Exploring User-definable Graph Layouts

Tim Bodyka Heng, Jeffrey Guenther, Chris D. Shaw
Simon Fraser University – School of Interactive Arts and Technology

Introduction

Graphs are a common type of data encountered by analysts. Social networks, co-author networks, and gene interactions can all be represented as graphs. These types of data can be difficult to understand without the aid of a visualization. Visual analytics tools with graph visualizations allows the analysts to view data as a node link diagram. Vertices or nodes represent objects. Edges represent the relationship between two nodes. To render graphs, a number of automatic graph layouts have been developed. These algorithms use properties of the graph, like the number of interconnections between nodes, to compute positions for the graph's vertices. While these algorithms have a number of parameters that can be adjusted to affect the layout, they are automatic – no analyst input is used to position the graph nodes.

In this paper, we will discuss how automated graph layouts do not meet the needs of analysts and propose an interaction technique to better meet the needs of the analyst. We then describe the current state of our implementation.

Problem Description

In the process of making sense of a data collection, analysts spend time organizing their data in a schema [6]. As analysts organize their data, they make sense of it. As analysts try to make sense of graph data, they need ways to organize objects and their relationships, or graph nodes and their edges.

In this section of the paper, we discuss in detail about the problems of how automated graph layout do not meet the demands of analysts. User definable graph layout, persistent selection and hierarchical graph layout will be discussed followed by obvious solutions and their drawbacks.

Graphical layout introduces a number of questions to be answered in terms of how the analyst would interact with the graph and how the graph should be presented to the analyst. Visual analytics tool needs to be able to provide the analyst with the power to express himself in a way that meets his understanding of the data. The analyst should be able to freely manipulate the nodes and control how the graph would look without any constraints.

There are a number of apparent approaches that could help in solving this problem. One approach is to use manual layout. The analyst is able to manipulate each node independently without affecting other nodes. Individual nodes can be moved in the graphical space depending

on where the analyst wants it to be. For example, if the analyst wants all the nodes to sit in a circle, the analyst can move the node one by one to form a circle. This seems like an easy approach but it would not help the analyst with the sense-making process if he works with a large amount of nodes. It would be a tedious and time consuming process to manually move each individual node one by one to form a circle.

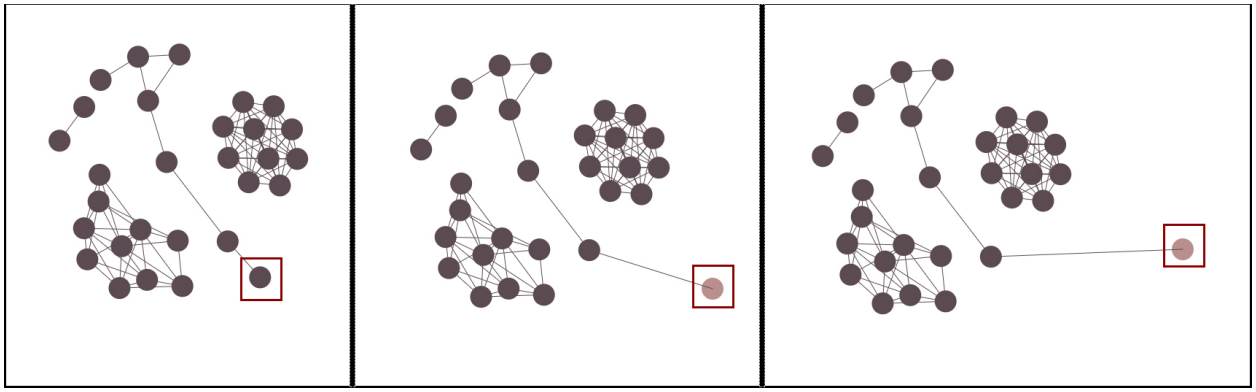

Figure 1: An example of how the highlighted node being moved manually without affecting other nodes.

Another possible approach is to use automated layout that would allow the analyst to lay out all the nodes to a type of layout he desires. For example, if the analyst is working with a graph layout with many interconnected nodes, he can choose “Circle Layout” and the system would rearrange all the nodes into a perfect circle layout. The problem with this approach is if the analyst is not able to apply a circle layout to a subset of the graph nodes.

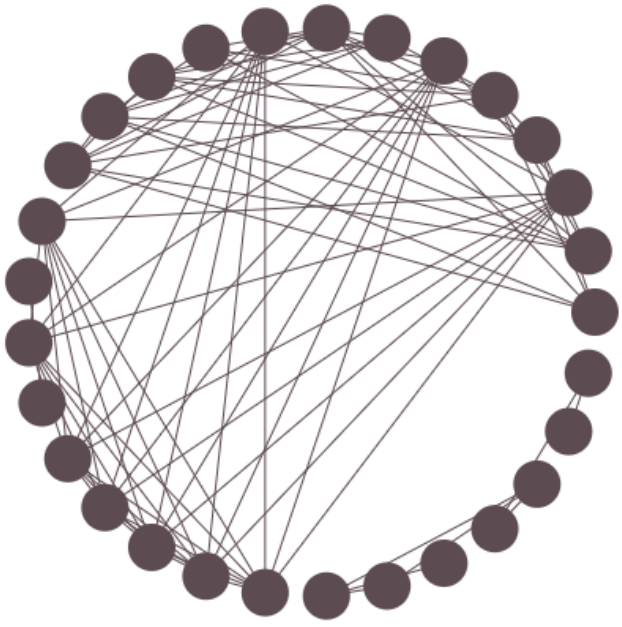

Figure 2: An example of automatic layout and how all nodes are automatically arranged into a circle.

Another approach is to combine the manual and automated layout. In other words, the analyst is able to choose an automatic layout algorithm to define how the graph is laid out and still allows the analyst to manually position each node. For example, the analyst can choose to use a “Circle Layout” and the system would arrange all the nodes in a circle. Once the layout algorithm is finished, the analyst can relocate any node as needed. Nevertheless, without the ability to select multiple nodes and apply a layout to those nodes, the analyst is required to locate those nodes manually. This procedure can be a time consuming and tedious process when the analyst has to work with a lot nodes.

The selections that the analyst make should be persistent so that the analyst can choose to come back and work on the previous selections at any time. Persistent selections help the analyst organize the data with the ability to go back to any previously made selections without reselecting the nodes. Using this approach, selections are always available to the analyst to work with no matter what stage the analyst is in. For instance, when the analyst works with multiple selections, there might be interesting relationships between two or more separate selections that he wants to explore further. With persistent selection, the analyst can access those selections without having to make new selections.

Moreover, hierarchical layout should also come in as part of the approach. Depending on the data, the analyst would want to view the graph as a hierarchical layout to help with the sense-making process. Hierarchical layout would help the analyst to understand the importance of each node comparing to others and the hierarchical structures between nodes. Hierarchical layout shows connections of each node in hierarchical structure. Parent nodes are placed on the top of layout with multiple children branch out from the parent nodes.

Previous Solutions

There are a number of published papers including tools that attempt to tackle these problems using different techniques. Kadivar et al.[3] discusses about a visual analytics tool called CzSaw, which allows analysts to view the data in graphical layout. The system allows the analyst to see how nodes are interconnected and the analyst can move each individual node independently. The analyst can also select multiple nodes and move them simultaneously around the space. CzSaw offers a variety of different layouts including grouped nodes, lists and scatterplot points etc.

However, CzSaw is still inadequate in supporting user definable graph layout. It does not fully support the analyst to be able to apply a new layout to a subset of all the nodes within the primary layout. For example, the analyst can move each individual node independently but he cannot select a subset of those nodes and apply a new layout for them. Thus, CzSaw lacks the ability to provide the analyst the power to arrange a selection of nodes into a desired graphical

layout option.

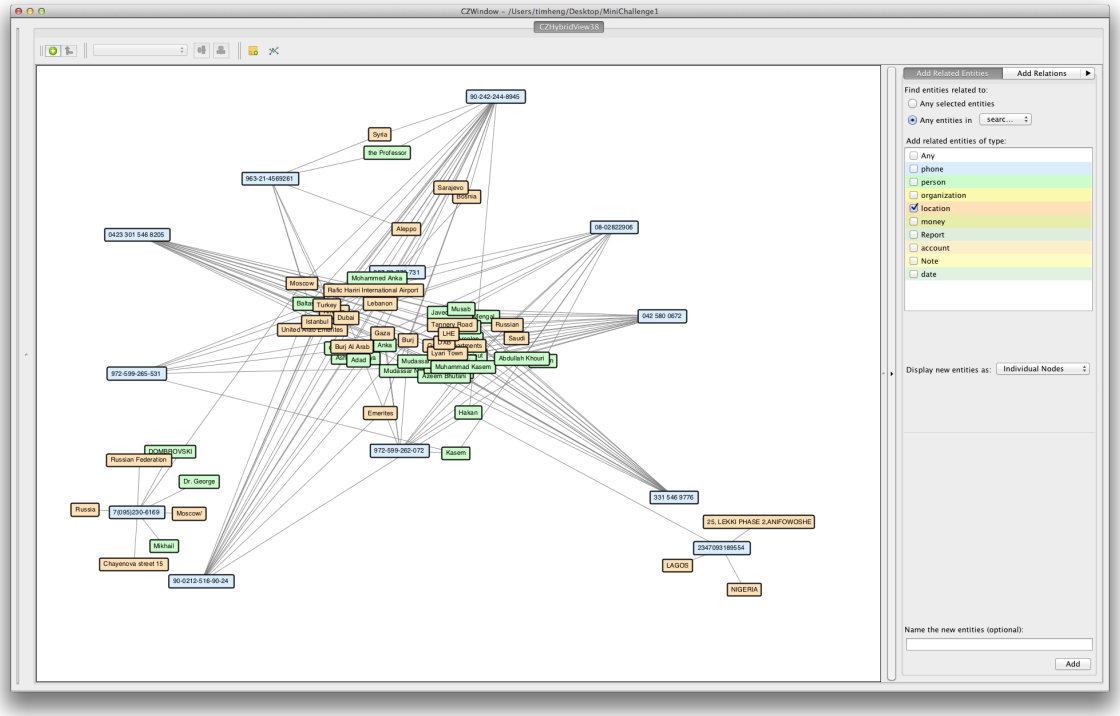

Figure 3: CzSaw's representation of graphical layout.

Another tool that works on the problems is Jigsaw [2]. Jigsaw allows analysts to work with documents in graphical layout representation. Documents are clustered into separate themes that support the analyst to look for a particular person, organization or place etc. It also presents those entities in a form of graphical layout by showing each entity as a node and all the nodes are interconnected. The analyst is able to manipulate each individual node to where they would be placed on the canvas without affecting other nodes, which is similar to what CzSaw is capable of. Nevertheless, it still lacks the power to support analysts to personally define how the graph layout should be presented. Similar to CzSaw, the analyst cannot work with a subset of nodes. Jigsaw would require the analyst to rely on global automated layout and later the analyst would have to manually manipulate each node independently.

In addition to CzSaw and Jigsaw, Cytoscape offers similar functionalities where the analyst can visualize the data in different graph layouts. Cytoscape provides the analysts with a variety of simple layouts that the analyst can use including "Grid Layout", "Circular Layout" and "Hierarchical Layout" etc. What differs Cytoscape from CzSaw and Jigsaw is that it allows the analyst to select a subset of nodes from the main graph and apply a new layout to it without affecting the primary graph layout. Having said that, Cytoscape does not allow the analyst to resize the sublayouts. The size of the sublayout is generated by calculating the amount of nodes in the sublayout. This limitation does not give the analyst the total freedom of defining how the sublayout would be organized. Another aspect that makes Cytoscape inadequate to fully support

user definable graph layout is the selections are not persistent. The analyst cannot go back and manipulate the previously selected group of nodes. Without persistent selection, the analyst is required to constantly make a new selection every time the analyst wants to work with multiple nodes.

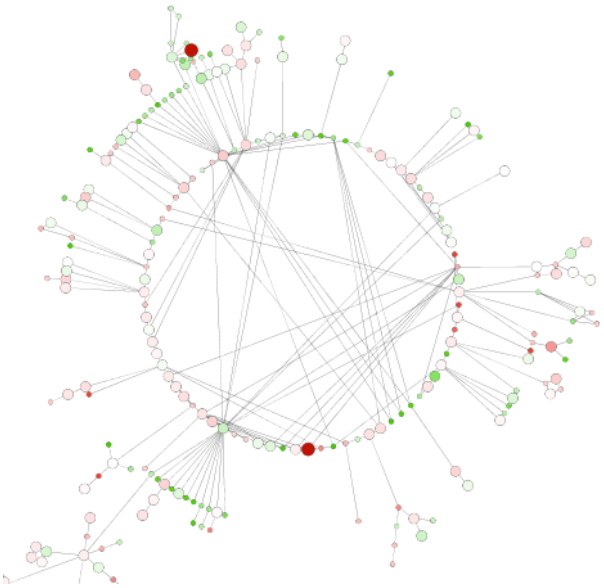

Figure 4: Circular Layout, one of the automatically generated layouts by Jigsaw

Like Cytoscape, McGuffin and Jurisica [5] discuss different selection and interaction techniques that would allow the analyst to select multiple nodes and lay them out with a selected layout option. By using a “Hot Box”, the system is able to support many complex interactions including resizing the sublayouts. However, same as Cytoscape the techniques discussed in this paper do not support persistent selection and hierarchical layout.

All the tools described above take different approaches to allow analysts to define the graph layout. However, due to their limitations, they are not able to solve the problems listed earlier. This lack of solutions motivates us to explore into interaction techniques and graphical layouts to:

- allow the analyst to have total freedom in defining the graph layout including nested sublayouts
- support persistent selections so the analyst can come back to work on the previously selected nodes
- support hierarchical graph layout

Solution

Because visual analytics tools described earlier do not support the analyst to create sublayouts with persistent selections, we’re proposing a solution to this problem by exploring into a new approach that would treat each selection as a group. Particularly, all the nodes that are selected

by the analyst are added into a newly created group. This approach would allow us to have persistent selection so the analyst can come back and manipulate this selection at any time. Furthermore, by creating a new group for every new selection it can also help to prepare the system for defining the sublayout. Once those selected nodes are added into a group, we can apply any type of graph layout to that particular group. For example, circle layout can be applied to a newly created group so that all the nodes within that group are arranged into a perfect circle. Moreover, this technique can also guide us to achieve the hierarchical layout implementation. Using different groups for different selections, the algorithm can support nested groups. Nested groups allow the system to compute the hierarchical layout of the nodes base on the hierarchical structure of the nested groups. In other words, parent groups are treated as parent nodes that branch out their children groups, which are children nodes.

Implementation

We built the prototype to support the solution mentioned above by using JUNG[1] to compute the graph layouts and JavaFX to render the graphs. Jung provides us with several layout algorithms necessary to build this prototype. Node position information is generated by JUNG and then passed to JavaFX for rendering. We use JavaFX for its scene graph and it allows us to easily build user interfaces in its scene graph. Next we'll describe the features of our prototype and explain the rationale behind them.

User Interface

Graphical user interface is featured in this prototype to support different user interactions and different features that the prototype has to offer. The toolbar on the top of the window allows the analyst to apply a layout type to the selection as well as defining the size of that layout (see figure 1). Below is a list of different controls with their descriptions:

- CircleLayout Button: Cluster all the selected into a new spring circle layout.
- SpringLayout Button: Cluster all the selected nodes into a new spring layout.
- FRLayout Button: Cluster all the selected nodes into a new Fruchterman-Reingold force-directed layout [1].
- KKLayout Button: Cluster all the selected nodes into a new Kamada-Kawai layout [4].
- StaiticLayout Button: Cluster all the selected nodes into a new Static layout.
- Slider: Set the size for the selected layout. (Left to Right: Small to Big)

[1] http://jung.sourceforge.net/

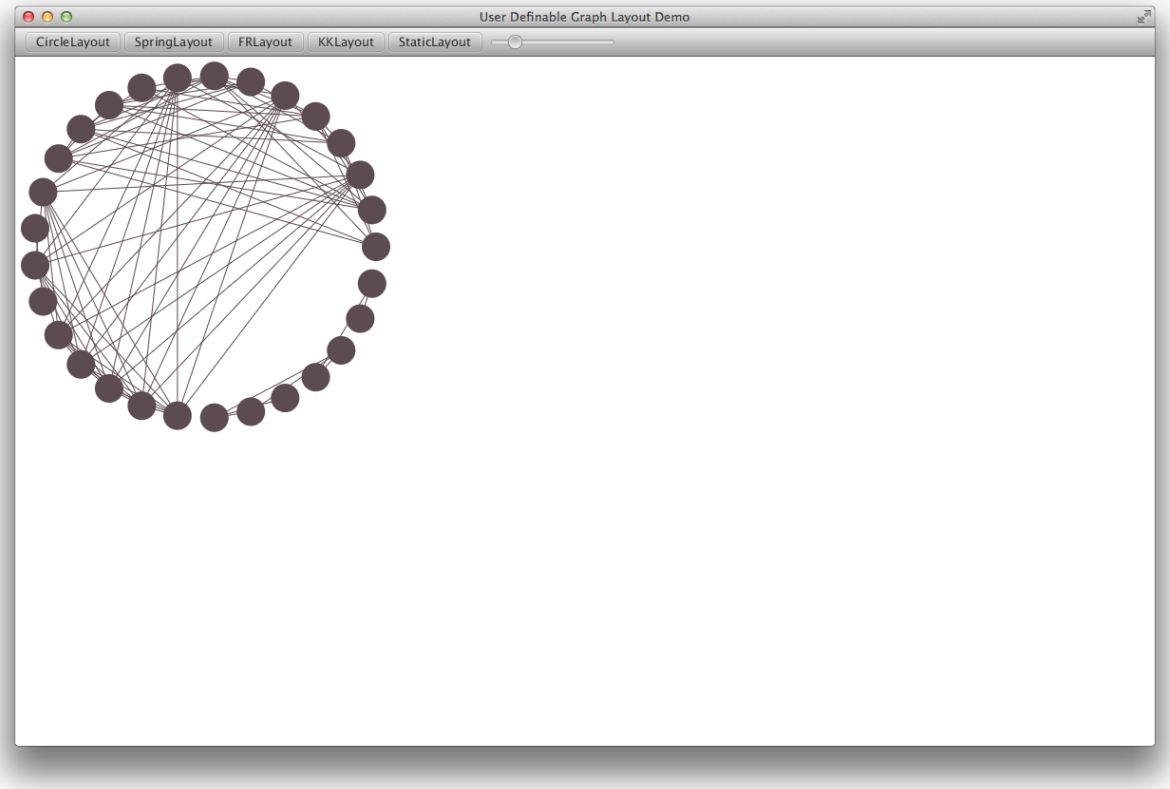

Figure 5: The user interface of the prototype.

Movable and Selectable Nodes

Each node can be moved anywhere on the canvas without affecting other nodes. Edges attached to those nodes would automatically update their endpoints according to the nodes. Using JavaFX, we are able to bind the endpoints of each edge with the appropriate nodes. We bind the X and Y value of each node with the X and Y value of an endpoint. This approach automatically updates the edges' position and direction as the nodes move. Each node can be selected with a mouse click. Selected node has its color changed from dark grey to red. The analyst can also deselect all nodes by pressing the "ESC" key. Also, multiple nodes can be moved simultaneously by drawing a rectangle around multiple nodes. Selected nodes are moved by dragging the box that contains them. This feature allows the analyst to have the flexibility to manipulate any individual node or multiple nodes simultaneously.

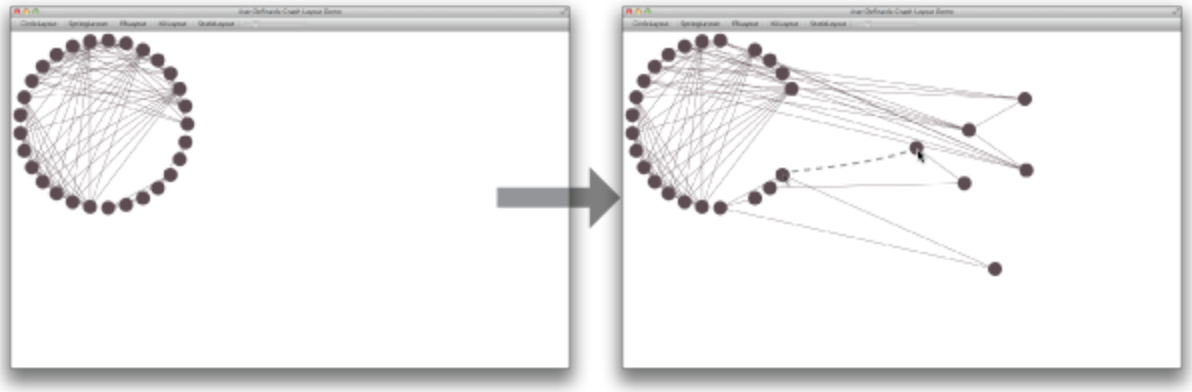

Figure 6: Moving individual node by click and drag the node

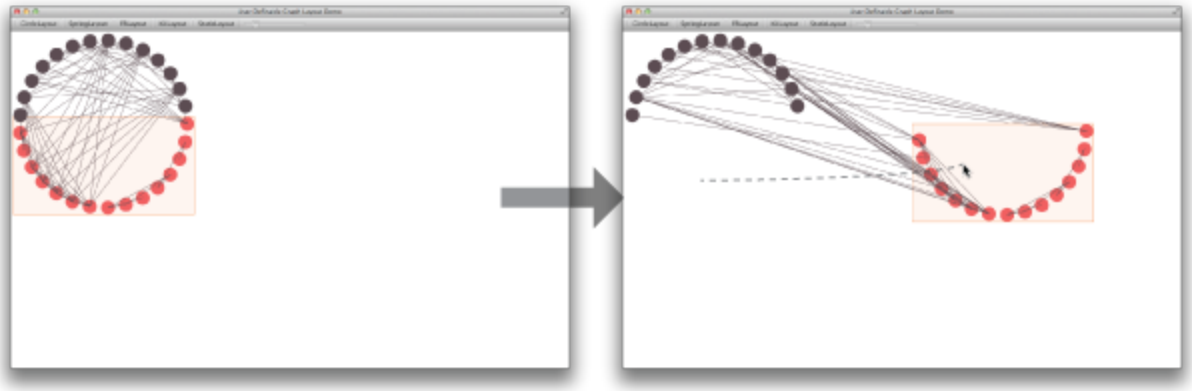

Figure 7: Moving a group of nodes by selecting multiple nodes and drag the group.

N- Number of Layouts Support

The current prototype is able to support nested sublayouts within one primary graph layout. The analyst can create as many layouts in one primary graph layout as he needs. We believe that this is an important feature to be implemented into the prototype because it can help visual analysts to be able to cluster nodes from the primary graph to a new layout system without affecting the primary graph layout. This feature lets the analyst defines as many layouts as possible on the canvas without changing the primary graph layout. This approach can help guiding the sense-making operation because it allows the analyst to focus on a subset of nodes to work with and not affecting the entire graph layout. Changing the primary graph layout while making sublayouts may cause the analyst to get lost in the sense-making process. It loses the schema that the analyst previously created for supporting his understanding of the data.

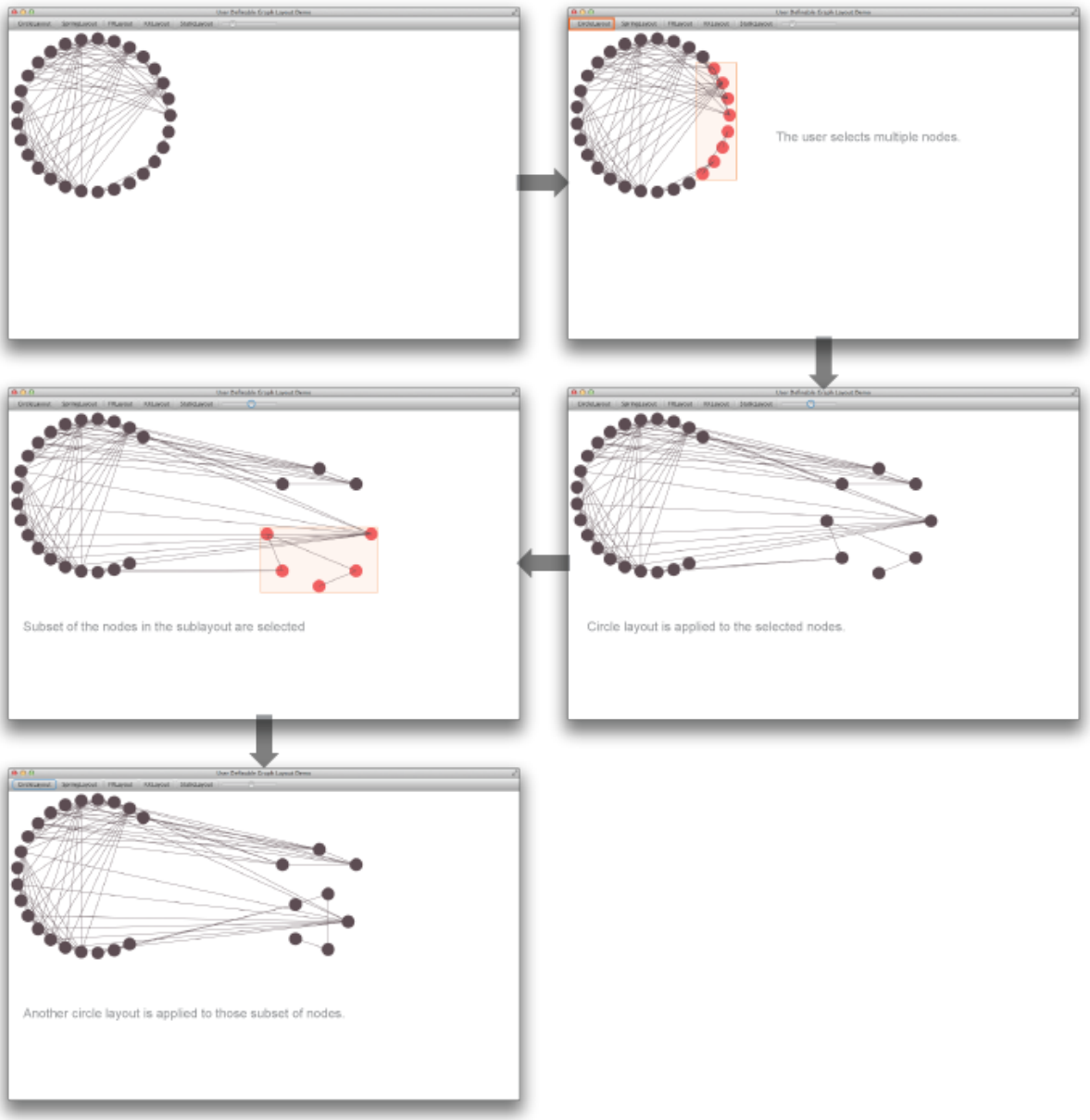

Figure 8: Steps to create a nested layout.

Resizable Layout

All selected groups are resizable by using the slider in the toolbar. The slider automatically adjusts the size of the layout with respect to the selected layout type. For example, if the circle layout is applied to a sublayout, by adjusting the slider it will resize the sublayout as a circle layout.

By allowing the analyst to be able to resize the layout, the analyst can manipulate how the layout would look. This capability can help the analyst to organize different layouts into different sizes.

For example, the analyst can select multiple nodes and apply circle layout to it. Now, if the analyst wants to have another bigger circle layout surrounds that original circle layout to create a centric circle layout, the analyst can easily achieve this requirement with resizable layout feature (see figure 10).

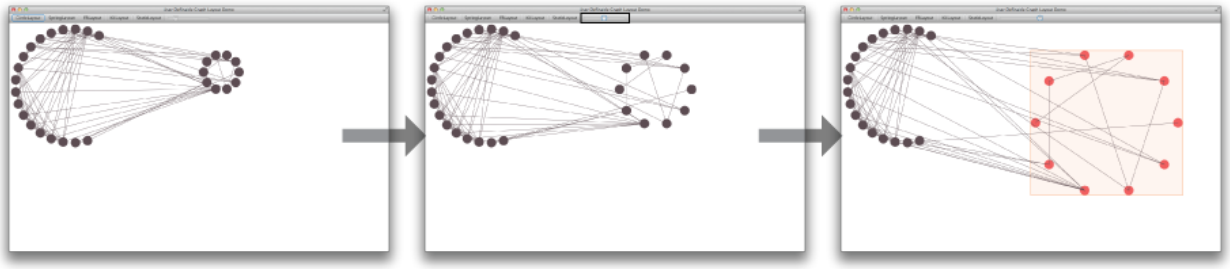

Figure 9: The circle sublayout is being resized using the slider in the toolbar

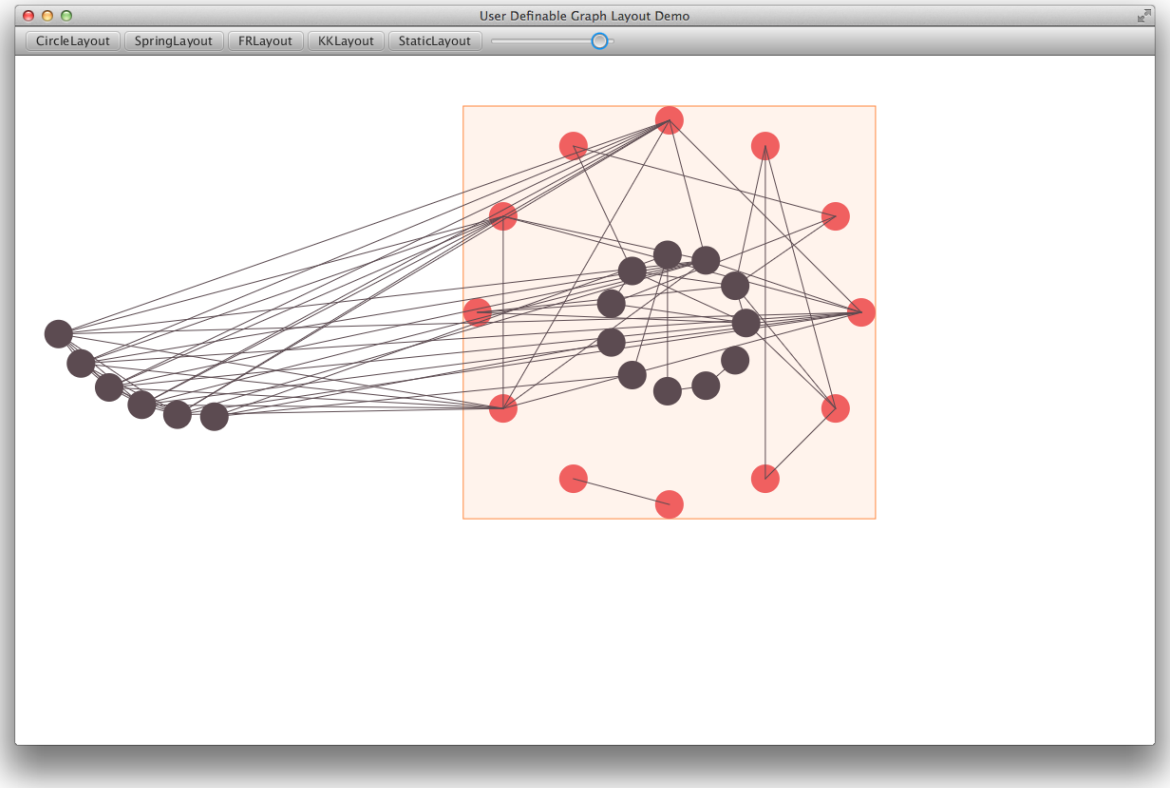

Figure 10: Using the same technique, the analyst can create a concentric circle layout

Persistent Selections

In this prototype, we implement persistent selections to grant the analyst the ability to come back to the previous selections at any time by hovering the mouse over that group. When the mouse is hovered over a particular group, that group and its nodes will be highlighted. This feature gives

the analyst the ability to manipulate different layout groups without having to reselect the previously selected groups.

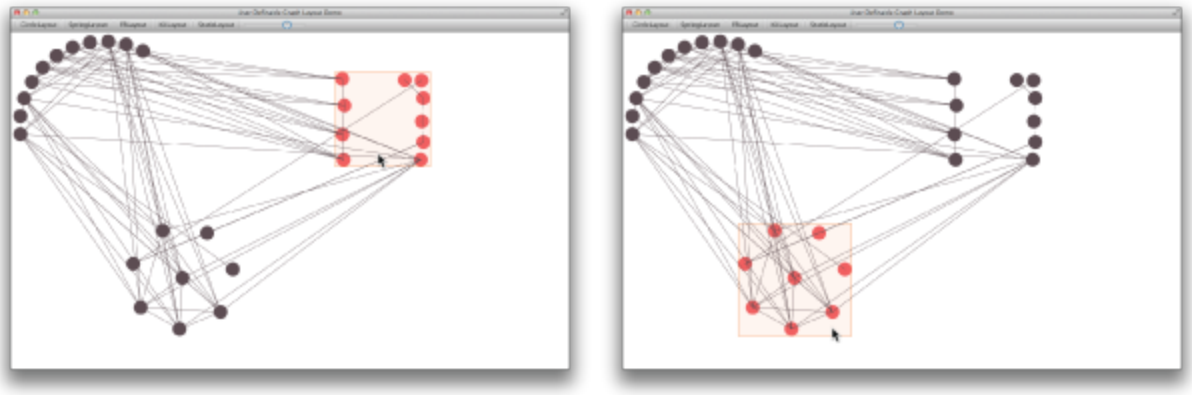

Figure 11: The analyst hovers over the FR Layout on top and the system highlights that group with the nodes in that group (Left). The analyst hovers over the KK Layout on the bottom and the system highlights that group with the nodes in in (Right).

How the implementation is a solution

As described above, this prototype has many features that would be a solution to the stated problems. This prototype is able to provide the analyst with the power to manipulate nodes and graphs with ease. The analyst is able to create as many sublayouts and nested sublayouts as needed, and the analyst is still able to come back to any of those previously created sublayouts at any given time. The layouts are resizable to enable creativity in how the analyst would want to organize the graph. The analyst is able create his own type of graph layout manually or with the help of the automated layout.

Although hierarchical graph layout is not implemented into this prototype, the way that this prototype is built prepares itself for hierarchical graph layout implementation for future work. This prototype has the features that can fill in the gaps where other visual analytics tools lack in order to give the analyst the ability to analyze data using graphical layouts.

Future Work

For future work, we are interested in implementing the hierarchical graph layout with the algorithms we have in this prototype. As mentioned earlier, the algorithm that we have built can support nested groups. By using nested groups, hierarchical layout can be constructed by computing the layout base on how the groups are structured. For example, the parent group can be placed at the top of the hierarchical layout while its children would branch out from that parent group. We also need to conduct user test with actual data analysts who work with visual

analytics tools on a daily basis. A user test would allow us to understand if our proposed solution would actually help the analysts in practical situations. In addition, it would also help us understand if the features we implemented would be helpful and if there are any other features that should be implemented as part of the solution. Another question that we're interested in from the user test is *"When working with a graph layout, is it better to have non-persistent or persistent selections?"*. In the process of constructing this prototype, we have learned that it is not always the case that persistent selection would be a better option than non-persistent selection. We are keen to understand how analysts respond to the two different types of collections.

Conclusion

Providing the analyst with the power to have persistent selection, he can define the graph layout and the sublayouts within in. This capability allows the analyst to express his understanding about the data and support the sense-making process. There are a lot of visual analytics tools that take different approaches in giving the analyst the ability to work with graph layout to help with the sense-making process. However, those tools are still inadequate to meet the needs of the analysts because they do not support the analyst to define the graph layout, and the lack of persistent selections makes it difficult to access old selections.

Of the tool surveyed, CzSaw and Jigsaw only allow the analyst to decide the the primary layout of the graph. While Cytoscape allows the analyst to have unlimited number of sublayouts, it does not support persistent selections and hierarchical layout. Last but not least, McGuffin and Jurisica discusses the techniques similar to Cytoscape by supporting nested sublayouts but those techniques do not support persistent selections and hierarchical layout. Our proposed solution expands on this previous work. Persistent selections would grant the analyst the ability to work with multiple sublayouts and nested sublayouts yet at the same time have the ability to retrieve previous selections.

References

[1] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991.
[2] C. Gorg, Z. Liu, N. Parekh, K. Singhal, and J. Stasko. Visual analytics with jigsaw. In *Visual Analytics Science and Technology, 2007. VAST 2007. IEEE Symposium on,* pages 201 –202, 30 2007-nov. 1 2007.
[3] N. Kadivar, V. Chen, D. Dunsmuir, E. Lee, C. Qian, J. Dill, C. Shaw, and R. Woodbury. Capturing and supporting the analysis process. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 131–138. IEEE, 2009.
[4] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Information processing letters*, 31(1):7–15, 1989.
[5] M. McGuffin and I. Jurisica. Interaction techniques for selecting and manipulating subgraphs in network visualizations. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6):937–944, 2009.
[6] K. Wheaton, K. Moore, S. Williams, J. Marsh, and K. Flanagan. *Wikis and Intelligence Analysis*. MCIIS Press, 2012.