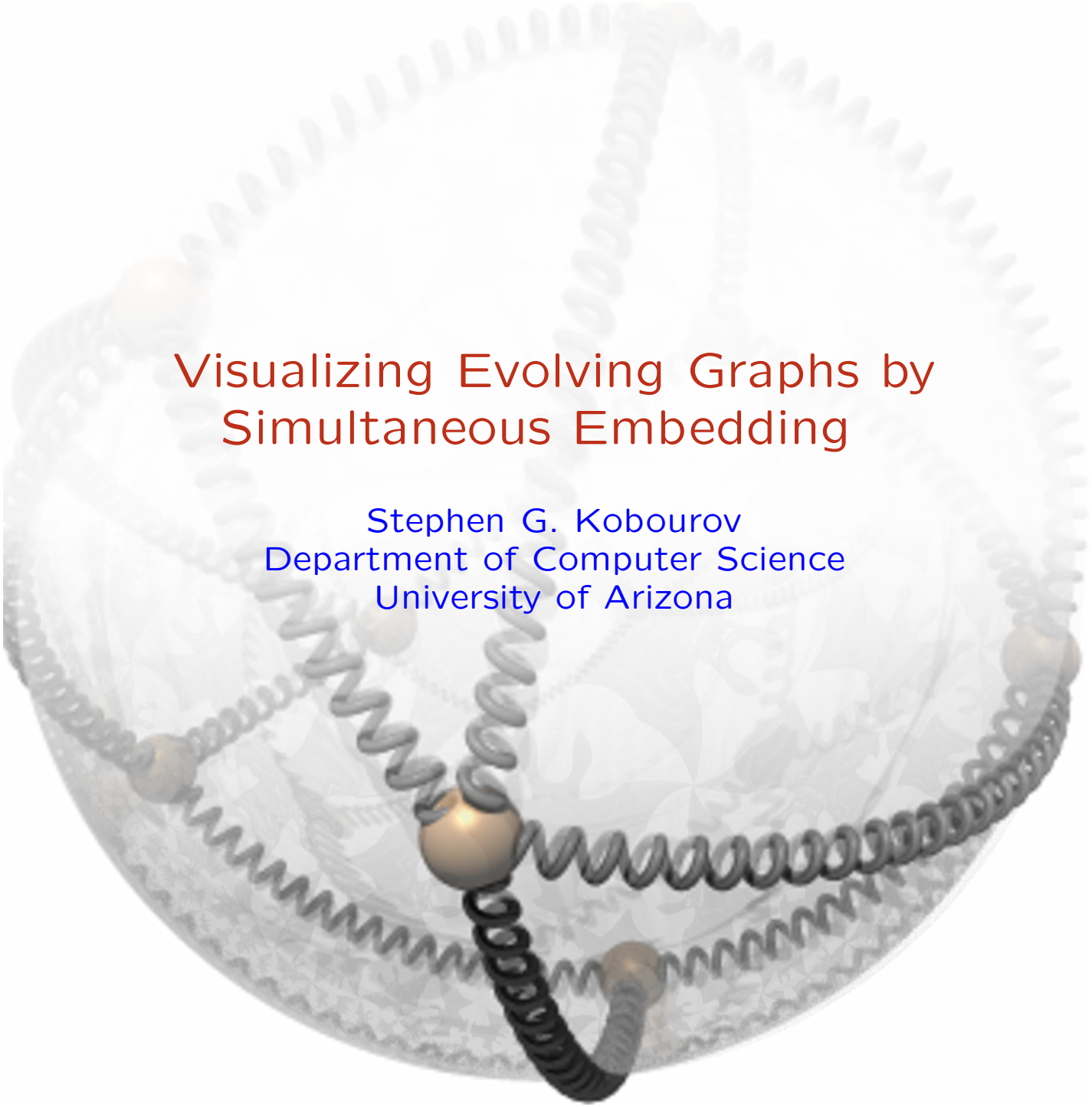
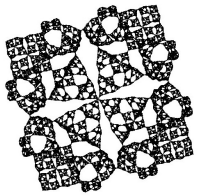
Visualizing Evolving Graphs by Simultaneous Embedding

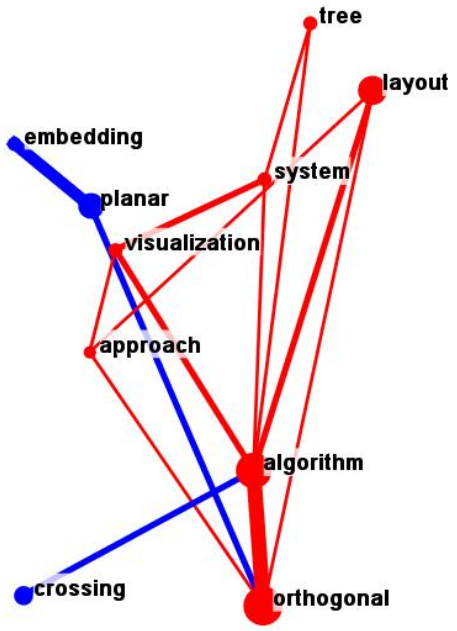
Stephen G. Kobourov
Department of Computer Science
University of Arizona

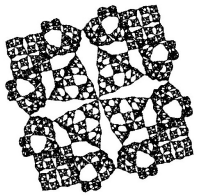




Visualizing Evolving Graphs

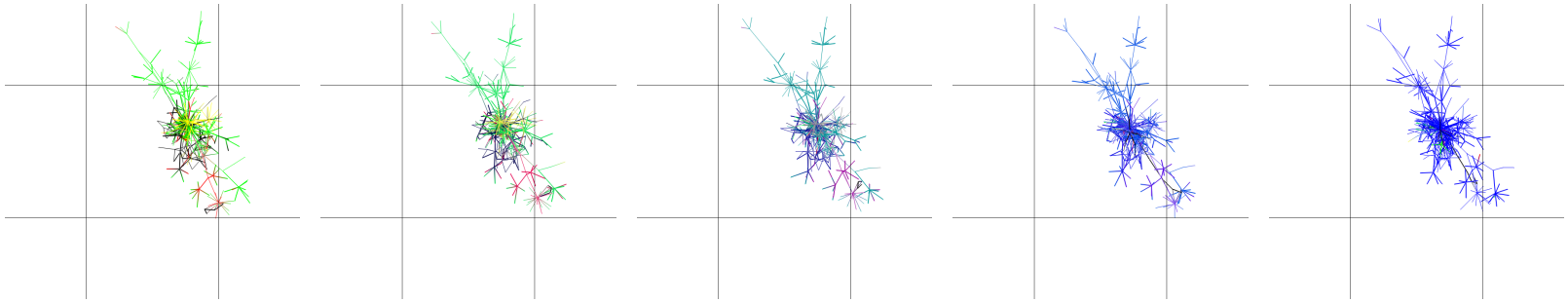
- What is a graph?
  - vertices (objects)
  - edges (relationships)
- Why visualize?
  - helpful in analysis
  - discover patterns, trends
- Dynamic or evolving?
  - objects appear/disappear
  - relationships change
- Applications
  - software design, social networks, ...

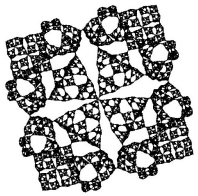




Motivation – Software Development

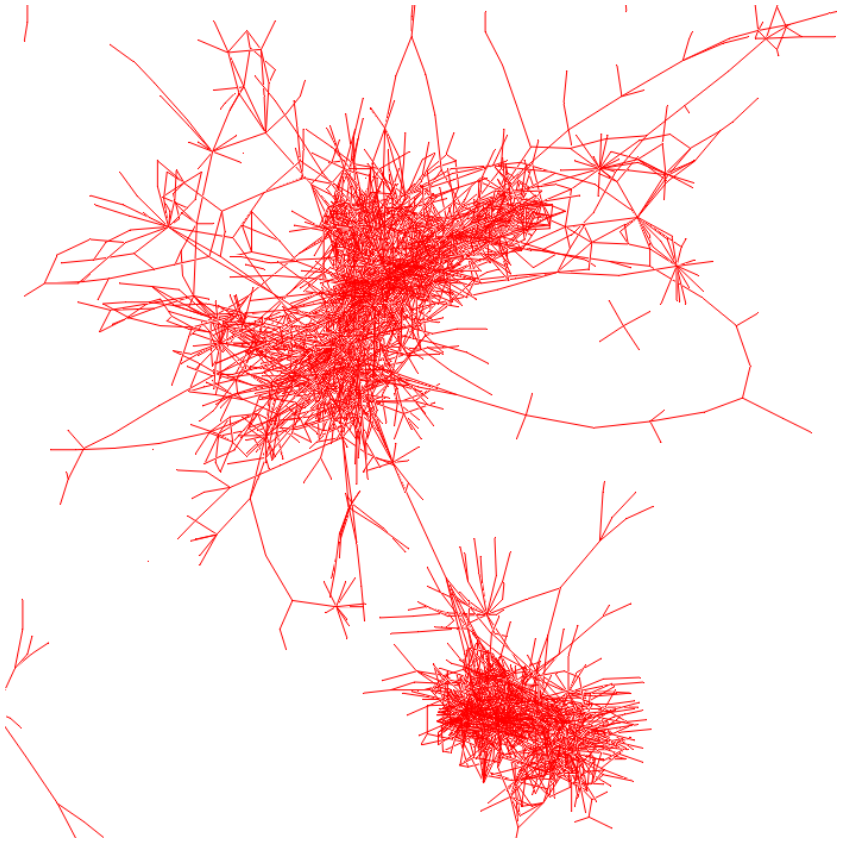
- Evolution of Software
  - why is the program structured that way?
  - who worked on which part and when?
  - which parts of the code are unstable?
- Graphs to Visualize
  - program call graphs
  - method control-flow graphs
  - inheritance graphs

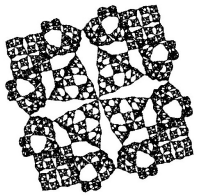




Motivation - Software Development

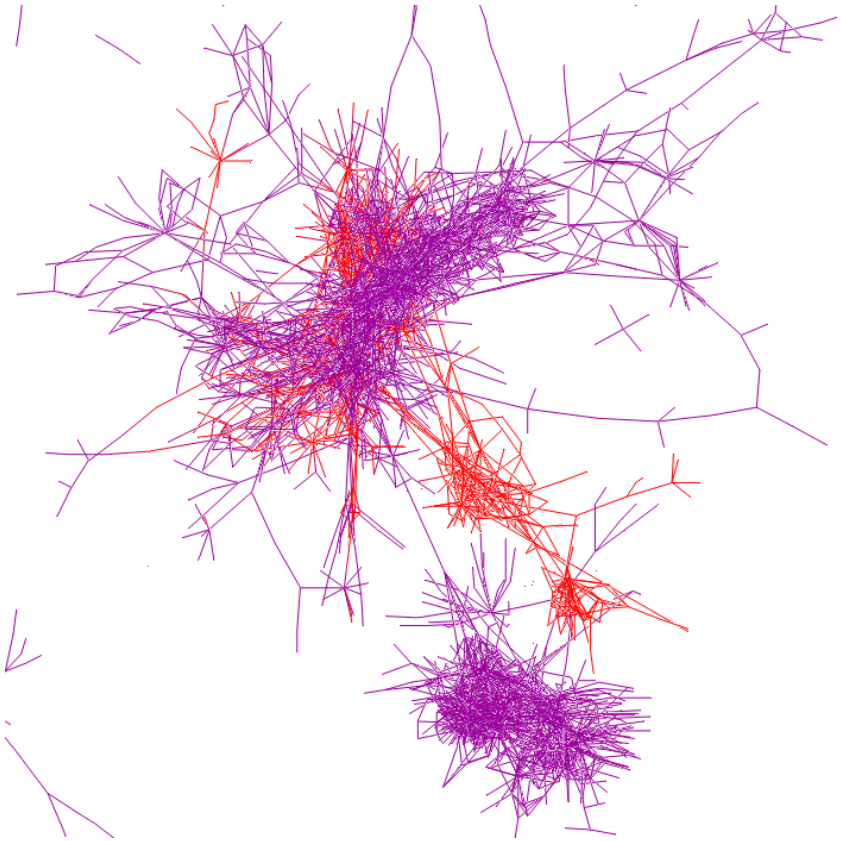
- Call-graph example

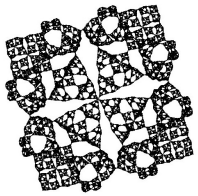




Motivation - Software Development

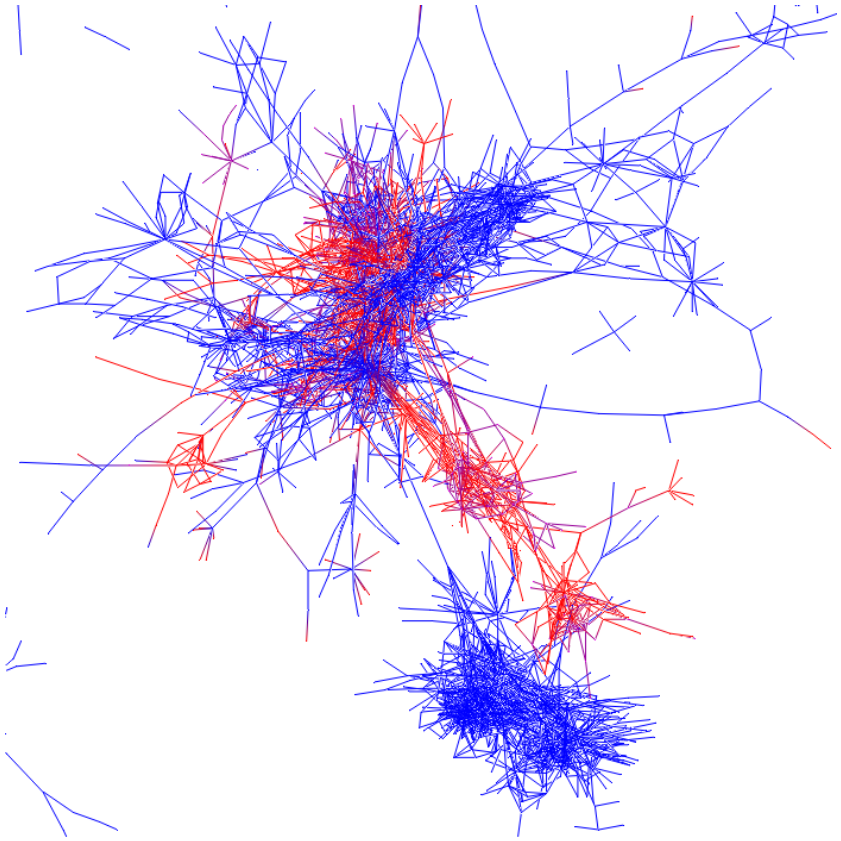
- Call-graph example

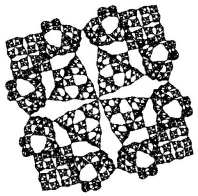




Motivation - Software Development

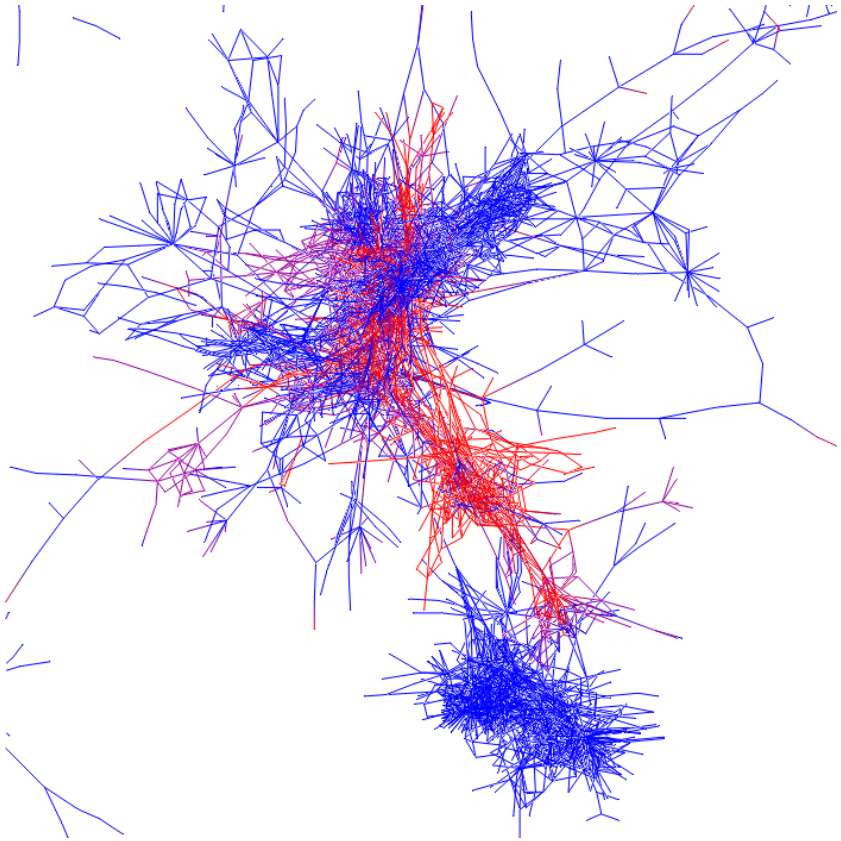
- Call-graph example

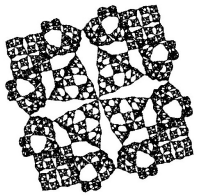




Motivation - Software Development

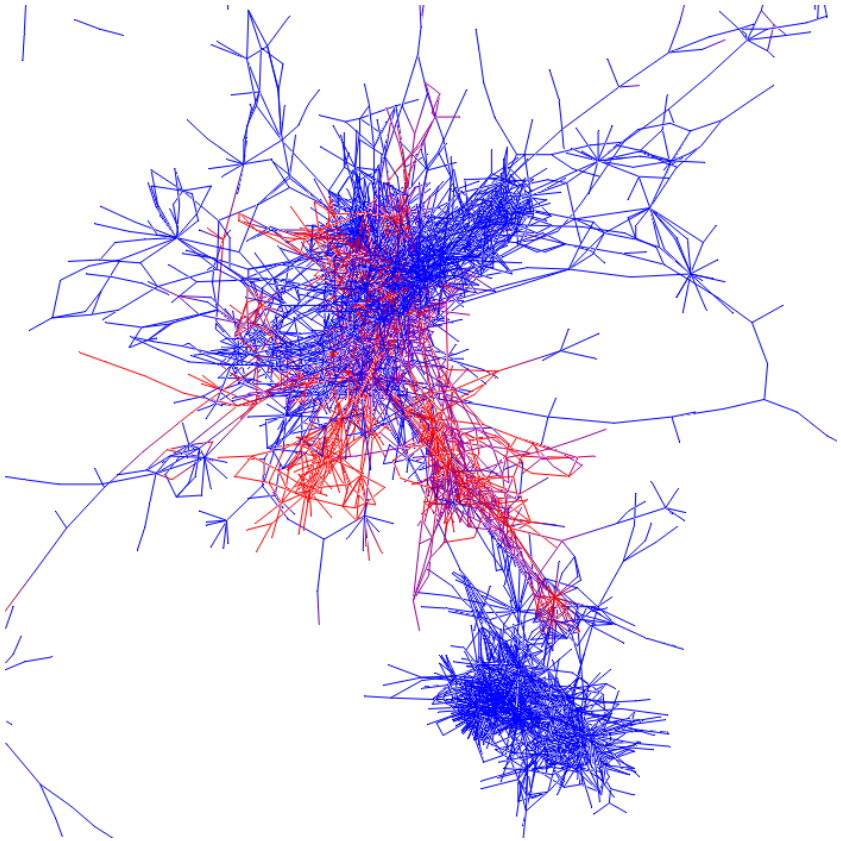
- Call-graph example

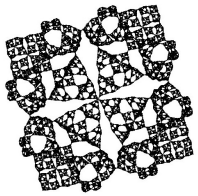




Motivation - Software Development

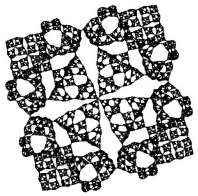
- Call-graph example





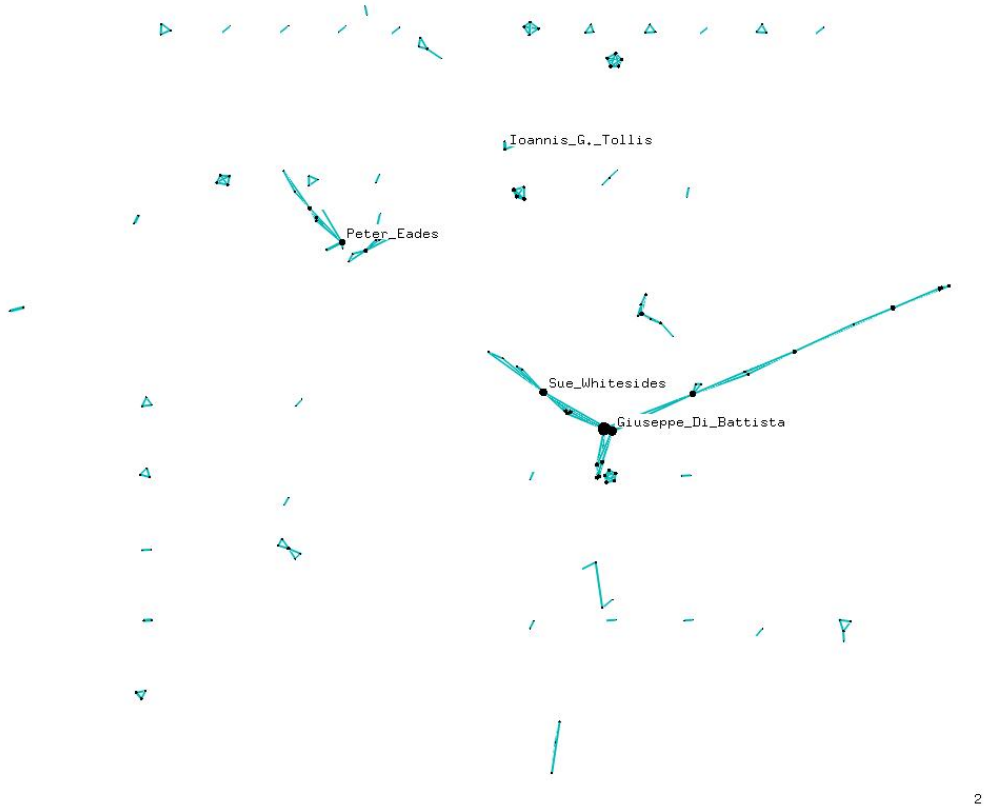
Motivation - Collaboration

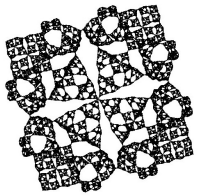
- Collaboration graph example



Motivation - Collaboration

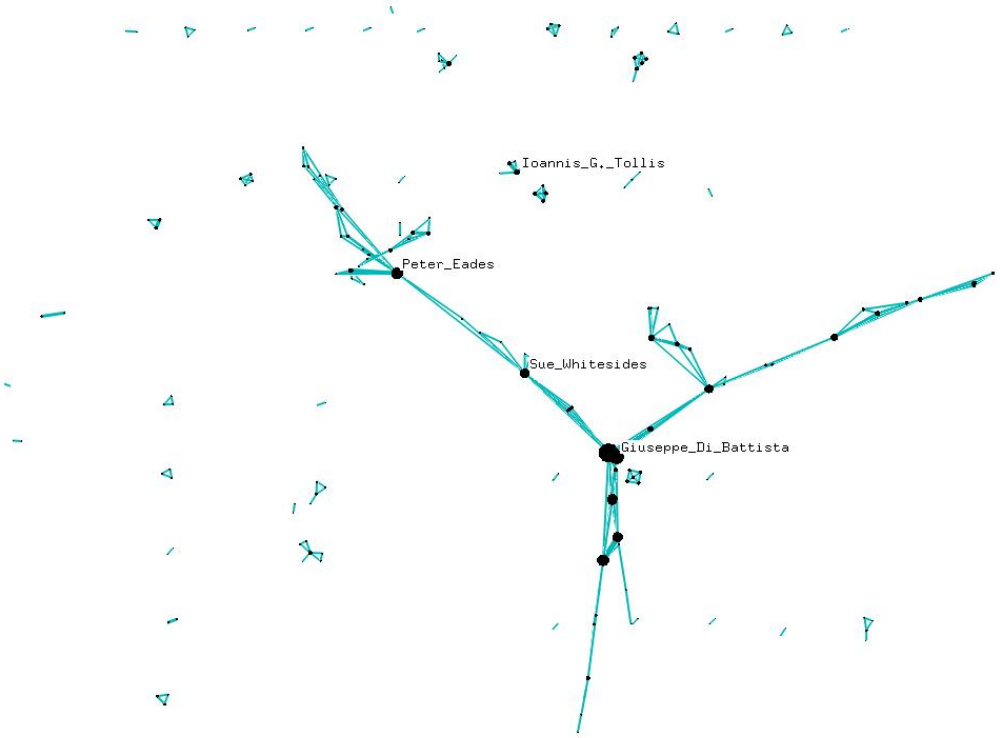
- Collaboration graph example

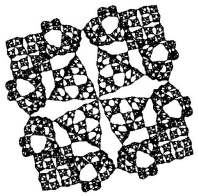




Motivation - Collaboration

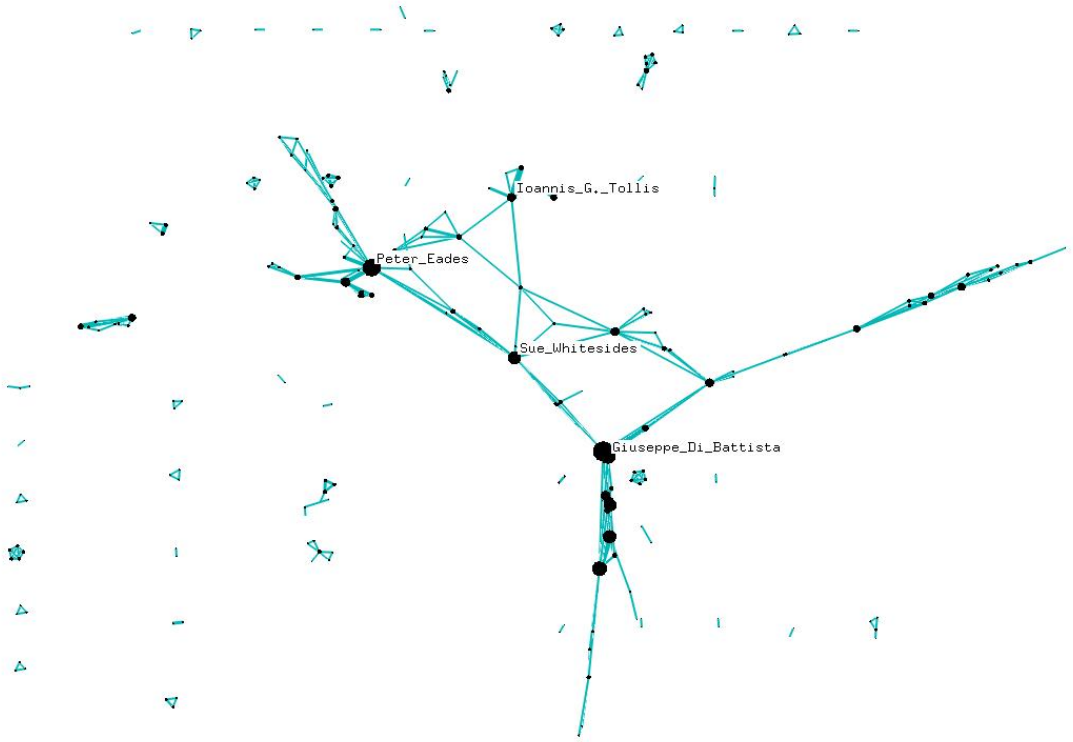
- Collaboration graph example

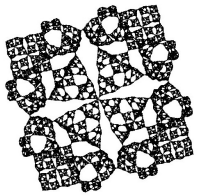




Motivation - Collaboration

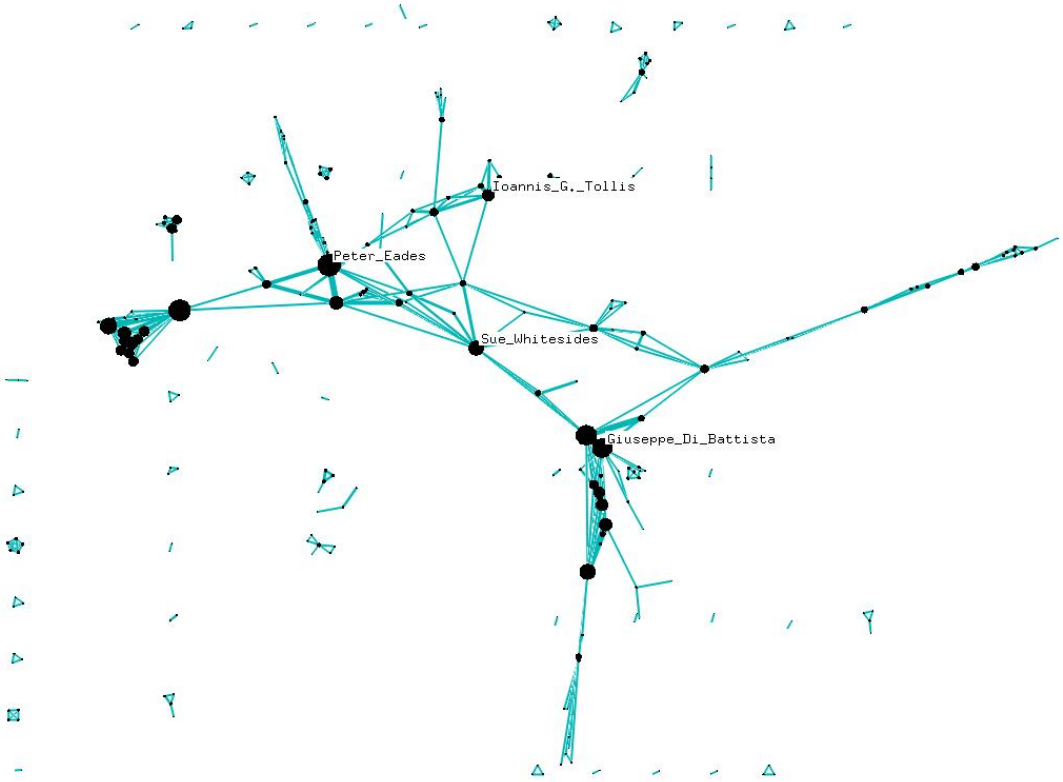
- Collaboration graph example

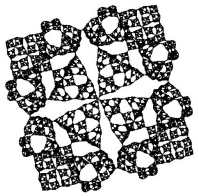




Motivation - Collaboration

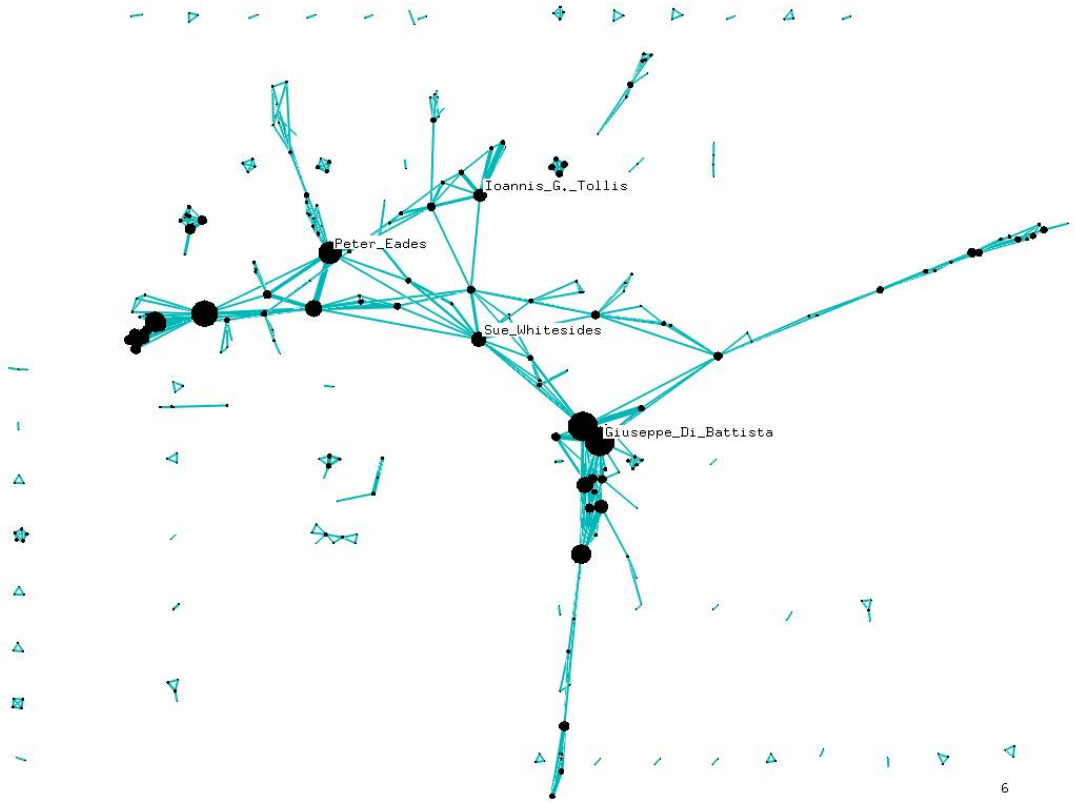
- Collaboration graph example

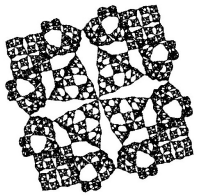




Motivation - Collaboration

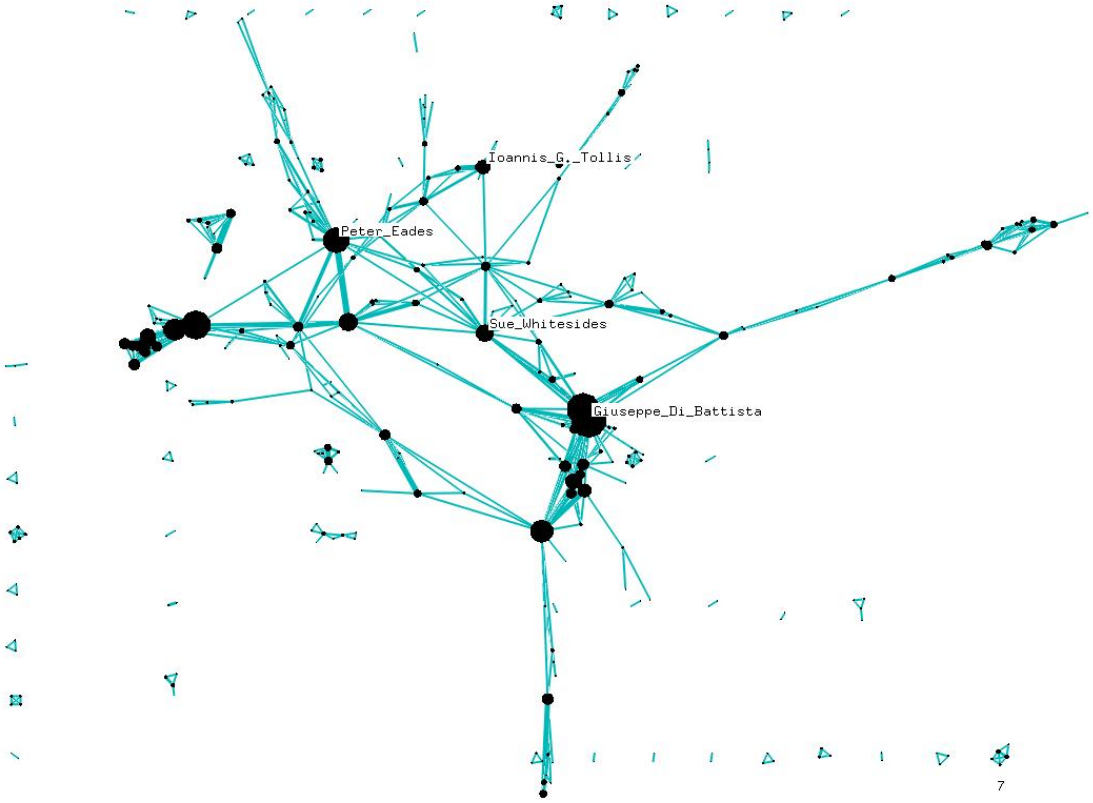
- Collaboration graph example

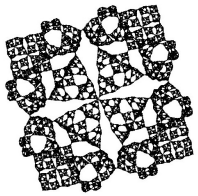




Motivation - Collaboration

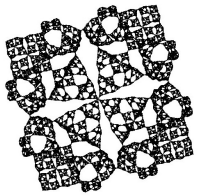
- Collaboration graph example





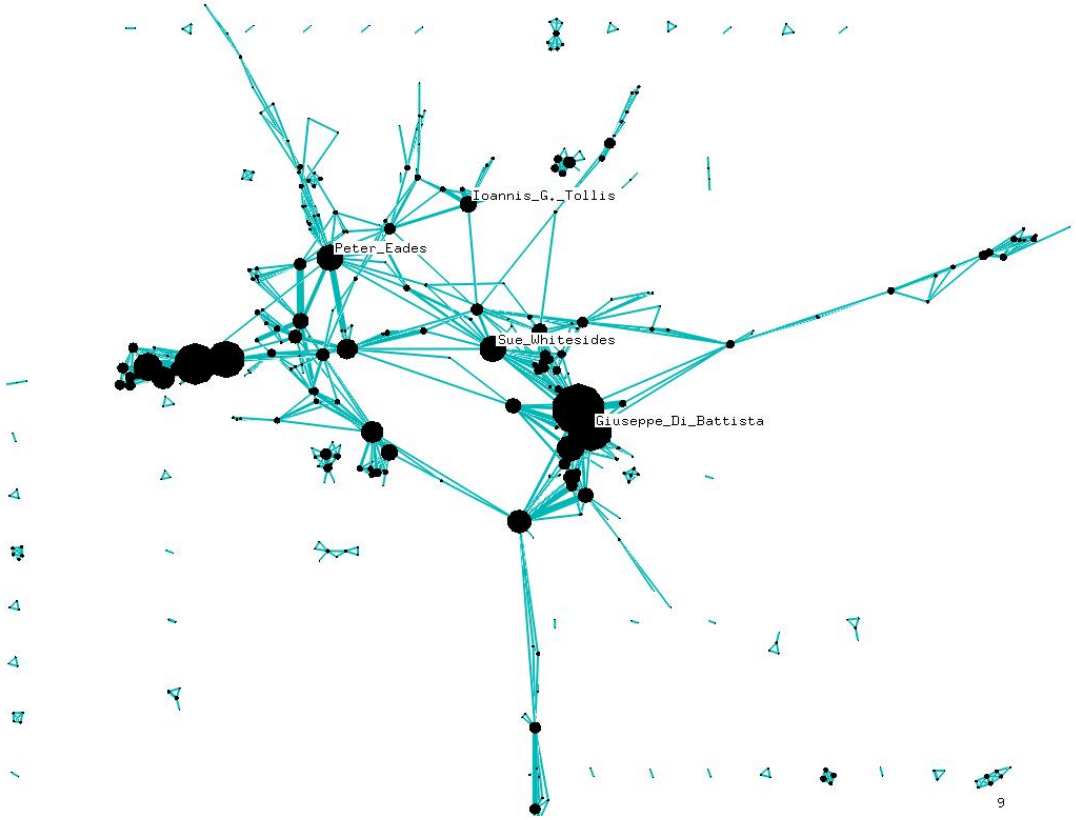
Motivation - Collaboration

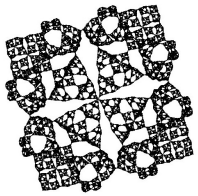
- Collaboration graph example



Motivation - Collaboration

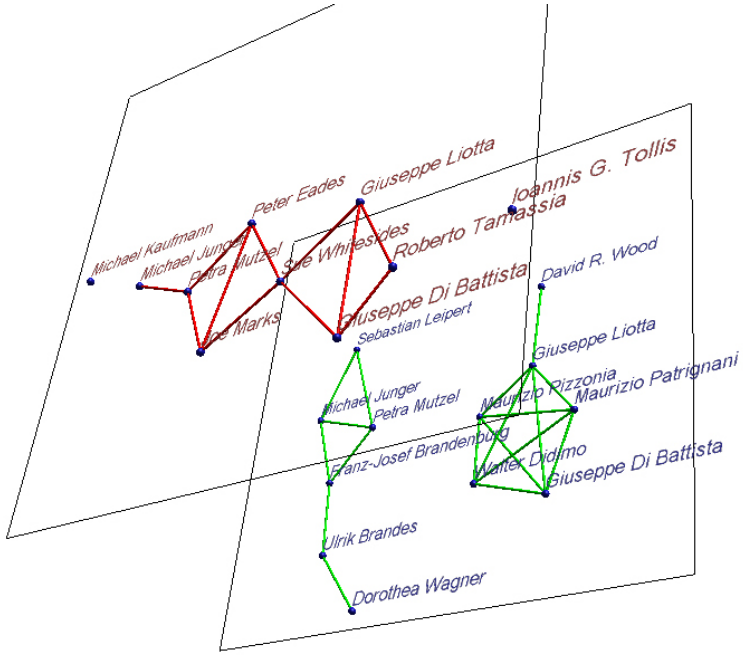
- Collaboration graph example

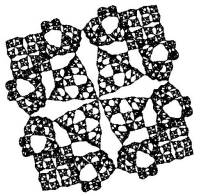




Visualization Wishlist

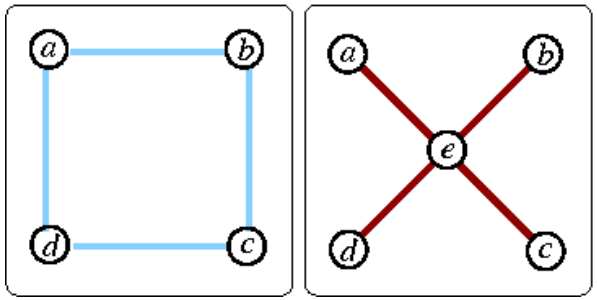
- Each drawing should be readable
  - individually "nice" layouts
  - no crossings, symmetries, etc.
- Mental map preservation
  - structures, relative locations
  - animation, morphing
- In theory ... but in practice...
  - conflicting goals
  - theoretical results
  - practical results

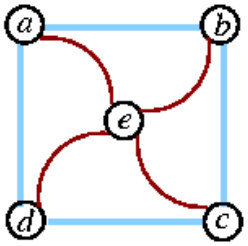


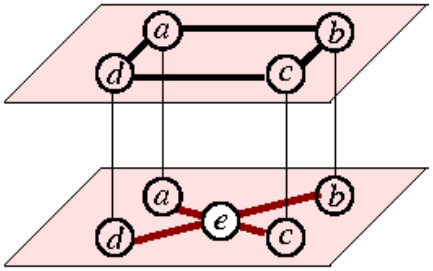


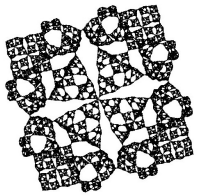
Visualizing Evolving Graphs

- Time model
  - a *timeslice* for each unit of time
  - a snapshot of the graph
- Viewing the evolving graph
  - split-window view
  - aggregate view
  - merged view
  - morphing between timeslices



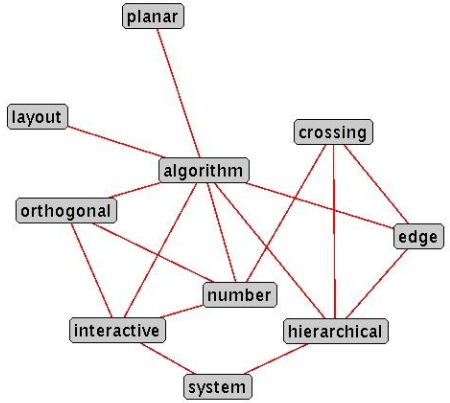


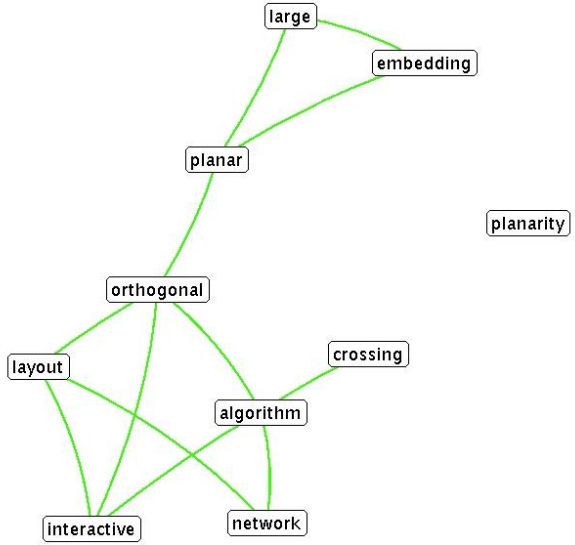


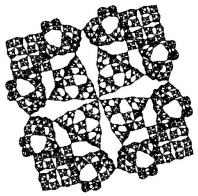


Readability and Mental Map

- Individual layouts for each graph
  - maximizes individual readability
  - no mental map preservation

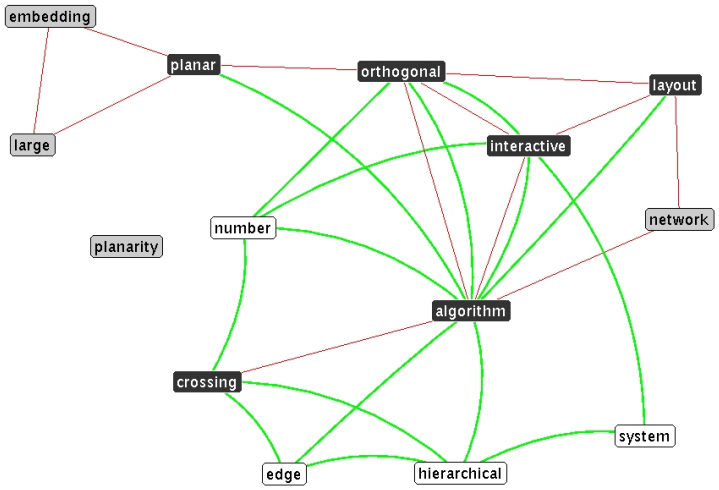


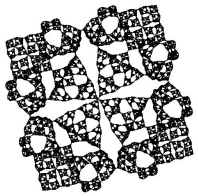




Readability and Mental Map

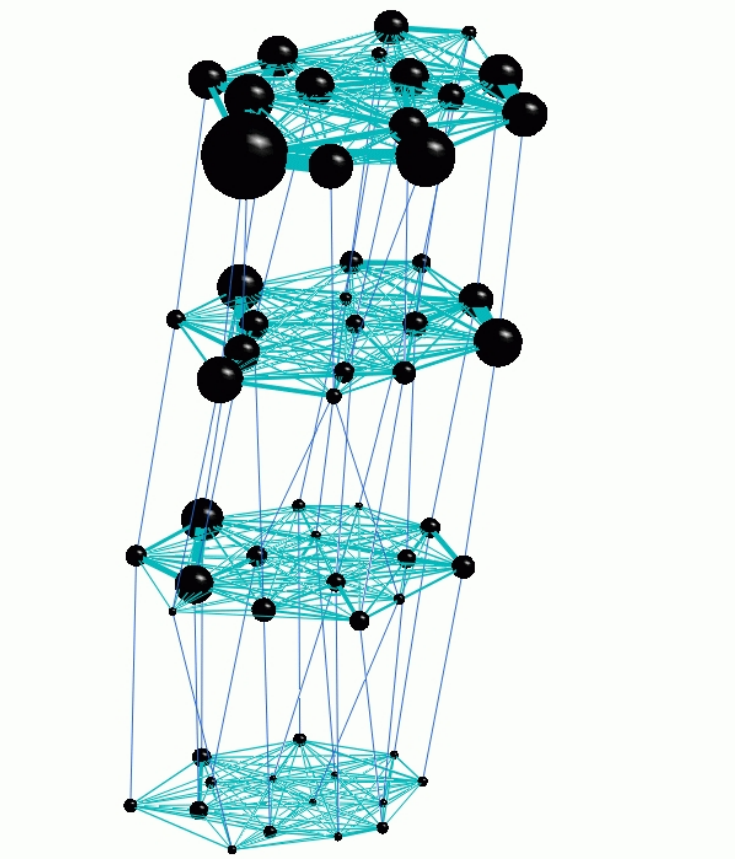
- Individual layouts for each graph
  - maximizes individual readability
  - no mental map preservation
- Simultaneous embedding
  - no individual readability
  - maximizes mental map preservation

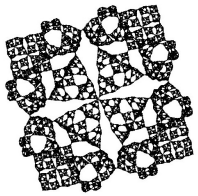




Making It Work

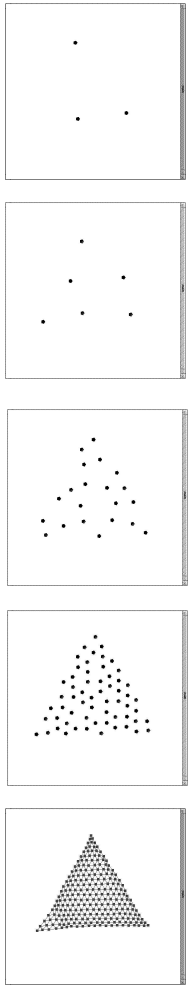
- Force-directed methods
  - modification of Kamada-Kawai
  - modification of Fruchterman-Reingold
- Large graphs become even larger
  - multi-scale methods
  - non-random initial placement
  - high-dimensional embedding
- Readability and mental map
  - enforced via inter-timeslice edges
  - using vertex-weights and edge-weights

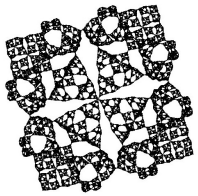




Related Work

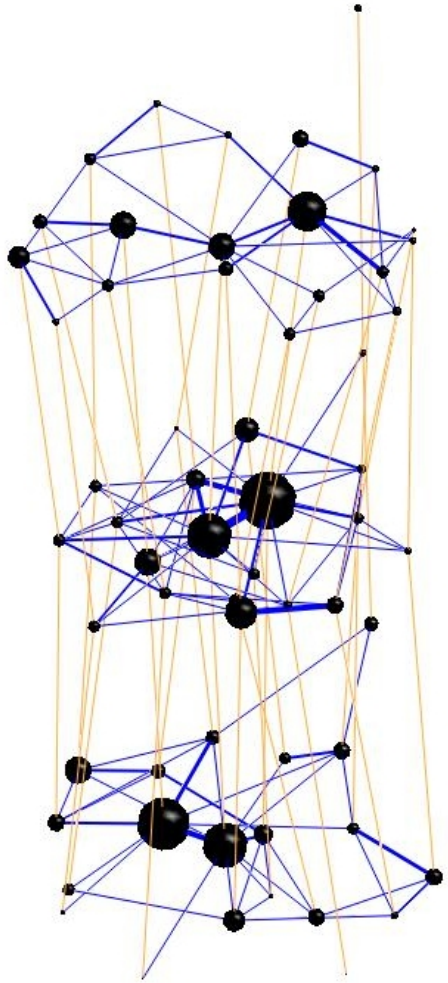
- Force-Directed Methods
  - Fruchterman-Reingold '91
  - Kamada-Kawai '89
- Large Graph Visualization
  - Harel-Koren, Walshaw, '00
- Dynamic Graph Visualization
  - North '96
  - Brandes and Wagner '98
- Mental Map Preservation
  - Eades *et al* '91
  - Misue *et al* '95

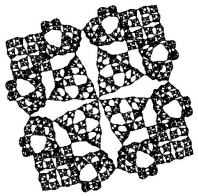




Merged Graph

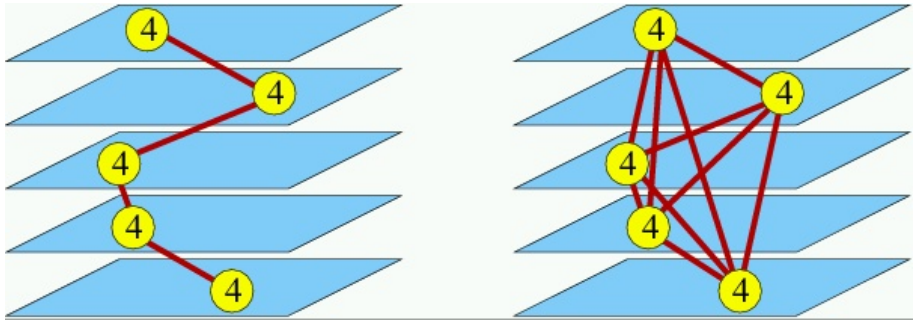
- $G_1, G_2, \ldots, G_k \Rightarrow$ merged graph $\mathcal{G} = (V, E)$
  - $V = V_1 \cup V_2 \cup \ldots \cup V_k$
  - $E = E_1 \cup E_2 \cup \ldots \cup E_k \cup E^*$
  - E^* contains the inter-timeslice edges
- Weights: $w(v)$ and $w(e)$
  - sum of weights in each timeslice
  - or cumulative (collaboration graph)
  - $w(e) = \beta(w(u) + w(v)), e = (u, v) \in E^*$
  - $0 \leq \beta < \infty$: *balance control*



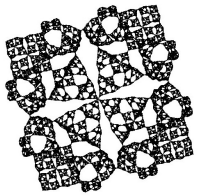


Modified Layout Algorithm

- Inter-timeslice options
  - weight of inter-timeslice edges in E^*
  - connectivity of inter-timeslice edges

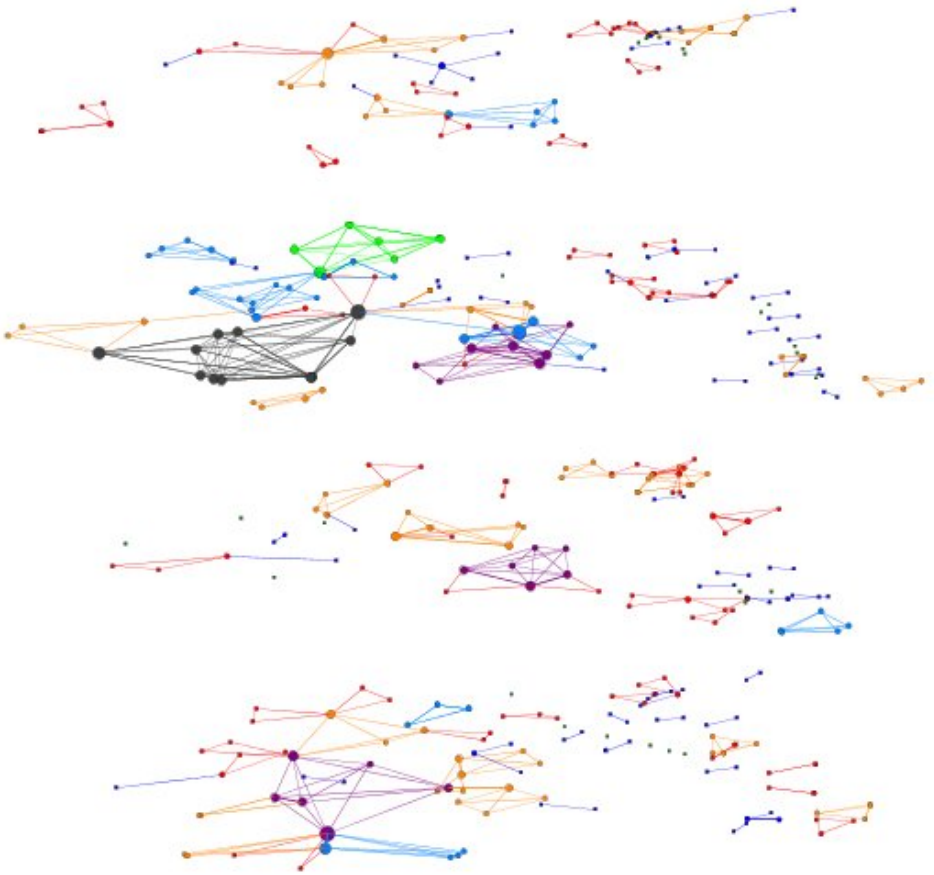


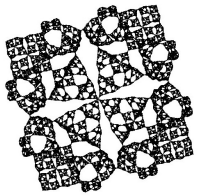
- Role of inter-timeslice edges
  - $w(e) = \beta(w(u) + w(v)), e = (u, v) \in E^*$
  - $\beta \rightarrow 0 \Rightarrow$ good individual layouts
  - $\beta \rightarrow \infty \Rightarrow$ good mental map preservation
- Need to deal with weighted graphs



Modified Layout Algorithm

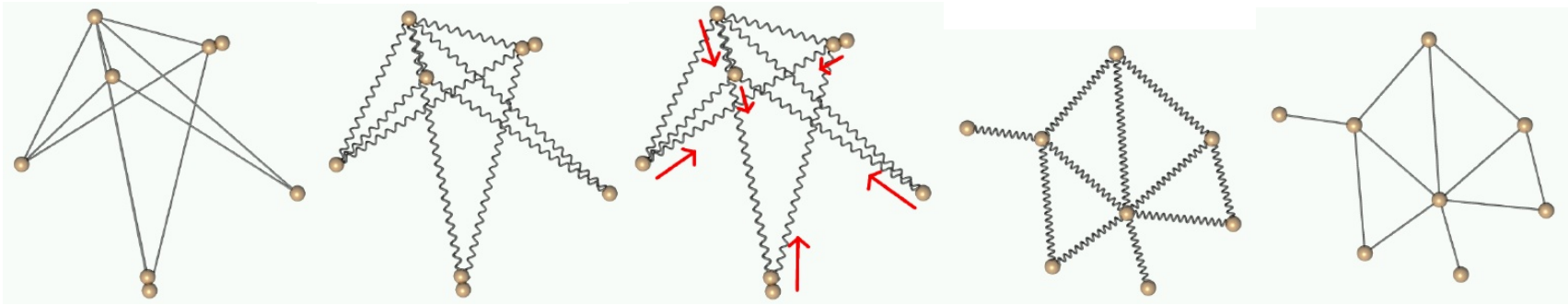
- Heavy vertices: persistent
  - further apart
  - closer to center
  - move little
- Light vertices: transient
  - (dis)appear on the periphery
  - move more
- Heavy edges: persistent
  - shorter in length
- Light edges: transient
  - less important

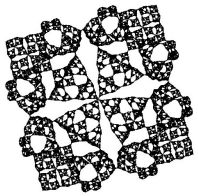




Force-Directed Methods

- Assign energy function to the current layout
  - based on graph distances [KK]
  - based on attractive/repulsive forces [FR]
- Energy model
  - characterizes stability
  - iterative improvement
  - minimal energy $\Rightarrow$ good layout





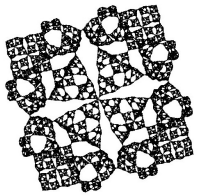
Modified Graph Layout

- Fruchterman-Reingold: $F(v) = F_r(v) + F_a(v)$
  - let d be the distance between two vertices
  - $\kappa = \sqrt{A_{frame}/n}$, ideal edge length
  - repulsive forces:

$$F_r(v) = \sum_{u \in N_i(v)} \frac{\kappa^2}{\|pos[u] - pos[v]\|^2}(pos[u] - pos[v])$$

  - attractive forces:

$$F_a(v) = \sum_{u \in Adj(v)} \frac{\|pos[u] - pos[v]\|^2}{\kappa^2}(pos[u] - pos[v])$$



Modified Graph Layout

- Modified Fruchterman-Reingold: $F_m(v) = F_r(v) + F_a(v)$
  - ideal edge length

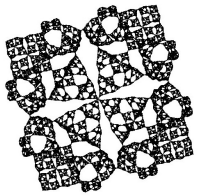
$$\kappa_m = \sqrt{\frac{A_{frame}}{n}} \times \frac{\sqrt{w(u)w(v)}}{w(e)}$$

  - repulsive forces, $\delta_{t_u t_v} = 0/1$ depending on u and v timeslices

$$F_r(v) = \delta_{t_u t_v} \sum_{u \in N_i(v)} \frac{\kappa_m^2}{\|pos[u] - pos[v]\|^2}(pos[u] - pos[v])$$

  - attractive forces

$$F_a(v) = \sum_{u \in Adj(v)} \frac{\|pos[u] - pos[v]\|^2}{\kappa_m^2}(pos[u] - pos[v])$$



Modified Graph Layout

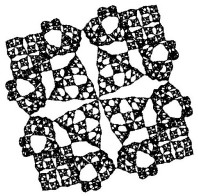
- Kamada-Kawai
  - tries to force Euclidean distances to match graph distances
  - one force vector

$$F(v) = \sum_{u \in N(v)} \left(\frac{\|pos[u] - pos[v]\|^2}{(\kappa \times dist_G(u,v))^2} - 1 \right) (pos[u] - pos[v])$$

- Modified Kamada-Kawai
  - can compute weighted shortest paths
  - or cheat

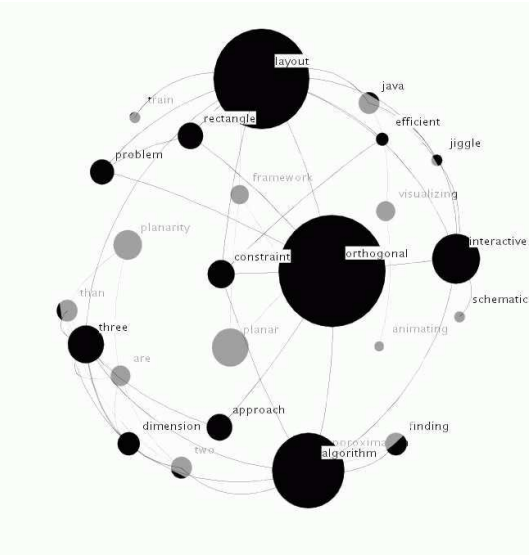
$$dist_G(u,v) = \sum_{i=1}^{n-1} \frac{\sqrt{w_{p_i} \cdot w_{p_{i-1}}}}{w_{e_{p_i p_{i-1}}}}$$

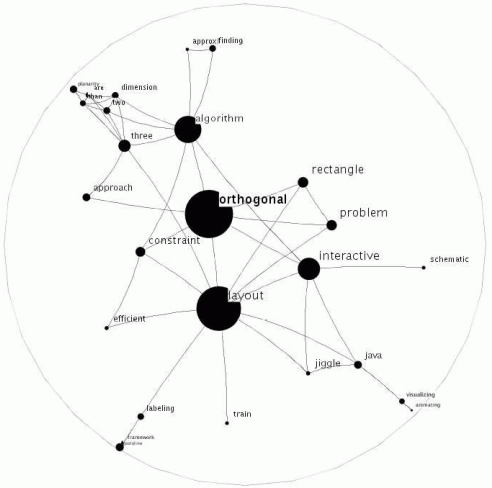
  - $p_1, p_2, \ldots, p_n$ are vertices on a shortest unweighted path $u \rightsquigarrow v$

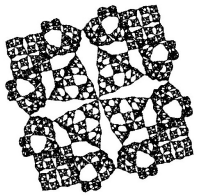


The graphael System

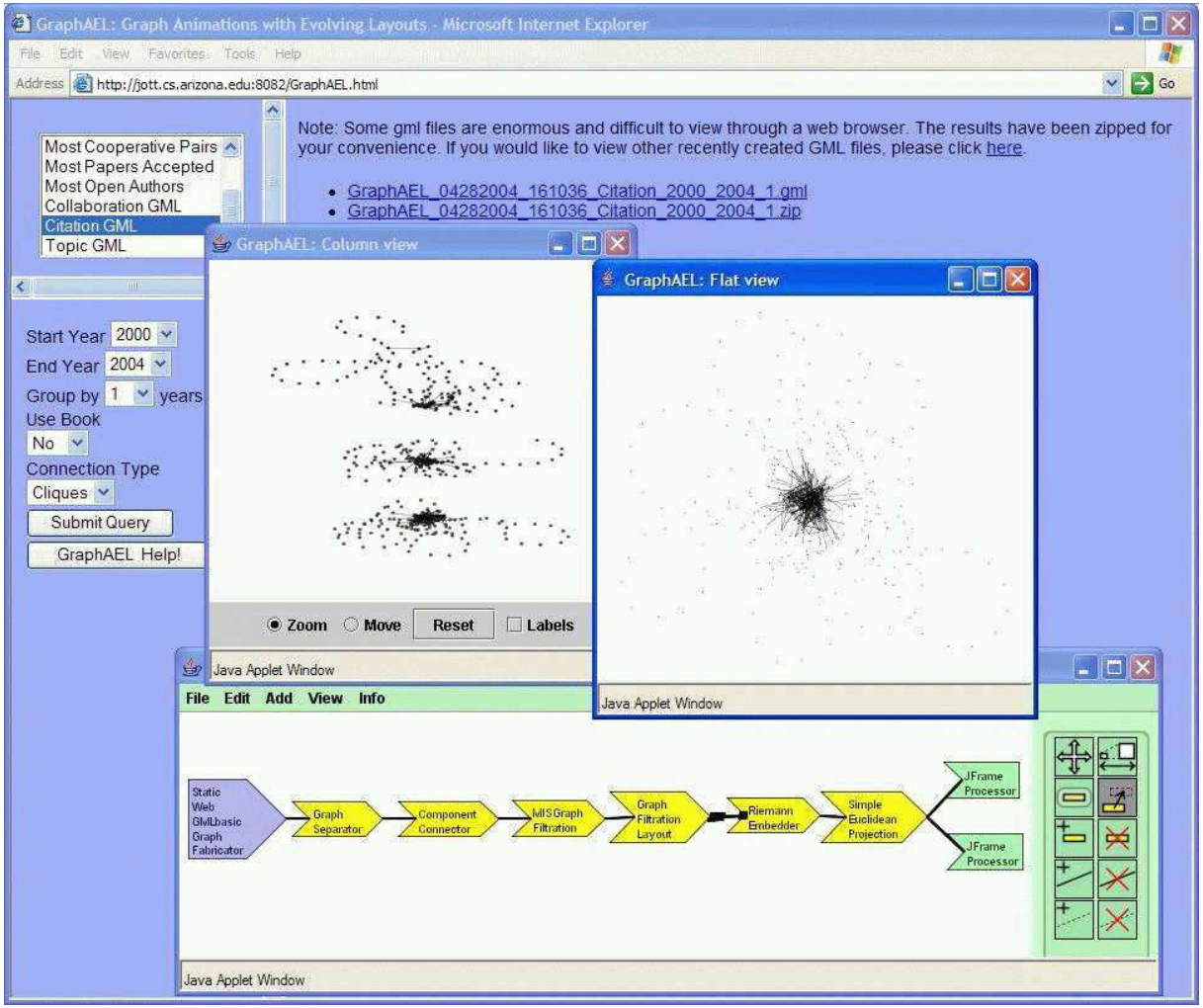
- Yet another graph drawing system
- Features
  - static, evolving graphs, morphing
  - hyperbolic, spherical embedding
  - as a database plugin: ACM, InfoVis, GD
  - user-control over CFG
- Availability
  - `http://graphael.cs.arizona.edu`
  - demo

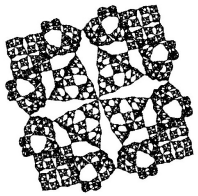






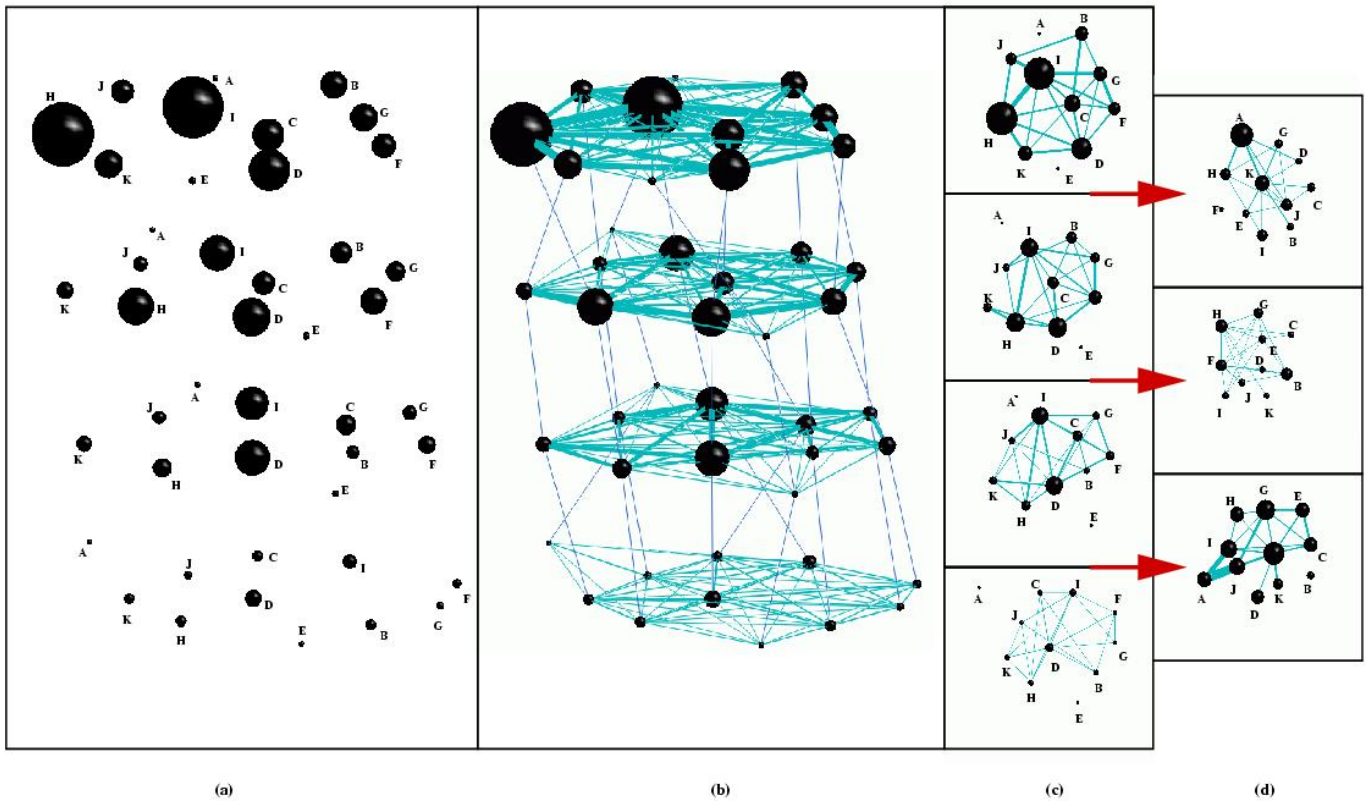
The graphael System

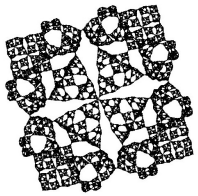




ACM Digital Library: Topics

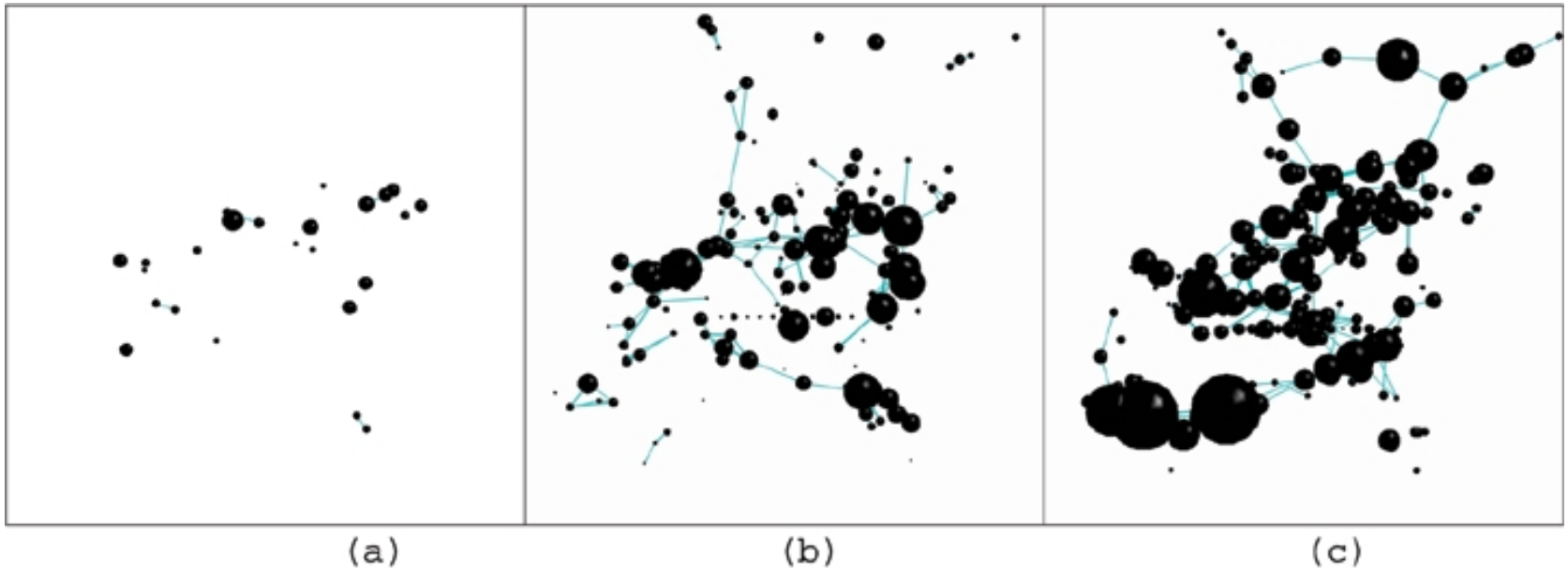
81-85, 86-90, 91-95, 96-00; A:General Literature, B:Hardware, C:Systems Organization, D;Software, E:Data, F:Theory of Computation, G:Mathematics of Computing, H:Information Systems, I:Computing Methodologies, J:Computer Applications, K:Computer Milieux

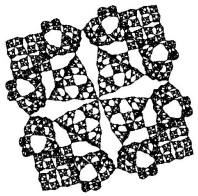




ACM Digital Library: Collaboration

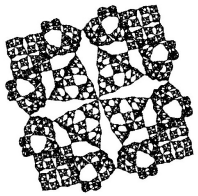
- category H.5: Information Interfaces and Presentation
- collaboration graphs with 3 timeslices
- 86-90, 91-95, 96-00





Conclusions

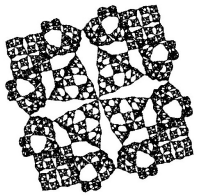
- Practical Results
  - evolving graphs *"Graph Animations with Evolving Layouts" [GD'03]*
  - spring embedders *"Non-Euclidean Spring Embedders" [IEEE InfoVis'04]*
- Theoretical Results
  - geometric embeddings *"Simultaneous Embedding of Planar Graphs" [WADS'03]*
  - relaxations *"Simultaneous Embedding with Few Bends" [GD'04]*
  - thickness *"On Geometric Thickness of Low Degree Graphs" [ACM SCG'04]*
- Applications
  - *"Graph Based Visualization of Software Evolution" [ACM SoftVis'03]*
  - *"Exploring the Computing Literature Using Temporal Graph Visualization" [VDA'04]*



Future Work

- Fly-by compiler/profiler/debugger
  - scalability
  - precomputation
  - clustering
- Simultaneous embedding
  - pairs of trees: simultaneous geometric?
  - pairs of planar graphs: decision algorithm?
  - universal point-sets using 1-bend?
- GD open problem WIKI
  - `http://problems.graphdrawing.org`

    "Open Problems in Graph Drawing" [GD'03]



Acknowledgments

- Cesim Erten (PhD)
- Justin Cappos (PhD)
- Kevin Wampler (MS)
- Chandan Pitta (MS)
- Kelly Hefner (BS)
- Ed Carter (BS)
- Gary Yee (BS)
- Phil Harding (BS)
- Armand Navabi (BS)
- David Forrester (BS)