Grouse: Feature-Based, Steerable Graph Hierarchy Exploration

Daniel Archambault\textsuperscript{1}  Tamara Munzner\textsuperscript{1}  David Auber\textsuperscript{2}

\textsuperscript{1}Department of Computer Science, Imager Laboratory University of British Columbia, Canada

\textsuperscript{2}Departement des Informatiques, LaBRI Université de Bordeaux I, France
Outline

1. Introduction
   Problem Overview
   Approach

2. Previous Work
   Layout Required Beforehand
   Steerable Exploration

3. Algorithm
   Subgraph Layout
   Validation

4. Conclusion
Layout Has High Computational Cost

- Generating full 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

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

(a) $\text{FM}^3$: 11 min
(b) SPF: 30 min
Overwhelming Visual Complexity

(a) TopoLayout

- All nodes and edges drawn: occlusion
Overwhelming Visual Complexity

- 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
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

- 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)
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

- 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

(a) Abello and van Ham, 2006
Algorithm: Grouse Approach

Graph Without Layout +

- 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

- **Open** metanode
  - circles containing their subgraph in graph view
  - white in cut diagram
Definitions: Cut Metanode

- **Cut** metanode
  - hexagon in the graph view
  - grey in cut diagram and graph view sketch
Definitions: Hidden Metanode

- **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

\[ \text{(a) Node } B \text{ is clicked on to be opened} \]
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
(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

- Can see more levels of the hierarchy at once
- Larger features given more appropriate space
Results: Force-Directed vs. 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