Task taxonomy for graph visualization

Bongshin Lee, Catherine Plaisant, Cynthia Sims, Jean-Daniel Fekete, Nathalie Henry

Task Taxonomy for Graph Visualization

Bongshin Lee, Catherine Plaisant, Cynthia Sims Parr
Human-Computer Interaction Lab
University of Maryland,
College Park, MD 20742, USA
+1-301-405-7445
{bongshin, plaisant, csparr}@cs.umd.edu

Jean-Daniel Fekete, Nathalie Henry
INRIA Futurs/LRI Bat. 490
Université Paris-Sud,
91405 ORSAY, France
+33-1-69153460
Jean-Daniel.Fekete@inria.fr, nhenry@lri.fr

ABSTRACT
Our goal is to define a list of tasks for graph visualization that has enough detail and specificity to be useful to designers who want to improve their system and to evaluators who want to compare graph visualization systems. In this paper, we suggest a list of tasks we believe are commonly encountered while analyzing graph data. We define graph specific objects and demonstrate how all complex tasks could be seen as a series of low-level tasks performed on those objects. We believe that our taxonomy, associated with benchmark datasets and specific tasks, would help evaluators generalize results collected through a series of controlled experiments.

Categories and Subject Descriptors
H.5.2 [**Information Interfaces and Presentation**]: User Interfaces – *Graphical user interfaces (GUI), Interaction styles, Screen design, Evaluation/methodology*.

General Terms
Design, Experimentation, Human Factors.

Keywords
Task Taxonomy, Graph Visualization.

1. INTRODUCTION
Despite a long history of graph visualization research, only a few graph visualization systems have actually been tested with real users. Furthermore, the tasks that were used in these studies have been highly domain-specific. To improve the evaluation of information visualization systems, it is important to have benchmark datasets and tasks [6]. In this paper, we suggest a list of tasks commonly encountered while analyzing graph data.

There have been a number of general InfoVis task taxonomies, such as the task by data taxonomy [7]. We first prepared lists of tasks with examples taken from several domains such as food webs, bibliography, and student class assignments. We used the taxonomy of tasks for tree visualization posted in the InfoVis 2003 contest [3] as a starting point. We then reviewed several user studies of graph visualization techniques and extracted the tasks used in those studies.

After making those two lists, we considered the set of low-level Visual Analytics tasks proposed by Amar *et al*. [2]. These tasks were extracted from a corpus of questions about tabular data. We realized that our tasks all seem to be compound tasks made up of Amar *et al*'s primitive tasks applied to the graph objects. When some tasks could not be represented with those tasks and objects, we added either an object or a low-level task. In this paper, we demonstrate how all complex tasks could be seen as a series of low-level tasks performed on those objects.

2. Graph-Specific Objects
A graph consists of two types of primitive elements, nodes and links. A subgraph of a graph G is a graph whose nodes and links are subsets of G. There are several meaningful subgraphs such as connected components.

2.1 Nodes
Nodes by nature have an attribute degree that is the number of links incident to that node. In a directed graph, nodes have two types of degrees according to the direction; indegree and outdegree. For practical use, nodes also have a special "label" attribute. They often have application-dependent attributes as well. In network analysis, there are various measures used to determine the centrality, or relative importance, of a node within the graph (for example, the importance of a person within a social network). Measures of centrality include betweenness and closeness. There is also a special kind of node called an articulation point, whose removal disconnects a graph.

2.2 Links
Links can have labels and application-specific attributes. For a directed graph, each link also has a "direction" attribute. A bridge is a link whose removal disconnects a graph.

2.3 Paths
A path is an alternating sequence of nodes and links, often represented as a sequence of just nodes, since there is only one link between two nodes in most cases. If the first and last nodes of the path are the same, we call it a cycle. The shortest path between two nodes is a path in which the sum of the weights of the constituent links is minimized. If the links are not weighted, we minimize the number of links in the path instead.

2.4 Graphs

We consider graphs to be objects, as users might want to compare graphs or see how a graph changes over time. Graphs have a "directed" attribute defined by whether or not links in the graph are directed and a "cyclic" attribute defined by whether or not the graph contains any cycles. Graphs can also have some computed attributes such as the number of nodes and links.

2.5 Groups

A group can be defined as a set of related nodes, such as nodes with common attribute values or nodes of interest to users.

2.6 Connected Components

A connected component is a maximal connected subgraph.

2.7 Clusters

A cluster is a set of objects that are spatially close together. For graphs, this is a subgraph of connected components whose nodes have high connectivity. Thus, in our terminology, clusters are based solely on link information, in contrast to a group.

3. Low-Level Tasks

Amar's low-level visual analytic tasks (shown in Table 6.1) [2] are all relevant to graphs. In the task descriptions, a "data case" is an entity in the dataset and an "aggregation function" is a function that creates a numeric representation for a set of data cases, such as average and sum.

Table 1. Ten Analytic tasks proposed by Amal *et al*.

Tasks	Descriptions
Retrieve Value	Given a set of cases, find attributes of those cases.
Filter	Given some conditions on attributes values, find data cases satisfying those conditions.
Compute Derived Value	Given a set of data cases, compute an aggregate numeric representation of those data cases. (e.g. average, median, and count)
Find Extremum	Find data cases possessing an extreme value of an attribute over its range within the data set.
Sort	Given a set of data cases, rank them according to some ordinal metric.
Determine Range	Given a set of data cases and an attribute of interest, find the span of values within the set.
Characterize Distribution	Given a set of data cases and a quantitative attribute of interest, characterize the distribution of that attribute's values over the set.
Find Anomalies	Identify any anomalies within a given set of data cases with respect to a given relationship or expectation, e.g. statistical outliers.
Cluster	Given a set of data cases, find clusters of similar attribute values.
Correlate	Given a set of data cases and two attributes, determine useful relationships between the values of those attributes.

We see that the last three tasks do not have ground truth answers that we can easily compare with users' answers. The "Correlate" task may have a statistical ground truth but we assume that in the field of Information Visualization, the intended meaning of "Correlate" is "identify possible correlations."

We propose one graph-specific task and two general tasks that are not covered by the above list.

- Find Adjacent Nodes: Given a node, find its adjacent nodes.
- Scan: Quickly review the list of items.

  This task differs from the "Retrieve Value" task, since it usually requires users to review many items at once but not necessarily to retrieve exact values. For example, if users want to find "Robin Williams" they can immediately move to the next item if it does not start with "R." They can also stop when they find an answer. Depending on the task, users may need to continue to review all items. The values may not be specific, for example users may need to scan for foreign names. They need not be text, for example users may need to scan for color-coded information.
- Set Operation: Given multiple sets of nodes, perform set operations on them. For example, find the intersection of the set of nodes.

4. Graph Task Taxonomy

In this section, we summarize a list of tasks commonly encountered while analyzing graph data. These suggested tasks are further categorized into four groups: topology-based tasks, attribute-based tasks, browsing tasks, and the overview task. Each task has general descriptions and example scenarios. FOAF, FW, GO, and ARM represent friend-of-a-friend graph, food webs, gene ontology, and airport routing map respectively. In addition, we show how each task can be decomposed into low-level tasks, shown in italics, on specified graph objects. While there might be several ways to decompose a task, we show one way.

Note that finding a node is a common starting point for many tasks. But this task may not be performed by users when a search feature is provided by the system. While we describe it as a component for each task, we may want to exclude it when we conduct a user study.

4.1 Topology-Based Tasks

4.1.1 Adjacency (direct connection)

General Descriptions:

- Find the set of nodes adjacent to a node.
- How many nodes are adjacent to a node?
- Which node has a maximum number of adjacent nodes?

Examples:

- (FOAF) Find the names of the direct friends of Eric.

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Retrieve Value* on Nodes]
- (FW) How many kinds of organisms do golden eagles eat?

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Filter* on Links + *Compute Derived Value (Count)* on Nodes]
- (FOAF) Who is the most popular person?

  [*Find Extremum* on Nodes]

4.1.2 Accessibility (direct or indirect connection)

Accessibility task can be treated as a repetition of the Adjacency task.

General Descriptions:

- Find the set of nodes accessible from a node.
- How many nodes are accessible from a node?
- Find the set of nodes accessible from a node where the distance is less than or equal to n.
- How many nodes are accessible from a node where the distance is less than or equal to n?

Examples:

- (FOAF) Who are your friends, your friends' friends, and so on?

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Retrieve Value* on Nodes) until no more new adjacent nodes are found]

- (FOAF) How many friends are you connected to in this way?

[*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes) until no more new adjacent nodes are found + *Count* on Nodes]

- (ARM) To what cities can we go from Seoul, Korea by changing planes only once?

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Filter* on Links + *Retrieve Value* on Nodes) twice]

4.1.3 Common Connection

General Descriptions:

- Given nodes, find a set of nodes that are connected to all of them.

Examples:

- (FOAF) Find all the people who know both John and Jack.

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Set Operation (Intersect)* on Groups]

4.1.4 Connectivity

General Descriptions:

- Find the shortest path between two nodes.
- Identify clusters.
- Identify connected components.
- Find bridges.
- Find articulation points.

Examples:

- (ARM) What is the shortest path from Seoul, Korea to Athens, Greece?

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes in a breadth-first manner) until find the path]

- (FOAF) Count the number of clusters.

  [*Scan* on Graphs to count clusters]

- (FW) There may be subgraphs independent of each other. Count the number of connected components in the graph.

  [*Scan* on Graphs to count connected components]

- (FOAF) Who is the person whose removal from the graph results in an unconnected graph?

  [*Scan* on Graphs to find an articulation point]

- (FW) Which is the eating link whose removal from the graph results in an unconnected graph?

  [*Scan* on Graphs to find a bridge]

4.2 Attribute-Based Tasks

All the previous topology tasks can be repeated with added *filter*, *compute*, *range*, or *distribution* tasks (opposed to solely *count* tasks) on the attributes either on nodes or on links.

4.2.1 On the Nodes

General Descriptions:

- Find the nodes having a specific attribute value.
- Review the set of nodes.

Examples:

- (FOAF) Who do you know from the people currently shown on screen?

  [*Filter* on Nodes + *Retrieve Value* on Nodes]

- (FOAF) How many people do you know from the ones currently shown on screen?

  [*Count* on Nodes while *Scan* on Nodes]

- (FOAF) Are there any foreign-sounding names?

  [*Scan* on Nodes until find an answer]

4.2.2 On the Links

General Descriptions:

- Given a node, find the nodes connected only by certain types of links.
- Which node is connected by a link having the largest/smallest value?

Examples:

- (GO) Find the nodes connected by "is-a" relationships from the "Biological Process" node.

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Filter* on Links + *Retrieve Value* on Nodes]

- (FW) If a link has an attribute representing the percentage of the diet, what is main food of American crow?

  [*Find* on Nodes + *Find Adjacent Nodes* on Nodes + *Find Extremun* on Links + *Retrieve Value* on Nodes]

4.3 Browsing Tasks

4.3.1 Follow Path

General Descriptions:

- Follow a given path.

Examples:

- (FOAF) A user looks into A's friend B, B's friend C, and C's friend D.

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Scan* on Nodes) three times]

- (FW) Follow the flow of energy from grasses, to a rabbit that eats grass, to a carnivore that eats the rabbit, and to a carnivore that eats that carnivore.

  [*Find* on Nodes + repeat (*Find Adjacent Nodes* on Nodes + *Scan* on Nodes) three times]

4.3.2 Revisit

General Descriptions:

- Return to a previously visited node.

Examples:

- (FOAF) After they follow a path in the above task, they may want to see A's other friends.

  [*Scan* on Nodes + *Find Adjacent Nodes* on Nodes]

- (FW) Find another carnivore that eats the same rabbit.

  [repeat (*Scan* on Nodes) twice to find + *Find Adjacent Nodes* on Nodes]

4.4 Overview Task

This is a compound exploratory task to get estimated values quickly. For example, we might ask users to estimate the size of the social network. Note that sometimes it is more important to be able to estimate the answer than to get an accurate one. Some of the topology tasks can be done easily using an overview of the graph as well. For example, using particular layout algorithms, it is easy to see whether or not there are clusters and connected components. Other algorithms help to find shortest paths between nodes. Overview also helps to find patterns.

5. High-Level Tasks

There are high-level tasks that are not covered by the above tasks.

- When we compare two or more food webs, we can ask the following questions: What do they have in common? What are the differences among those food webs? Is there any missing or conflicting information?
- Due to errors in the data, several nodes may represent the same entity. For example, the co-authorship graphs often have duplicate author nodes. One important task is to identify whether two or more nodes represent the same person.
- How has the graph changed over time?

6. Characterizing Graph Visualization Tools

Graph visualization tools can be characterized by which objects (nodes or links) and tasks they focus on as shown in Table 2. For example, TreePlus [5] focuses on nodes – less on links and not at all on clusters. NVSS [8] and NetLens [4] focus on groups, and most classic node-link diagrams and matrix representations [1] do well at clusters. User studies for these tools focus on certain tasks, e.g. scanning and following path for TreePlus, or finding clusters and bridges in the graph layout experiments. Some tools seem most strong at certain tasks. For example, NetLens particularly excels at showing a distribution of items and filtering and sorting items. Since NVSS partitions a large network into several smaller non-overlapping regions by node attributes, users were able to quickly identify patterns of interest among nodes.

It would be useful if we could further characterize graph visualization tools by which graph characteristics they effectively visualize. For example, some tools may be better at handling directed graphs and others at high-density graphs.

7. CONCLUSION

We have defined graph-specific objects and demonstrated how all complex tasks for graph visualization could be seen as a series of low-level tasks performed on those objects. We believe that our taxonomy, associated with benchmark datasets, would help evaluators generalize results collected through a series of controlled experiments.

8. ACKNOWLEDGMENTS

We would like to thank Ben Shneiderman for encouragement and many helpful comments.

9. REFERENCES

[1] Abello, J. and Korn, J. MGV: A System for Visualizing Massive Multigraphs, *IEEE Trans. Visualization and Computer Graphics*, vol. 8, no, 1, pp. 21-38, Jan.-Mar. 2002.

[2] Amar, R., Eagan, J., and Stasko, J. Low-Level Components of Analytic Activity in Information Visualization, *Proceedings of the Symposium on Information Visualization (InfoVis '05)*, pp. 111-117, 2005.

[3] InfoVis 2003 Contest: Visualization and Pair Wise Comparison of Trees, http://www.cs.umd.edu/hcil/iv03contest.

[4] Kang, H., Plaisant, C., Lee, B., and Bederson, B.B. NetLens: An Interface Using Iterative Query Refinement in Bipartite Network Data Model, To appear in *Proceedings of Joint Conference on Digital Libraries (JCDL '06)*, Demonstration, 2006.

[5] Lee, B., Parr, C.S., Plaisant, C., Bederson, B.B., Veksler, V.D. Gray, W.D., and Kotfila, C. TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts, To appear in *IEEE TVCG Special Issue on Visual Analytics*.

[6] Plaisant, C. The Challenge of Information Visualization Evaluation, *Proceedings of the working conference on Advanced Visual Interfaces (AVI '04)*, pp. 109-116, 2004.

[7] Shneiderman, B. The eyes have it: A task by data type taxonomy for information visualizations, *Proceedings of the Symposium on Visual Languages (VL '96)*, pp. 336-343, 1996.

[8] Shneiderman, B. and Aris, A. Network Visualization by Semantic Substrates, University of Maryland Technical Report, 2006.

Table 2 Characterizing graph visualization tools based on their focus objects and tasks. These characterizations assume large graphs. Large, bold X represents particular strengths.

	TreePlus	Matrices	NVSS	Classic node link	NetLens
OBJECT FOCUS					
Nodes	**X**	X	x	X	X
Links		X	x	X	
Paths	x			**X**	
Graphs					
Groups			**X**		X
Connected Components	x	X		**X**	
Clusters		X		**X**	
GENERAL LOW-LEVEL TASKS					
Retrieve Value	X				X
Filter			**X**		**X**
Compute Derived Value					
Extremum					X
Sort	x				**X**
Range					X
Distribution		x		X	**X**
Anomalies				x	
Cluster		X		**X**	
Correlate					
Scan	**X**			x	X
Set Operation	x				X
GRAPH-SPECIFIC TASK					
Find Adjacent Nodes	x	x		X	
COMPLEX TASKS					
Topology					
Adjacency	X	X	x	x	X
Accessibility	X			x	
Common Connection	Interaction	Interaction		Visual	
Find Shortest Path	Computed			Visual/Computed	
Find Clusters		Computed/Visual		Visual	
Find Connected Components	Computed			Visual	
Find Bridges				Visual	
Find Articulation Points				Visual	
Attribute-Based					
On the Nodes	**X**		**X**		**X**
On the Links			X	x	
Browsing					
Follow Path	**X**			x	
Revisit	**X**			X	
Overview		x		**X**	X