BGS (Big Graph Surfer): A Large-Scale Graph Visualization Tool

Fangyan Zhang[1]
Song Zhang[1] (Advisor)
Christopher Lightsey[1]
Sarah Harun[1]
Pak Chung Wong[2]

[1]Mississippi State University, Mississippi State, MS, USA
[2]ACT, Iowa City, IA, USA

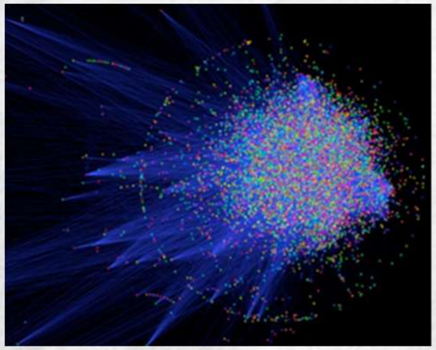
EI 2018, Burlingame, California USA February 27, 2019

Outline

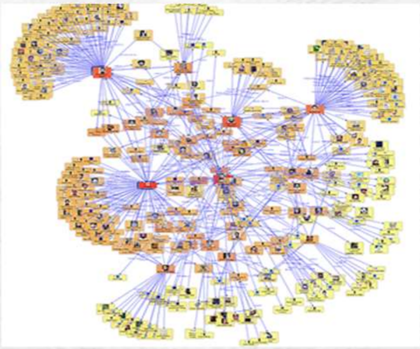
- Introduction
- Related Work
- Methodology
- Case Study
- Discussion

Introduction

– Graphs are widely used to represent a variety of information.



biological network



social network

citation network

……

Introduction

- Graph Visualization

Social Network Visualization

Objective:

How can we help users explore large-scale graphs using graph visual analytics?

Introduction

- Why do we develop a scalable graph visualization tool-BGS?
- Some issues in graph visualization
  - memory issue
  - display issue
  - layout issue
  - interaction issue

Related work

Tools	Representation	Hierarchy	Graph-scale	Pros and cons	Choice
ZAME[1]	Adjacency matrix	Pyramid aggregation	Million scale	Pros: fast rendering using GPU, fit for large graphs Cons: not easy to identify interesting structure	
ASK-GraphView[2]	Node-link	MCL clustering	Million scale	Pros: fit for large graphs, automatically cluster on large graphs Cons: context not helpful in vertical navigation, no crossover edges	✓
TeGViz[3]	Adjacent matrix	No hierarchy	billion scale	Pros: scalable, fit for large graphs Cons: no clustering and hierarchy, only work for random graph, R-MAT/Kronecker	✓
Vizster[4]	Node-link	No hierarchy	Less than million scale	Pros: easy to use, support visual search and analysis Cons: only fit for small graphs	
Network Explorer[5]	Node-link	No hierarchy	Less than million scale	Pros: easy to expand or collapse clusters Cons: not scalable, only fit for small graphs	✓
GraphVizdb[6]	Node-link	Abstraction layers	Million scale	Pros: scalable, support keyword search Cons: no relation between layers, does not support vertical navigation	✓
Matrix Zoom[7]	Matrix view	Clustering with relational constraint	Million scale	Pros: scalable, deal with semi-external graph Cons: limitation in clustering methods	

Related work

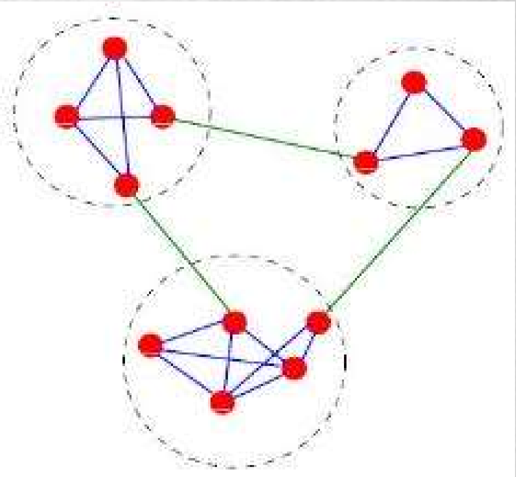
- Pros and cons of hierarchy

	Pros	Cons
Hierarchy	Provide overview for large graphs [2]; Increase visualization efficiency; reduce overlaps in visualization, etc.	Not easy to construct hierarchical structure[7]; easy to lose context while navigating in hierarchy; vertices may be not evenly distributed in hierarchy, etc.

- Balance Pros and Cons

Related work

- Divisive algorithms
  - which work from top to bottom by detecting inter-cluster links and removing them recursively.
  - Newman clustering algorithm[8] O ($|V|*|E|$)
- Agglomerative algorithms
  - which start from its own singleton cluster, and merge similar clusters recursively.
  - MCL clustering algorithm[9] $O(|V|^3)$
- Optimization algorithms
  - These algorithms usually use a modularity value as an object function to measure the quality of clustering. They adjust clusters in each step trying to increase modularity values as high as possible.
  - Louvain clustering algorithm[10] O ($|V|$)



Related work

- Louvain Clustering
  - Modularity indicates the density of links within clusters as compared to links between clusters

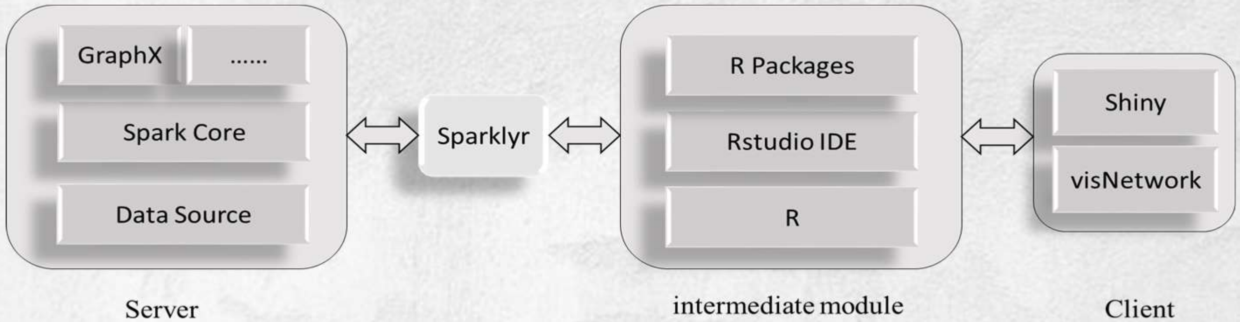
Modularity value: $Q = \frac{1}{2m}\sum_{i,j}\left[A_{i,j} - \frac{k_i k_j}{2m}\right]\delta(c_i, c_j) = \frac{1}{2m}\sum_c[\sum in - \frac{(\sum tot)^2}{2m}]$

  - $A_{i,j}$: edge weights between *i* and *j*.
  - k_i : sum of edge weights that come from or go to vertex *i*.
  - m : $\frac{1}{2}\sum_{i,j} A_{i,j}$
  - $\delta(c_i, c_j)$: 1 while vertex *i* and vertex *j* belong to the same cluster, 0 otherwise.
  - $\sum in$: sum of weights of edges within cluster *c*
  - $\sum tot$: sum of weights of edges of whole cluster *c*.

$$\Delta Q = \left[\frac{\sum in + k_{i,in}}{2m} - (\frac{\sum tot + k_i}{2m})^2\right] - \left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

Methodology: Architecture & Layout

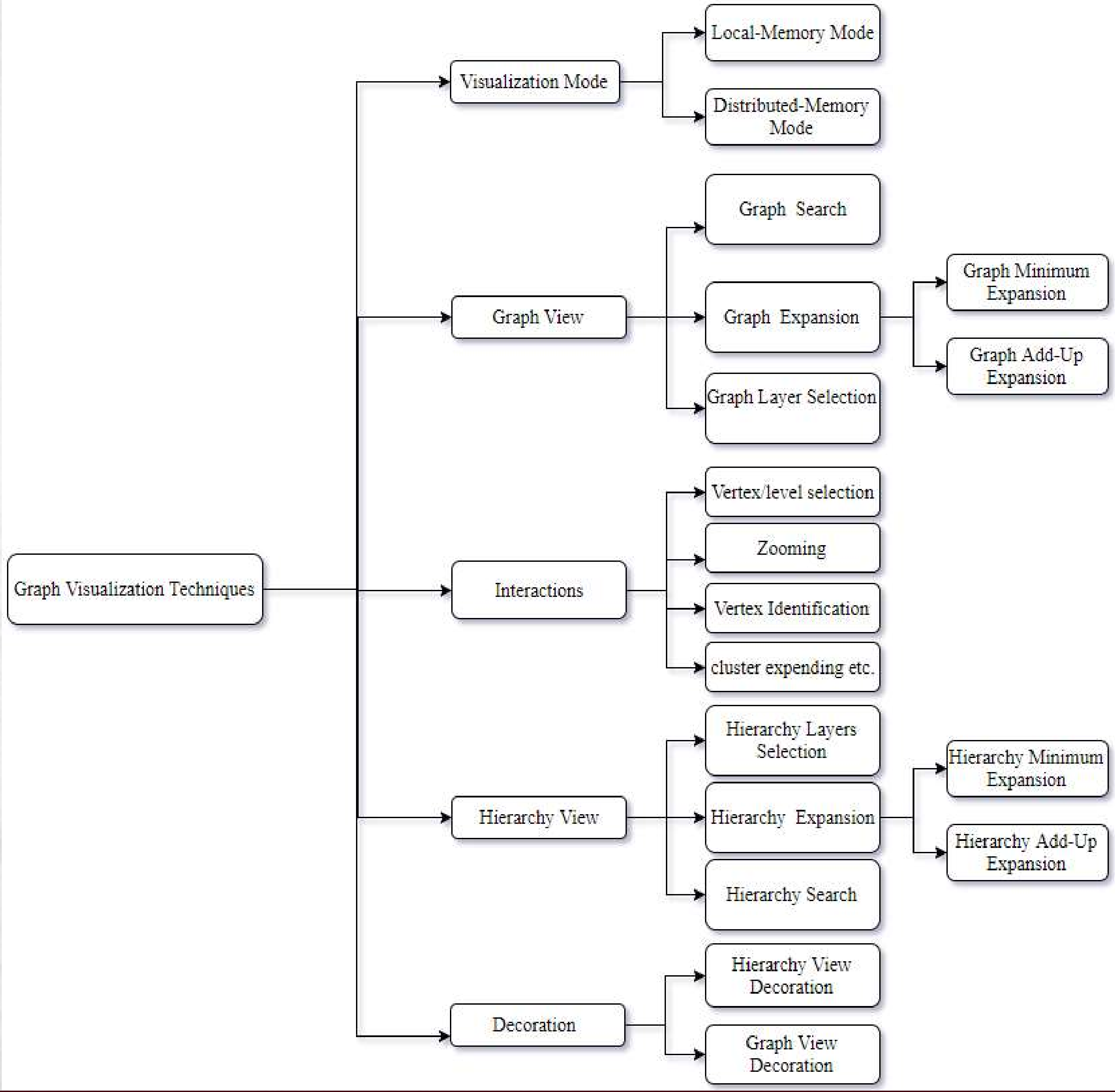
- Architecture



- Layout

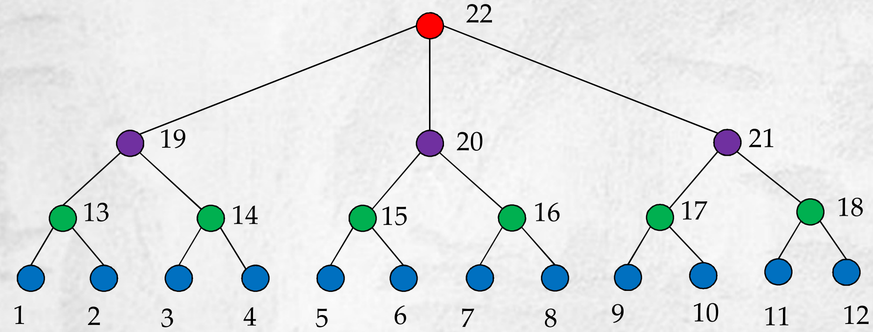
  Thirteen graph layouts (iGraph)

  Real-time computation

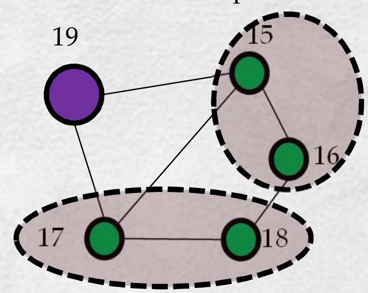

Graph Visualization Techniques
Visualization Mode
Local-Memory Mode
Distributed-Memory Mode
Graph View
Graph Search
Graph Expansion
Graph Minimum Expansion
Graph Add-Up Expansion
Graph Layer Selection
Interactions
Vertex/level selection
Zooming
Vertex Identification
cluster expending etc.
Hierarchy View
Hierarchy Layers Selection
Hierarchy Expansion
Hierarchy Minimum Expansion
Hierarchy Add-Up Expansion
Hierarchy Search
Decoration
Hierarchy View Decoration
Graph View Decoration

Methodology: Hierarchy View and Graph View

- Hierarchy View and Graph View



Hierarchy View



Graph View

Methodology: Expansion Mode

Minimum mode

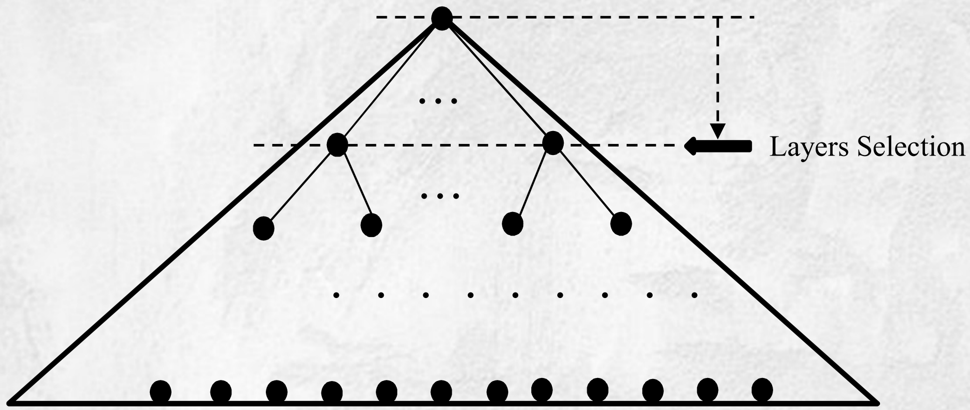
- The previously expanded clusters or nodes will be automatically collapsed into a cluster.
- This mode maintains high efficiency in large-scale graph visualization.

Add-Up mode

- The previously expanded clusters or nodes will be preserved.
- We can observe detailed relations among multiple clusters or nodes.

Methodology: Hierarchy Exploration

- Hierarchy Layers Selection

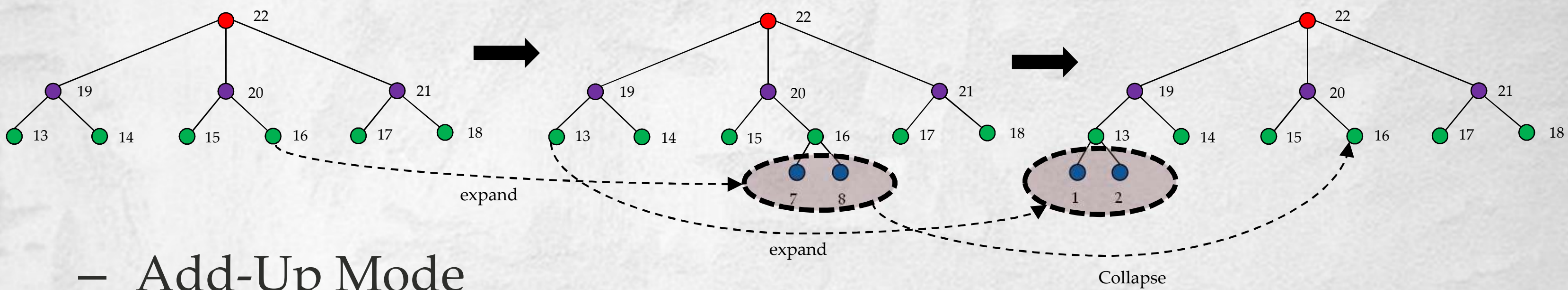


- If one hierarchy has depth *h*, and the initial hierarchy has *s* layers, then the initial hierarchy is $\{Ti,\ h\text{-}s+1 < i <= h\}$ which provides informative context for users to explore the graph hierarchy.
- The several top levels in the hierarchy will consistently exist with expanding clusters.

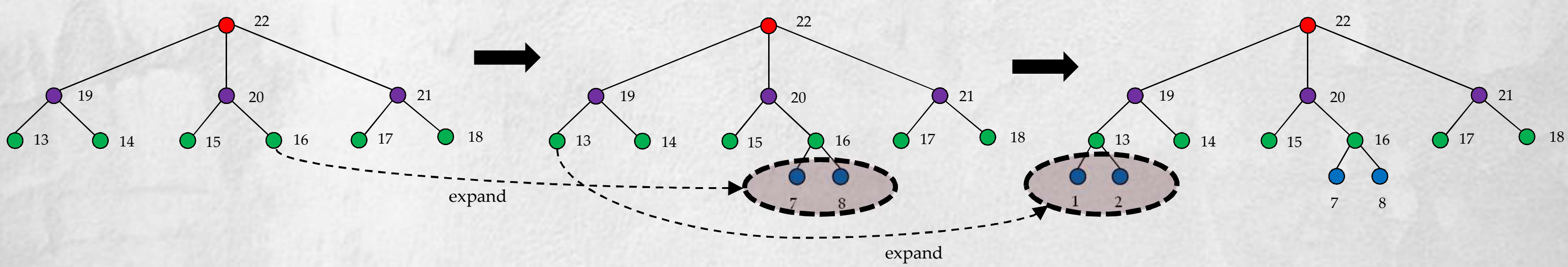
Methodology : Hierarchy Exploration

- Hierarchy Expansion
  - Minimum Mode

Note: Hierarchy Layers Selection = 3



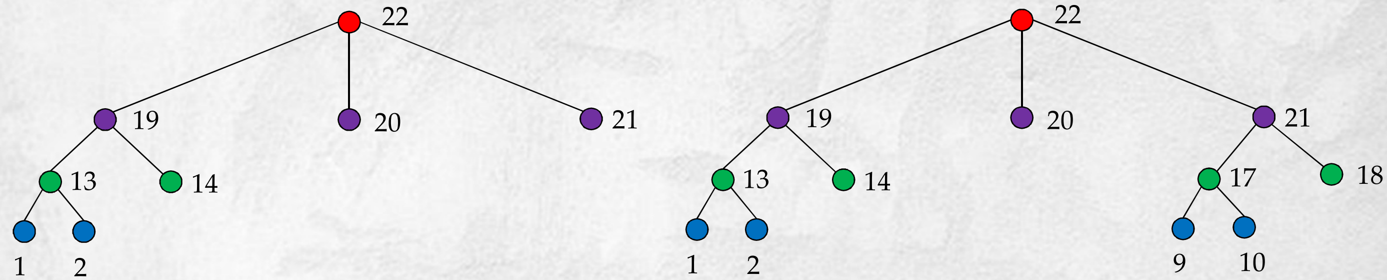
  - Add-Up Mode



Methodology: Hierarchy Exploration

- Hierarchy Search

Note: Hierarchy Layers Selection = 2

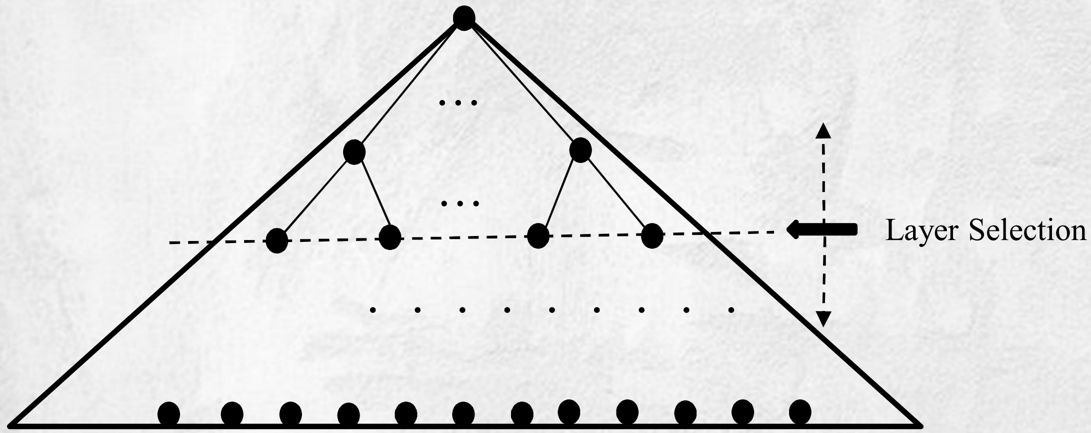


Minimum mode

Add-Up mode

Visualization: Graph Exploration

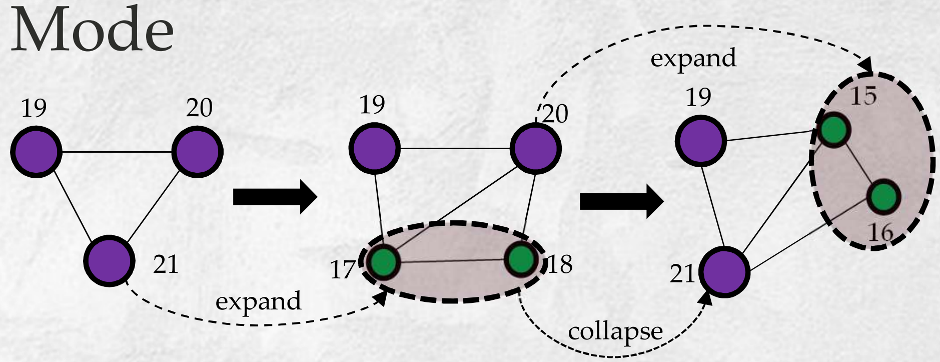
- Graph Layer Selection



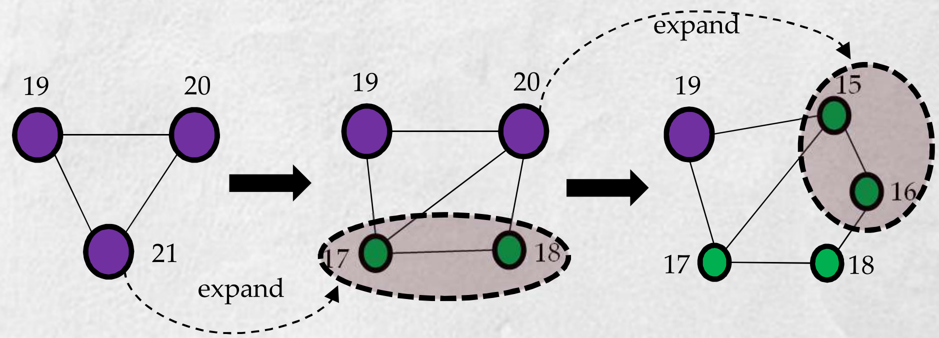
- Initially, BGS visualizes the top layer graph Gh (h is the depth of the hierarchy) in graph view.
- Users are permitted to select another starting layer Gi to visualize.

Methodology: Graph Exploration

- Graph Expansion
  - Minimum Mode

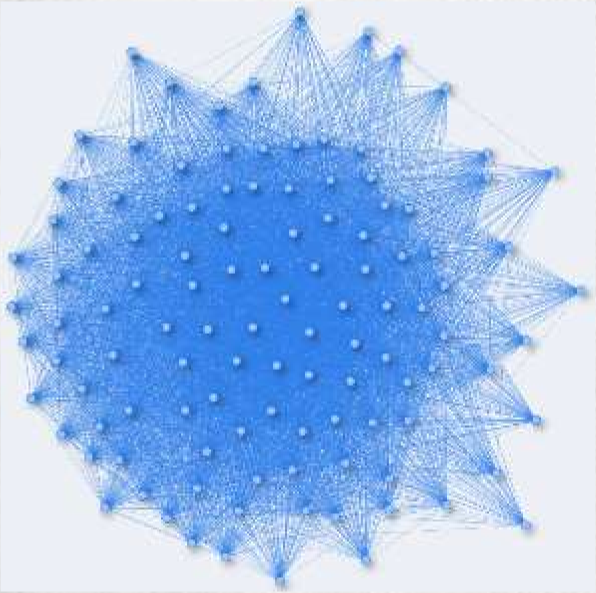


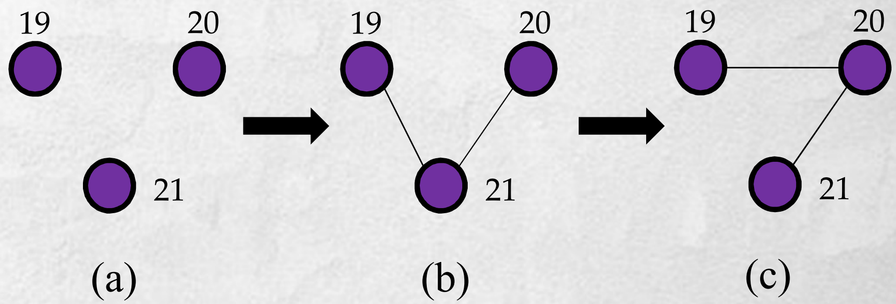
  - Add-Up Mode



Methodology: Graph Exploration

- Graph View mode
  - Regular Mode
  - Edge-Free Mode

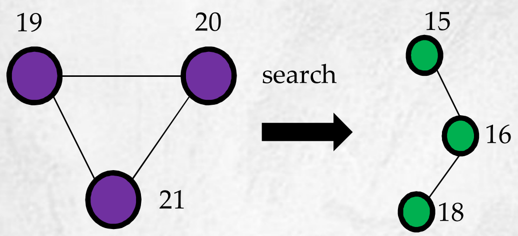




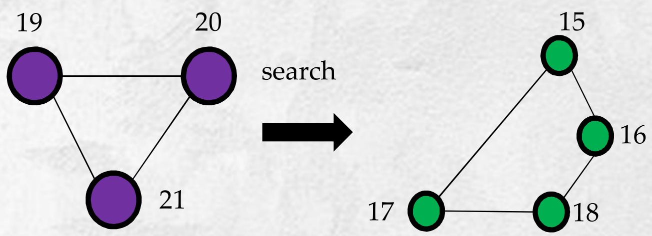
Increase readability
Improve efficiency

Methodology: Graph Exploration

- Graph Search



Search 16



Search 16 and 18

Methodology: Visualization Mode

- Local-Memory mode
  - Designed for small graphs
  - Graph data can be completely loaded into main memory.
  - Crossover edge generation is done on local machine.
- Distributed-Memory mode
  - Designed for large-scale graphs.
  - Graph and its hierarchy data are distributed into multiple machines.
  - To minimize the data requests to Spark, only required data is retrieved from Spark.

Methodology: Decorations and Interactions

- Decorations and Interactions
  - Zooming in/out
    - Adjust the viewpoint to observe details.
  - Vertex identification
    - To find out one vertex
  - Vertex selection and layer selection
    - To select one vertex or a group of vertices

BGS YouTube Video: (search Big Graph Surfer)
https://www.youtube.com/watch?v=YWvuDcMAfAE&t=

Case Study

- BGS Functionalities
- BGS Scalability
- Interactions

Graph	Vertices	Edges	Attributed	File Size
Facebook	4,039	88,234	No	1MB
Flight	3,110	32,381	Yes	1MB
Friendster	65,608,366	1,806,067,135	No	30GB

Case Study

- BGS Scalability

Table 1: clustering time, loading time, and visualization time for graph datasets

Graph	Clustering Time	Graph Load Time	Visualization Delay Time	Visualization Mode
Facebook	49 s	1 min	1-3 s	Local Memory
Flight	————	50 s	1-3 s	Local Memory
Friendster	4.2 h	3 min	about 20 s	Distributed Memory

- Interactions
  - zooming in or out
  - vertex identification
  - vertex selection or level selection
  - clusters expanding etc.

Discussion

Table 2: Combination of visualization mode and expansion mode

View mode / Visualization mode	**Minimum mode**	**Add-Up mode**
Local-Memory mode	1) Fit for small-scale graph; 2) Constantly high efficient; 3) Constantly high requirement for local memory;	1) Fit for small scale graph; 2) Relative high efficient; 3) High requirement for local memory;
Distributed-Memory mode	1) Fit for large-scale graph; 2) Constantly relative high efficient; 3) Constantly low requirement for local memory;	1) Fit for large-scale graph; 2) Low efficient; 3) Low requirement for local memory;

Reference

- [1] N. Elmqvis, T. N. Do, H. Goodell, N. Henry, and J. D. Fekete, "ZAME: Interactive large-scale graph visualization," *IEEE Pacific Vis. Symp. 2008, PacificVis - Proc.*, pp. 215–222, 2008.
- [2] J. Abello, F. Van Ham, and N. Krishnan, "ASK-Graph View: A large scale graph visualization system," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, p. 669, 2006.
- [3] B. Jeon, I. Jeon, and U. Kang, "TeGViz: Distributed Tera-Scale Graph Generation and Visualization," in 2015 IEEE International Conference on Data Mining Workshop (ICDMW), 2015, pp. 1620–1623
- [4] J. Heer, "Vizster : Visualizing Online Social Networks," 2003.
- [5] J. A. Guerra-gomez, A. Wilson, J. Liu, and D. Davies, "Network Explorer : Design , Implementation , and Real World Deployment of a Large Network Visualization Tool," *Proc. Int. Work. Conf. Adv. Vis. Interfaces - AVI '16*, pp. 108–111, 2016.
- [6] N. Bikakis, J. Liagouris, M. Krommyda, G. Papastefanatos, and T. Sellis, "GraphVizdb: A scalable platform for interactive large graph visualization," in *2016 IEEE 32nd International Conference on Data Engineering, ICDE 2016*, 2016, pp. 1342–1345.
- [7] J. Abello and F. Van Ham, "Matrix zoom: A visual interface to semi-external graphs," in *Proceedings - IEEE Symposium on Information Visualization, INFO VIS*, 2004, pp. 183–190.
- [8] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks.," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 99, pp. 7821–7826, 2002.
- [9] S. van Dongen, "Graph clustering by flow simulation," *Graph Stimul. by flow Clust.*, vol. PhD thesis, p. University of Utrecht, 2000.
- [10] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of community hierarchies in large networks," *Networks*, pp. 1–6, 2008.

Questions?

Thank you!

Fangyan Zhang
Email: fz56@msstate.edu

Acknowledgment

This work has been supported by the United States Army Corps of Engineers under Contracts W912HZ-17-C-0016 and W912HZ-17-C-0015, by the U.S. Department of Defense, and by the Pacific Northwest National Laboratory which is managed for the U.S. Department of Energy by Battelle under Contract DE-AC05-76RL01830