University of California, Davis, Computer Science Department Technical Report CSE-2002-13

Focus+Context Display of the Visualization Process

T.J. Jankun-Kelly and Kwan-Liu Ma
Computer Science Department
University of California, Davis*

Abstract

The purpose of the visualization process is to extract insight from information via visual representations. However, the visualization process itself is information and can be in turn visualized. This work describes a method for visualizing the visualization process using a new focus+context radial graph algorithm. Four different metrics for extracting information from the visualization are examined. Using an example from a network visualization system, the effectiveness of the visualization and metrics are demonstrated. Using the methods discussed here, a better understanding of a particular visualization process—and its corresponding results—is facilitated.

Keywords: information visualization, focus+context, visualization process, visualization models, graph drawing

1 Introduction

During the visualization process, a user iteratively explores a very large space of visualization parameters in order to discover visualization results of interest. The search of this space can be costly—especially for very large data sets. By presenting the visualization process to the user, unnecessary re-explorations can be avoided. Such a presentation also assists the user understand and thus navigate the visualization space. Visualization process information is also useful for system designers: Representations of the process can highlight inefficiencies of the system or suggest common patterns of exploration. Methods for visualizing the visualization process is thus beneficial both in data exploration and system design.

This paper describes a new method for extracting visual representations of the visualization process. These *visualization process graphs* are visualized using a deformable radial graph layout to provide a focus+context view of the process. Different metrics utilizing a parameter-based model of the visualization process are used to build the process graphs. Together, the methods presented here allow effective exploration of different properties of the visualization exploration process.

2 Related Work

2.1 Focus+Context Graph Visualization

Graph visualization has been extensively studied in information visualization (see [6] for a survey). Of interest to this work are approaches which display both focus and context for graphs. There have been three main approaches in this area. The first approach distorts the graph after it has been laid out—“fish-eye” techniques fall into this category [2, 5, 10, 22, 23]. The second method maps the graph onto a higher dimensional surface and then re-projects onto two dimensions [2, 11, 12, 24]. The final approach uses non-Euclidean geometry (such as hyperbolic geometry) during layout to “open up” the space [13, 14, 19, 20, 21]. More rare are methods that use Euclidean geometry but have distortion effects in the layout of the graph [18].

Each of the above approaches has their advantages and disadvantages. Fish-eye techniques are a post-process, and thus can be easily added to existing systems; unfortunately, rendering a full fish-eye distortion of the nodes *and* edges can be costly. The higher dimensional and non-Euclidean projection approaches are effective at the cost of implementation complexity. Distorted, Euclidean layout techniques, however, can potentially offer focus+context effects with less computational cost than fish-eye and lower implementation complexity than non-Euclidean techniques. For this reason, the work presented here explores a new radial layout technique with distortion.

2.2 Radial Graph Visualization

Radial graph layouts were introduced by [4]; [3] also details radial graph layouts. Graphs using a radial layout have two features: the focus node is at the center of the layout and nodes connected to the focus node radiate outward on uniformly distant rings. This distance of a node from the root of the spanning tree induced by the focus node in the process graph determines the ring to which that node belongs. Construction of the layout proceeds from the center node in a breadth-first manner. In the traditional algorithm, each node is assigned a sector of its ring depending on the node’s angular size and the size of all of its children. Children with larger subtrees are given more area than children with smaller subtree’s. Though the original layout algorithm ensures the placement of children is concave, more recent applications for graph visualization (including this one) relax this constraint [7, 26]. Variable size nodes and animation in dynamically changing radial graphs has also been discussed [27].

The layout algorithm described here extends previous work by including a deformation of the rings. Unlike previous approaches, the spacing between the rings is no longer uniform. Like [27], graph nodes can have varying size. In this case, both radial spacing and node size decreases as the distance from the focus node increases. Using this technique, the focus element is highlighted while the rest of the graph remains in context. For our purposes, the focus is a specific visualization result while the context are the results in the process related to the focus.

2.3 Visualization Exploration Interfaces

There are few visualization interfaces tailored specifically to display the visualization process. Three interfaces of interest are the Design Galleries system [17], image graphs [16], and the spreadsheet-like exploration interface of [8]. All these interfaces consider data exploration a process of exploring a multidimensional space of visualization parameters. Design Galleries use a preprocessing step to generate a 3D representation of the design space. The user then navigates this space to find their desired image. New results are added to the representation as they are generated.

*Visualization and Graphics Research Group, Computer Science Department, University of California, Davis, CA 95616. E-mail: `{kelly, ma}@cs.ucdavis.edu`

Like the Design Galleries system, the image graph system follows a similar structured approach to exploration. An image graph is a graph representation of the visualization process that distinctly displays the relationship between generated images via glyph edges. The graph is used to explore the space of visualization parameters. Operations upon the edges and nodes in the graph can be used to generate further results, propagating such changes down the graph. The resulting graph is similar to the graphs produced by the p-set derivation metric discussed later. Unlike the result presented here, the image graph cannot display any of the other metrics of section 4.1.

To overcome screen real-estate issues when using the image graph, the visualization exploration spreadsheet of [8] was developed. This interface addresses visualization exploration by providing a movable, scalable window into the visualization parameter space. By manipulating the visualization parameters, the user changes the position and size of this "window" into visualization space. This manipulation occurs by changing the the default values for non-displayed parameters or by changing which parameters are displayed along the rows and columns. By using this interface, the data exploration process becomes the process of manipulating the spreadsheet window through visualization space.

All of these interfaces support different levels of context for understanding the visualization process. The Design Galleries displays all the results at-a-glance. While this is useful for providing context, the display does not allow one to focus very well on a specific result or determine relationships between images. The image graph, while allowing one to see relationships between some results, does not display the entire process at once; one must pan the graph in order to get the entire context. The exploration spreadsheet mitigates some of these issues by displaying more results than the image graph with better focus than the Design Galleries system. None of these systems simultaneously provide a focus and context display of the visualization process and the relationship between results in the process. This work addresses these issues.

3 The Visualization Process Model

The visualization process for both information and scientific visualization is an iterative sequence of user applied transformations from data to view [1, 25]. The fundamental operation that occurs during the visualization process is the formation of parameter value sets to derive visualization results. These parameter value sets, or *p-sets*, posses a parameter value for each parameter in a visualization transform. When applied to a visualization transform, a p-set corresponds to a rendered result. In [9], a model of the visualization process based upon a parameter derivation calculus is developed. The calculus describes how p-sets—and thus the results rendered from them—are derived from previous p-sets. New p-sets are created by user interaction with the visualization system in one of three ways:

I. $p_2(i)|\, p_0 \mapsto p_1$: parameter value $p_0(i) \in p_0$ is replaced by $p_2(i) \in p_2$ in order to derive p-set p_1.

II. $[p_0(i), p_1(i)]|\, p_0 \mapsto p_1$: a continuous range of parameters values is generated between discrete parameter values $p_0(i)$ and $p_1(i)$ and applied to p-set p_0. p_1 represents the p-set at the end of the continuous interaction.

III. $p_0(i) \rightarrow p_1(i)|\, p_0 \mapsto p_1$: parameter value $p_1(i)$ was calculated from $p_0(i)$ by some function and then applied to p_0 to generate p_1.

where the $p_j = \{p_j(1), \ldots, p_j(n)\}$ are p-sets, and each $p_j(i) \in P_i$ is a different parameter value for the same parameter type P_i. All derivations are expressed as *parameter-list* | *input-tuple* $\mapsto$ *output-tuple*. Such constructions are *parameter derivation calculus instances*. The parameter list contains input and output parameter values. Input parameter values are used only to derive output parameter values. Output parameter values are applied to the elements in the input p-set tuple to generate the output p-set tuple. In the templates, $p_0(i)$ is an input parameter value while $p_1(i)$ and $p_2(i)$ are output parameter values. It is possible to have multiple input parameter values, output parameter values, p-sets in a input tuple, or p-sets in an output tuple. For example, Figure 1 demonstrates a function derivation with two output parameter values (in braces) and two elements in the output tuple (in angle brackets).

The parameter derivation calculus is the basis for recording the visualization exploration session. Formally, a visualization session consists of a set of *visualization session results*. A visualization session result is a tuple containing a p-set, the visualization result derived from the p-set, a timestamp to place the result in temporal context, and a parameter derivation calculus instance detailing how the result was derived. Example session results are given in Figure 1. Each session result represents the generation of a single visualization result. As the example illustrates, it is possible for parameter calculus instances to be the same for two or more session results (the third and fourth lines in the example). This disparity is due to the fact that calculus instances correspond to user actions while session results correspond to rendered results. The same user action can create more than one rendered result, all sharing the same timestamp. Though it is possible for a user to re-visit the same visualization result by generating the same p-set more than once, each is a unique session result identified by a distinct timestamp.

The information stored within the visualization process is fairly complex. To gain an understanding of visualization sessions—and perhaps a better understanding of the data originally visualized—this information needs to be visualized.

$$\begin{array}{l}
\langle p_0, r_0, t_0, \emptyset\rangle \\
\langle p_1, r_1, t_1, p_1(2)|\, p_0 \mapsto p_1\rangle \\
\langle p_2, r_2, t_2, p_1(1) \rightarrow \{p_2(1), p_3(1)\}|\, p_1 \mapsto \langle p_2, p_3\rangle\rangle \\
\langle p_3, r_3, t_2, p_1(1) \rightarrow \{p_2(1), p_3(1)\}|\, p_1 \mapsto \langle p_2, p_3\rangle\rangle \\
\langle p_4, r_4, t_3, p_4(2)|\, p_0 \mapsto p_4\rangle
\end{array}$$

Figure 1: A series of visualization session results. A session result is a tuple of a p-set (the p_i), the visualization result corresponding to that p-set (r_i), a timestamp (t_i), and information detailing how the result was derived. In this example, the second session result was derived from the first in the second timestep before the third and fourth results were both derived in the third timestep. Afterwards, the fifth result was derived from the first.

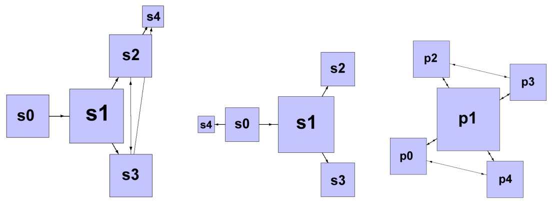


Figure 2: Visualization process graphs for Figure 1. The history metric was used for the left image, the p-set derivation metric for the middle image, and the parameter difference metric for the right image. The s_i are session results while the p_i are p-sets. The graphs provide an "at-a-glance" overview of different aspects of the visualization process.

4 Visualizing the Visualization Process

Given visualization sessions represented using the process model, it is possible to distill different graphical representations of that process. Both [15] and [9] describe different graphs of the visualization process. To the best of our knowledge, neither of these representations have been visualized directly—they have been used for presentation only. Visualizing this information allows one to interactively explore the process information to gain a better "mental map" of the process. For visualization sessions past a trivial size, visualization of the session is much more feasible than direct analysis.

4.1 Visualization Process Metrics

Before the visualization process can be presented visually, the relationships within the process must be extracted. The relationships are represented by (possibly directed) graphs—an edge exists between two session results s_i and s_j if they satisfy some relation. Different types of relations emphasizes different characteristics of the process. Currently four different visualization process metrics are implemented; these metrics determine whether two edges are connected (and the direction of the connection in a directed graph):

- **History Metric** This metric only uses the timestamps of the visualization session results in order to determine result differences. The signed distance between any two results is the difference of their timestamps. For results generated during the same timestamp (due to a operator application in an image graph, for example), an implicit distance of one is assigned in order to keep different results distinct. Process graphs visualized with this metric are directed, with edges indicating time—results generated during the same timestep are connected in both directions.

- **Session Result Derivation Metric** This metric utilizes the parameter calculus instance information associated with a session result to determine distances between results. There is a connection between two session results s_i and s_j if a parameter from s_i was used by s_j in its input or output parameter list or s_i's p-set is a member of s_j's input tuple. In addition, s_i must be the last session result with the p-set p_i recorded before s_j. Due to this restriction, session results with the same p-set but different timestamps correspond to distinct nodes in the directed graph constructed with this metric. There is also a restriction such that session results generated with the same derivation (such as the third and fourth results in Figure 1) are not considered derived from each other.

- **P-set Derivation Metric** The p-set derivation metric is similar to the session result derivation metric, but it does not care about timestamps during derivation. As a consequence, results with the same p-set will be mapped to the same vertex. The distance between any two session results is the fewest number of derivations between any session result with the same p-set. The process graph generated with this metric is thus potentially more compact than the previous graph.

- **P-set Difference Metric** The final metric discussed only considers differences in session result p-sets. The distance between any two session results is the number of differences between parameter values in their respective p-sets. For example, in volume visualization, two session results with identical p-sets except for their color and opacities maps would have a distance of two. Unlike the previous metrics, the process graph induced by this metric is undirected. In addition, the nodes in this graph are p-sets, not session results.

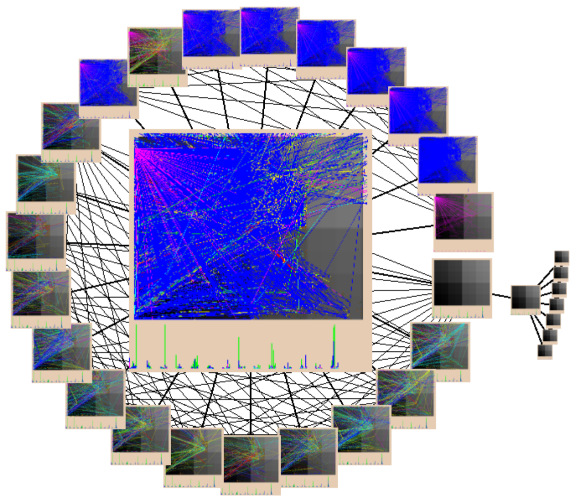


Figure 3: A p-set difference visualization process graph for a Internet router anomaly detection system.

Each metric highlights different aspects of the visualization process (See Figure 2). Though parameter derivation information is not present in the graphs using the history metric, it gives a clear sense of the flow of time during the visualization process. Both the session result and p-set derivation metrics are needed to get a sense of relationships during the visualization process. The session result derivation metric combines the sense of time the history metric provides with a notion of p-set relationship. The p-set derivation metric can collapse some of the process structure, indicating where users returned to previous result (through cycles). Thus, the p-set metric finds where re-visiting occurred while the session result metric displays the temporal order of the re-visiting. Finally, the p-set difference metric gives a sense for the depth of exploration during the process. Shallow spanning trees of graphs using this metric signify a visualization process that did not deeply search the space of parameter values while deep spanning trees could suggest lack of focus. More examples of using visualization process graphs to understand the visualization process are given in section 5.

4.2 Layout and Interaction

After the visualization process graph has been constructed, it is visualized in three steps. First, a focus is chosen. This focus will become the root of a spanning tree in the visualization process graph. Next, this tree is positioned using the distorting radial layout technique we have developed. Finally, a user specified rotation occurs and the graph is rendered. When rendering, children of the spanning tree have their edges thickened. As demonstrated by Figure 3, when the process graph is visualized, the visualization results are also displayed in order to orient the user.

The radial layout deforms the ring radii and node sizes depending on two factors: the depth of the spanning tree from the node and a user controlled focus strength. Initially, the focus strength allocates a quarter of the display radius (half the size of the display area) to the focus node; this becomes the radius of a circle in which the focus node's result will be inscribed. The remaining length along the display radius is split among the spanning tree levels geometrically:

$$n_i = (1 - f)r_d/2^i$$

where i is the ring level (level in the spanning tree), r_d is the display radius, f is the focus strength (between 0 and 1), and n_i is

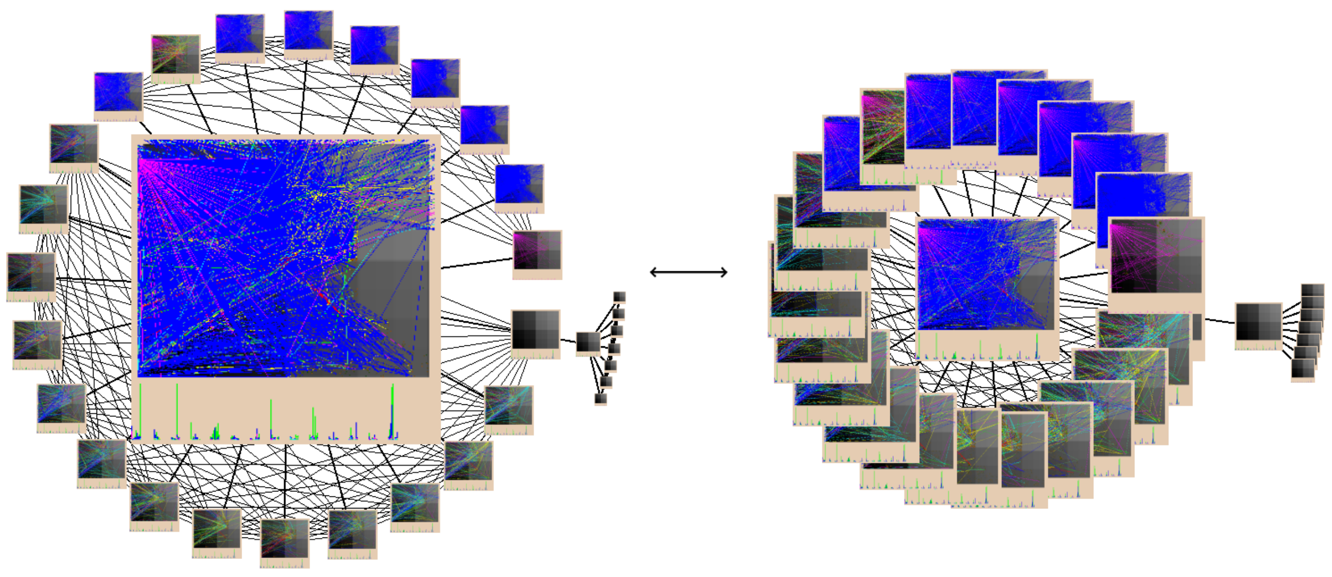


Figure 4: Example of deforming the radial layout. As the focus element is de-emphasized, the ring radii and node sizes increase.

the radius of the circle that all graph elements for that level will be inscribed (the node radius for that level). For simplicity, node sizes smaller than a given threshold are clamped to that threshold. After the node sizes are calculated, the radius of each ring is determined. The radius of the first ring is the sum of the focus node's size and the node size for the first level. Subsequent ring radii are the previous ring's radius plus the node sizes for the current and previous levels. When the focus or focus strength changes, the layout is recalculated. The effect of changing the focus strength is demonstrated by Figure 4. As the focus strength increases, the rest of the graph (the context) is pushed to the periphery of the layout. Decreasing the focus strength allows more detail to be given to the ring levels as the node size increases.

Besides controlling the focus strength, the user can also change the orientation of the process graph by rotating the rings around the focus. Also, to facilitate exploration of the process graph, the user is able to change focus by selecting a node. Finally, since the visualization result does not by itself signify the visualization parameters that generated the result, the user can also bring up a dialog that displays a result's parameter values. Combined, these facilities enable effective exploration of the process graph.

5 Examples

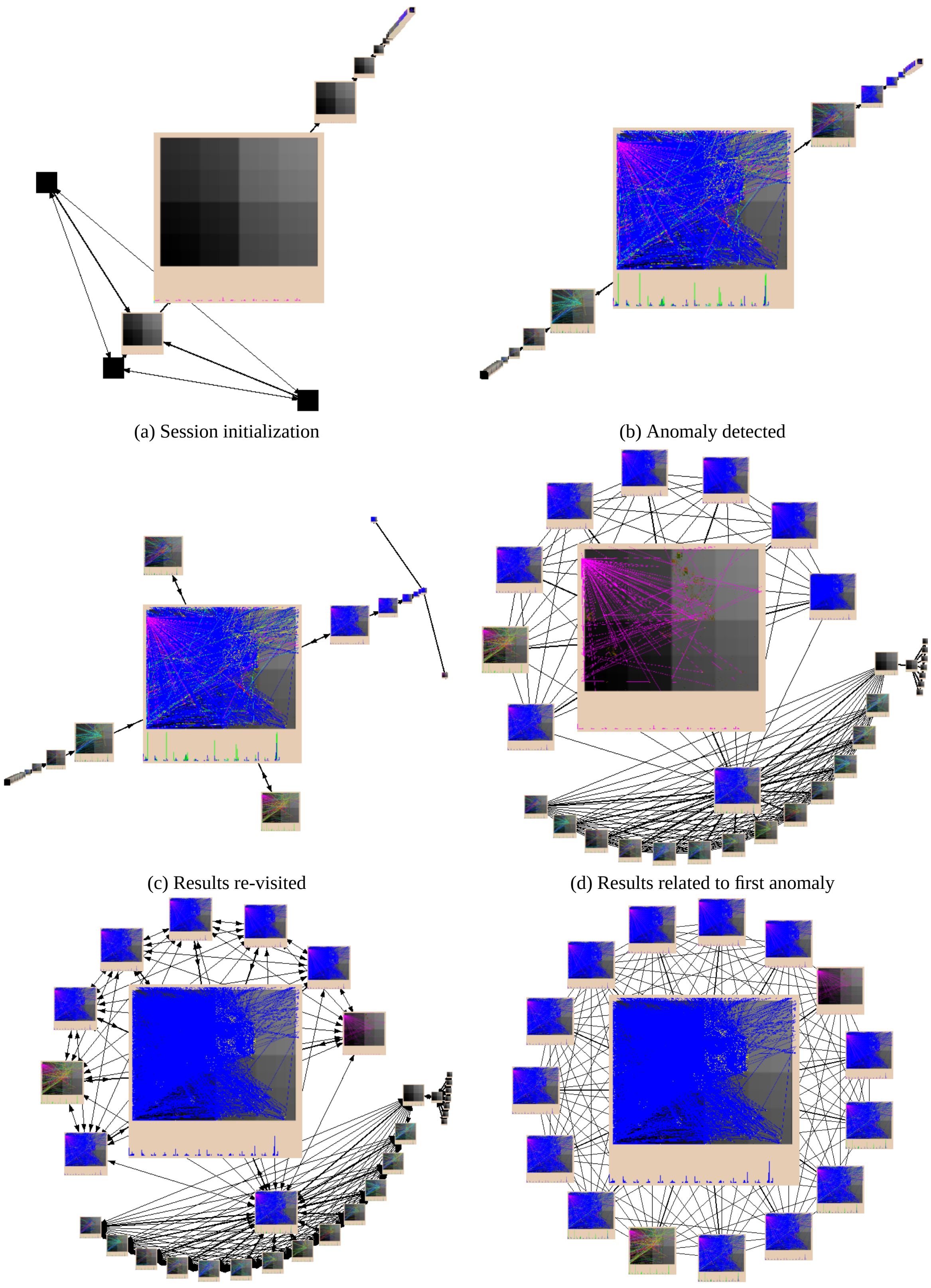
To demonstrate the visual analysis of the a visualization process encapsulated by our model, visualization sessions using a tool to visualize Internet routing anomalies were recorded. The tool displays different types of changes to ownership of autonomous systems (ASes)—groups of hosts on the Internet. The different types of changes correspond to different colored lines or cubes depending on the visualization used. The tool allows a user to browse through dates with different types of AS changes highlighted. Anomalies are found by visually searching the dates for unusual patterns. In this example, a visualization session examining a routing anomaly on August 14, 2000 was used.

Initially, a history metric process graph was used in order to follow the sequence of exploration (Figure 5a). The black squares correspond to results for the default parameter values for the different visualization transforms—in this case, these results were never explored so no result was stored for them. All three of these default results (for the three different transforms used by the visualization tool: planar, 3D, and fish-eye) were created during the same timestep; this explains the edges between them. The user then followed the directed graph edges until they reached the first result corresponding to the target date of August 14 (Figure 5b). Comparing the focus element to some subsequent elements in the graph, it became apparent that some of the results were similar to the focus result (for example, the second child of the focus towards the upper right of 5b). By changing to a view using the p-set derivation metric (Figure 5c), the reason for this similarity becomes clear. The in- and out-going directed edges from the focus result indicate that the focused result was visited several times during the visualization session. This suggests the user may have been unfamiliar with the mapping of color to AS change, and thus had to "explore the possibilities" in order to determine which change type caused the anomaly. A parameter difference metric graph was then used to distinguish the different anomalies that occurred from each other (Figure 5d-e). Finally, a specialized parameter difference metric—one that only measures differences in displayed ASes–was used on one of the anomalies (Figure 5f). All nine of the the focus' neighbors in Figure 5e are present in Figure 5f, thus identifying all those results as displayed AS changes. Similar specialized metrics can be used on other graphs to extract parameter-value specific information.

6 Conclusions

A technique for visualizing the visualization process has been presented using a novel focus+context radial graph algorithm. Our method allows a user to examine the details of a visualization result while keeping the context of the visualization process visible. Four different metrics examining different parts of the visualization process were presented, and an in-depth example demonstrated the uses of this kind of visualization.

There are several potential applications for the work presented here. Visualizations of the visualization process give insight into the process that can be used in different ways. For example, the analysis of the example suggests that a more intuitive mapping



(a) Session initialization

(b) Anomaly detected

(c) Results re-visited

(d) Results related to first anomaly

(e) Results related to second anomaly

(f) AS changes from (e)

Figure 5: A series of visualization process graphs for a session using a Internet router anomaly detection tool. Each highlights a different portion of the exploration process.

of AS change type to color, a more effective parameter manipulation control, or better user training could be needed to remove the re-rendering cycle discovered. In addition, collaborators can use such visual representations of the visualization process to familiarize themselves with previous explorations. This exploration could in turn suggest further avenues of pursuit with the data under study. The deformable radial layout discussed here could also be applied to other graph visualization applications where focus+context is desired.

6.1 Future Work

Several enhancements to this work could be implemented. Primary among these is better visualization parameter display. Though a separate display of parameters is currently available, an in-place, iconic display of parameters is preferable. Such a display would assist in providing context for the results.

Different aspects of the layout algorithm could also be further investigated. One consequence of the technique for dense graphs is that results can become occluded. A dynamic layout to un-occlude selected results would help solve this problem. In addition, a layout algorithm that allows multiple foci is desirable. For example, selected rings in the layout could be expanded to fill more space while the other rings (and their contents) are shrunk—this would also help mitigate the occlusion issue. One potential solution is to combine some of the space distortion techniques cited with our layout algorithm to address these issues.

One of the eventual goals of this project is to extend this work into a full visualization exploration system. In addition to visualizing current and previous results, this future system would allow new results to be rendered and added to the display. We also wish to implement parameter operators that propagate through the graph in a manner similar to that of the image graph. Such a system would be a powerful tool for visual exploration.

Acknowledgments

This work was supported by the National Science Foundation under contracts 9983641 (PECASE Award) and ACI 9982251 (LSS-DSV program), and Lawrence Livermore National Laboratory under ASCI ASAP Level-2 Memorandum Agreements B347878 and B503159. We would like to thank the members of the UC Davis Visualization and Graphics Research Group for their input and assistance, especially Soon Tee Teoh who provided us with the Border Gate Protocol visualization tool.

References

[1] Stuart K. Card, Jock D. Mackinlay, and Ben Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, 1999.

[2] M. Sheelagh T. Carpendale, David J. Cowperthwaite, F. David Fracchia, and Thomas C. Shermer. Graph folding: Extending detail and context viewing into a tool for subgraph comparisons. In Franz J. Brandenburg, editor, *Proc. 3rd Int. Symp. Graph Drawing, GD*, number 1027, pages 127–139, Berlin, Germany, 20-22 1995. Springer-Verlag.

[3] G. di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1999.

[4] Peter Eades. Drawing free trees. *Bulletin for the Institute for Combinatorics and its Applications*, 5:10–36, 1992.

[5] Arno Formella and Jorg Keller. Generalized fisheye views of graphs. In Franz J. Brandenburg, editor, *Proc. 3rd Int. Symp. Graph Drawing, GD*, number 1027, pages 242–253, Berlin, Germany, 20-22 1995. Springer-Verlag.

[6] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, January-March 2000.

[7] Ivan Herman, Guy Melanon, M. M. de Ruiter, and M. Delest. Latour – A tree visualisation system. In *Proceedings of the Symposium on Graph Drawing '99*, pages 392–399, 1999.

[8] T. J. Jankun-Kelly and Kwan-Liu Ma. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Transactions on Visualization and Computer Graphics*, 7(3):275–287, July/September 2001.

[9] T. J. Jankun-Kelly, Kwan-Liu Ma, and Michael Gertz. A model for the visualization exploration process. In *Proceedings of IEEE Visualization 2002*, 2002. (To Appear).

[10] T. Keahey and E. Robertson. Techniques for non-linear magnification transformations. In *Proceedings of the IEEE Symposium on Information Visualization '96*, pages 38–45, 1996.

[11] M. Kreuseler, N. López, and H. Schumann. A scalable framework for information visualization. In *Proceedings of the IEEE Symposium on Information Vizualization*, pages 27–36, 2000.

[12] Matthias Kreuseler and Heidrun Schumann. Information visualization using a new focus+context technique in combination with dynamic clustering of information space. In *Workshop on New Paradigms in Information Visualization and Manipulation*, pages 1–5, 1999.

[13] John Lamping and Ramana Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *J. Visual Languages and Computing*, 7(1):33–55, March 1996.

[14] John Lamping, Ramana Rao, and Peter Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proc. ACM Conf. Human Factors in Computing Systems, CHI*, pages 401–408. ACM, May 1995.

[15] John Peter Lee and George G. Grinstein. An architecture for retaining and analyzing visual explorations of databases. In *Proceedings of IEEE Visualization '95*, pages 101–108, 1995.

[16] Kwan-Liu Ma. Image graphs - a novel approach to visual data exploration. In David Ebert, Markus Gross, and Bernd Hamann, editors, *Proceedings of IEEE Visualization '99*, pages 81–88, 1999.

[17] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In *Proceedings of SIGGRAPH97*, pages 389–400, 1997.

[18] G. Melançon and I. Herman. Circular drawing of rooted trees. Technical report, Centre for Mathematics and Computer Sciences, 1998. http://www.cwi.nl/InfoVisu/Papers/circular.pdf.

[19] Tamara Munzner. H3: Laying out large directed graphs in 3D hyperbolic space. In L. Lavagno and W. Reisig, editors, *Proc. IEEE Symp. Information Visualization*, pages 2–10, 20–21 October 1997.

[20] Tamara Munzner. Drawing large graphs with H3Viewer and Site Manager. In *Graph Drawing*, pages 384–393, 1998.

[21] Tamara Munzner. Exploring large graphs in 3D hyperbolic space. *IEEE Computer Graphics and Applications*, 18(4):18–23, July/August 1998.

[22] Manojit Sarkar and Marc H. Brown. Graphical fisheye views of graphs. In Penny Bauersfeld, John Bennett, and Gene Lynch, editors, *Human Factors in Computing Systems, CHI'92 Conference Proceedings: Striking A Balance*, pages 83–91. ACM Press, May 1992.

[23] Manojit Sarkar and Marc H. Brown. Graphical fisheye views. *Communications of the ACM*, 37(12):73–84, 1994.

[24] M. Sheelagh, T. Carpendale, David J. Cowperthwaite, and F. David Fracchia. 3-dimensional pliable surfaces: For the effective presentation of visual information. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, Information Navigation, pages 217–226, 1995.

[25] Craig Upson, Thomas A. Faulhaber, Jr., David Kamins, David Laidlaw, David Schlegel, Jeffrey Vroom, Robert Gurwitz, and Andries van Dam. The Application Visualization System: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 9(4):30–42, July 1989.

[26] Graham J. Wills. NicheWorks — interactive visualization of very large graphs. *Journal of Computational and Graphical Statistics*, 8(2):190–212, 1999.

[27] Ka-Ping Yee, Danyel Fisher, Rachna Dhamija, and Marti Hearst. Animated exploration of dynamic graphs with radial layout. In *Proceedings of IEEE Symposium on Information Visualization 2001*, pages 43–50, 2001.