2 e 1)
Jackson, Samuel L.
Schwarzenegger, Arnold
Unknown 26: n 10 e 33 (Hier: n 10
Brando, Marlon
Coppola, Francis Ford
Coppola, Sofia

Previous Work: Hierarchy Exploration

- Simplify graph by abstracting subgraphs away



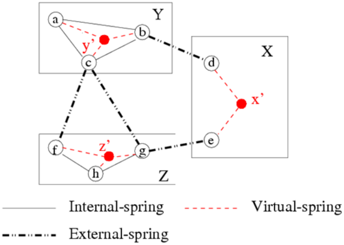
(a) Gansner *et al.* 2004



(b) van Ham and van Wijk 2004

- Advantages and disadvantages
  - reduces graph complexity
  - interaction helps understanding
  - require precomputed layout of entire graph
  - hierarchy not feature-based

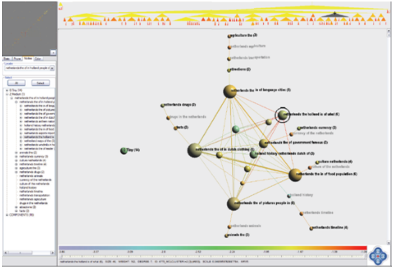
Previous Work, Steerable Exploration: DA-TU



(a) DA-TU, Huang and Eades, 2000

- Explore hierarchy by expanding/contracting metanodes
- Modify hierarchy by selection
- Force directed layout of entire visible graph
  - does not scale to large visible graphs
  - is not feature-based

Previous Work, Steerable Exploration: ASK-GraphView

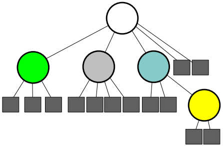


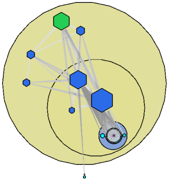
(a) Abello and van Ham, 2006

- Some automated feature-based hierarchy creation
  - modify hierarchy to limit size of subgraph in metanode
- No feature-based layout
- Subgraphs scaled to fit inside metanode

Algorithm: Grouse Approach

Graph Without Layout +



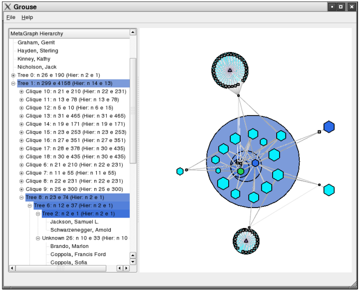


Input

Output

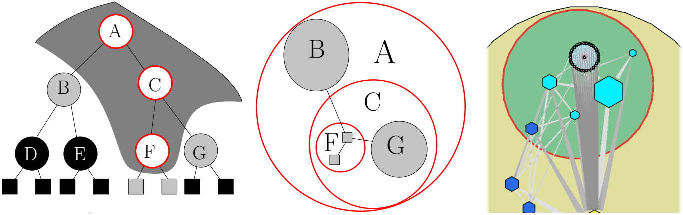
- leaf node is input size
- metanode size estimate is subgraph size
- layout on demand and update metanode sizes

Algorithm: Grouse Interface Overview



- closing a metanode
  - close metanode ↔ save layout and replace by node
- opening a metanode
- combination of open metanode events
  - open all metanodes along a path
  - open all paths below a metanode

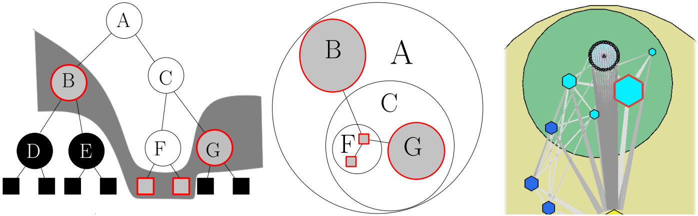
Definitions: Open Metanode



(a) Metanode hierarchy (b) Graph View Sketch (c) Graph View

- **Open** metanode
  - circles containing their subgraph in graph view
  - white in cut diagram

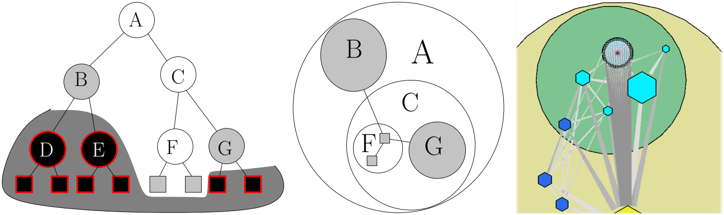
Definitions: Cut Metanode



(a) Metanode hierarchy (b) Graph View Sketch (c) Graph View

- **Cut** metanode
  - hexagon in the graph view
  - grey in cut diagram and graph view sketch

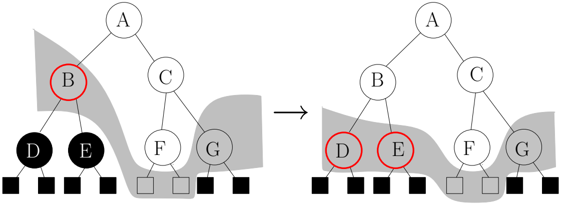
Definitions: Hidden Metanode

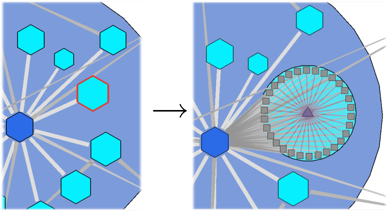


(a) Metanode hierarchy (b) Graph View Sketch (c) Graph View

- **Hidden** metanode
  - not visible in graph view
  - black in cut diagram
  - accessible from list view of hierarchy

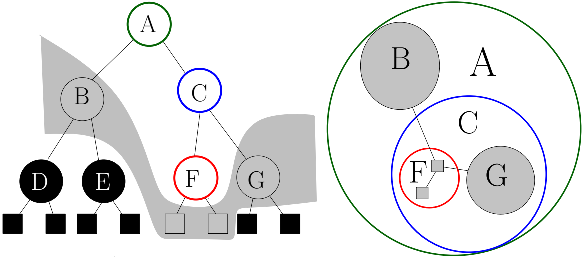
Open Metanode Event





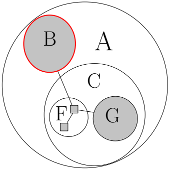
- Animate transition from cut into open metanode

Cascade Relayout



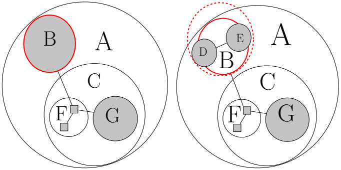
- Relayout along the path in the hierarchy to the root
  - only nodes on path require relayout
  - other nodes may move, but unchanged internally
- Complexity depends on
  - layout algorithm for each node on the path
  - number of nodes on path through hierarchy
  - worst case: $O(d)$ relayouts
    - d maximum hierarchy depth
  - near-balanced hierarchies $O(\log N)$

Cascading Relayout



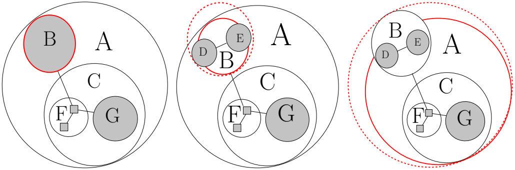
- **(a)** Node B is clicked on to be opened

Cascading Relayout



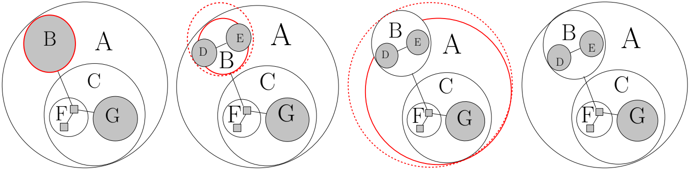
- **(a)** Node B is clicked on to be opened
- **(b)** Subgraph below B is laid out for first time (D and E) and size of B updated

Cascading Relayout



- **(a)** Node `B` is clicked on to be opened
- **(b)** Subgraph below `B` is laid out for first time (`D` and `E`) and size of `B` updated
- **(c)** Subgraph below `A` is laid out (parent of `B`). `C` is not laid out.

Cascading Relayout

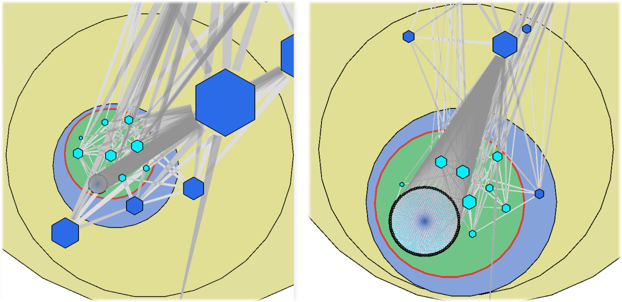


- **(a)** Node B is clicked on to be opened
- **(b)** Subgraph below B is laid out for first time (D and E) and size of B updated
- **(c)** Subgraph below A is laid out (parent of B). C is not laid out.
- **(d)** Final drawing

Layout Algorithms

- Appropriate algorithms used for each topological feature
    - topology unknown: GEM force-directed
- Algorithms applied to minimize node movement when nothing changes
- Save edge and node traversal order
    - for most algorithms this is sufficient
- GEM uses old placement as a starting point
    - future work use dynamic graph drawing approach (Frishman and Tal 2007)

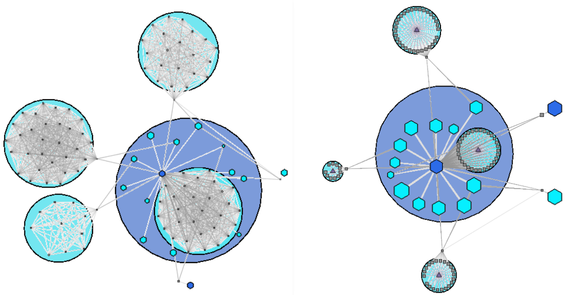
Results: Scale vs. Relayout



(a) Scale (b) Relayout

- Can see more levels of the hierarchy at once
- Larger features given more appropriate space

Results: Force-Directed vs. Feature-Based



(a) Force-Directed (b) Feature-Based

- Different layout algorithms highlight features of interest
- Simpler representation for cliques
  - glyph spoked wheel attached at triangle centre
  - represents $O(N^2)$ edges

Conclusion and Future Work

- Future work
  - attribute data driven features
  - hierarchy modification
- Contributions
  - first steerable, feature-based exploration of graph and associated hierarchy
  - relayout technique
    - more hierarchy levels visible at once
    - features closer to their true size