Prov Viewer: A Graph-Based Visualization Tool for Interactive Exploration of Provenance Data

Troy Kohwalter[1(✉)], Thiago Oliveira[1], Juliana Freire[2], Esteban Clua[1], and Leonardo Murta[1]

[1] Instituto de Computação, Universidade Federal Fluminense, Niterói, RJ, Brazil
tkohwalter@ic.uff.br
[2] New York University, New York, NY, USA

Abstract. The analysis of provenance data for an experiment is often crucial to understand the achieved results. For long-running experiments or when provenance is captured at a low granularity, this analysis process can be overwhelming to the user due to the large volume of provenance data. In this paper we introduce, *Prov Viewer*, a provenance visualization tool that enables users to interactively explore provenance data. Among the visualization and exploratory features, we can cite zooming, filtering, and coloring. Moreover, we use of other properties such as shape and size to distinguish visual elements. These exploratory features are linked to the provenance semantics to ease the comprehension process. We also introduce collapsing and filtering strategies, allowing different levels of granularity exploration and analysis. We describe case studies that show how *Prov Viewer* has been successfully used to explore provenance in different domains, including games and urban data.

Keywords: Provenance · Visualization · Graph · Analysis · Tool

1 Introduction

Displaying provenance is an issue in present times. While there are many tools that capture data provenance, most of them offers basic visualizations or requires exporting the data to a format compatible with existing visualization tools. Ideally, provenance data are captured and stored for each task of the scientific workflow in terms of basic relationships among individual tasks. More complex relationships are inferred during an analysis process [5] by using visualization techniques to facilitate the understanding of the captured data, especially when dealing with complex workflows.

However, a provenance graph might contain data that did not cause any significant change or provides information that might not be related to the desired analysis. Another problem occurs when analyzing provenance data that is segregated into different trials (workflow executions), which results in analyzing multiple individual graphs to extract meaningful knowledge since each trial is normally represented by a separate graph. Moreover, a common representation of the provenance graph is based on traditional node-link diagrams. These diagrams may impair the analysis process depending on the size and domain peculiarities of the graph. Using simple node-link

M. Mattoso and B. Glavic (Eds.): IPAW 2016, LNCS 9672, pp. 71–82, 2016.
DOI: 10.1007/978-3-319-40593-3_6

diagrams to represent provenance data can also harden the graph understanding when dealing with the wealth of information that can be contained in a single provenance node, even when using the different shapes to distinguish the information.

Although there are some tools in the literature for provenance analysis [2, 8, 18, 19], they are based on these simple node-link diagrams with only basic visualization features, such as labels and colors to distinguish edges and vertices, neighbor detection, and size for different intensities. Moreover, they are not directly compatible with the PROV model [17], requiring additional steps to convert the data.

In this paper, we introduce *Prov Viewer*, a novel graph-based visualization tool for interactive exploration of provenance data that is compatible with the PROV, and consequently, with any other application that exports provenance using the PROV model. *Prov Viewer* processes the collected provenance data to generate an interactive provenance graph to provide advanced visualization features for identifying steps and contributors to a given result.

The PROV model has proved to be useful for other domains besides scientific workflows, such as electronic games or urban data. Thus, in a previous work [13], we introduced the usage of digital provenance in games and proposed a conceptual framework (PinG) for collecting information during a game session, mapping them to provenance terms, and providing the means for a post-game analysis. We experimented with rudimentary ways to visualize provenance in the context of games [12]. We developed an initial prototype for provenance visualization and used it on the SDM [15] game to assess whether provenance data visualization can be helpful in the understanding of game events [14]. This prototype was also used in another application that extracts game provenance through image processing mechanisms [11]. More recently, it is being used to visualize provenance from urban data.

The tool we present in this paper is the result of several extensions and new techniques we developed to address issues encountered in different scenarios. We have designed new visual representations and interaction mechanisms that address many of the aforementioned challenges: (1) collapsing, highlighting the relevant information in the graph; (2) filtering, removing information that is not relevant for a given analysis; (3) graph merge, integrating the analysis of multiple trials; (4) specialized layouts, organizing the graph in a more understandable way; (5) domain configuration, customizing the visualization for specific needs; (6) shapes, sizes, and colors, supporting a clear distinction of information types, and (7) interoperability, supporting PROV-N for importing provenance data.

This paper is organized as follows: Sect. 2 presents some of the related work in the area of provenance visualization. Section 3 details our provenance visualization tool, *Prov Viewer*. Section 4 presents two different case studies using our tool. Section 5 concludes this work by listing some future work.

2 Related Work

Our related work can be grouped into two categories: workflow management systems that have built-in provenance visualizations and standalone provenance visualization tools. The workflow management systems that have build-in provenance visualizations

[1, 4, 10] allow easy integration between provenance collection and analysis. However, they have a shortcoming of not supporting provenance data generated by other workflow management systems or standalone provenance gathering tools, even when they are compatible with well known provenance models. Furthermore, workflow management systems normally lack graph manipulation features for viewing provenance graphs.

On the other hand, there are some standalone provenance visualization tools that resemble our work. Provenance Explorer [5] takes RDF-based provenance outputs from capture systems and dynamically generates customized views of provenance trail. However, it focuses on provenance data and inference rules associated with processing events in a laboratory or manufacturing plant, lacking the support for data processing activities in the digital domain. Furthermore, their provenance model is based on the ABC ontology model. Lastly, their collapse feature only supports one expansion level, instead of multiple levels of detail.

The ZOOM [3] prototype provide users with an interface to query provenance information generated by a workflow system through SQL queries. An interesting aspect is that it allows the user to dynamically modify the graph by hiding irrelevant information, updating the provenance graph for the new view. Another existing tool is the PROV Toolbox, which converts W3C PROV data model representations. However it lacks a built-in visualization and requires the use of a generic graph tool (*Graphviz*) to visualize the provenance data. Another similar tool is PROV Translator, which validates PROV representations and translates them to other representations. It also provides graph visualizations based on their previous work [7], which displays a provenance graph using PROV's vertex shape and color to identity the vertex type. The PROV-O-Viz [9] tool is a web-based visualization tool for provenance based on PROV that uses *Sankey Diagrams* for visualization. *Sankey Diagrams* are used to visualize flow magnitude between nodes in a network and in PROV-O-Viz the activity or entity width is based on the information flow.

Some related work is limited to specific domains (*i.e.*, Provenance Explorer), require additional knowledge (*i.e.*, ZOOM, PROV-O-VIZ), or they are not compatible with provenance data from other tools (*i.e.*, Kepler, VisTrails, Taverna). Furthermore, they individually provide some interesting features, such as interactive graphs, level of detail, summary nodes, merges, and filters. However, these approaches do not provide these features in an integrated way, hindering the analysis due to visualization and manipulation restrictions, which sometimes require additional external procedures to analyze the data. Moreover, they lack any means of overlaying provenance information onto a spatial structure for analysis.

3 Prov Viewer

In this paper we present a provenance visualization tool named *Prov Viewer* (Provenance Viewer[1]). Our tool is compatible with the PROV-N notation, allowing its adoption in different domains and applications. The provenance data, which contains

[1] *Prov Viewer* is available at https://github.com/gems-uff/prov-viewer.

the provenance information among entities and their relationships, is processed to generate a provenance graph. This graph is a visual representation of the provenance data and supports user interaction, which is a key feature for understanding how each action influenced in the outcome and how they influenced each other. It is also possible to manipulate the graph by omitting facts and collapsing chains of actions for a better understanding and visualization experience. No information is lost in this process, so that the user can undo any changes made during analysis.

Prov Viewer uses the PROV notation, where square vertices represent activities, circles represent entities, and pentagons represent agents. Furthermore, each vertex is composed of multiple attributes that describe the vertex. Each attribute contains a name and a value that is associated with it (*e.g.*, startTime: 2012-05-25T11:15:00, endTime: 2012-05-25T12:00:00). The edges in the provenance graph represent the relationships between vertices. As such, activity vertex can be positively or negatively influenced by other vertex and have relationships with entities and agents.

Before using *Prov Viewer*, it is necessary to configure it to understand the domain peculiarities and customize the visualization features. This is accomplished by creating a config.xml file based on the configuration schema of *Prov Viewer*. This configuration file allows the user customize the graph visualization. *Prov Viewer* also has a feature for automatic detection and configuration for each edge type and color scheme, which represents most of the configuration effort of the tool, according to the graph being used. Note that the user will need to manually input specific parameters in the configuration xml in order to use some of our tool layouts. However, this task is done only once for a new domain.

Figure 1 shows the high-level architecture of our tool, illustrating some of its features available that allow users to interact with the provenance data to identify relevant actions that impacted in the results. The following sub-sections describe the most relevant features, including shapes and colors to distinguish information, manual collapses, graph merges, layouts, and an automatic collapse feature.

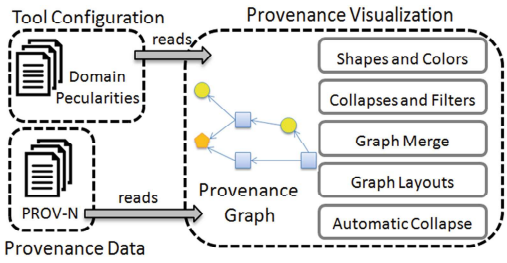

Fig. 1. *Prov Viewer's* high-level architecture (Color figure online)

3.1 Shapes and Colors

Prov Viewer builds its visualization strategy based on shapes and colors, for both vertices and edges. Shapes are used to map semantic concepts from the provenance and colors are used to map scalar values, such as intensity or orientation. The vertex shape

is directly related to provenance semantics (*i.e.*, agent, activity, entity), while the vertex color is used for mapping scalar values through the usage of a color scheme. When selecting the desired attribute, all vertices with the specified status have their colors changed according to their respective values. We adopt the traffic light scale [6], which indicates the status of the variable using gradients from three colors: red, yellow, and green. The resulting color is automatically inferred from minimum and maximum values for that attribute or using boundaries manually specified by the user in the configuration xml. Enabling this type of feature allows the user to easily identify situations where the desired attribute value fluctuates throughout the data.

Both the edge shape (*i.e.*, thickness) and its color are used to show the intensity of the relationship. The intensity is the value associated with the edge, if any, and is more common on influences (*i.e.*, *wasInfluencedBy*). A thin edge with a darker color represents a low influence relationship (*i.e.*, the assigned value to the edge is low). On the other hand, thicker and brighter edges represent a strong or intense relationship. Figure 2a shows an example of edges with different colors and thickness and Fig. 2b shows the vertex color based on their time values (also represented by columns). This feature can be used to quickly identify strong influences in the graph just by looking at the edge's thickness and brightness. The edge's color is also used to represent any additional numeric information contained in the relationship (*e.g.*, influences that has numeric data), which can be any of these three types: positive, which is represented as green and indicates an increase in the numeric value (*i.e.*, when the edge has an associated positive value); negative, which is represented as red and indicates a decrease; and neutral, which is represented as blue and indicates no numeric chances.

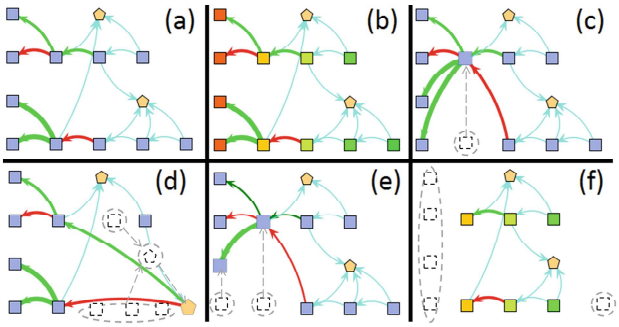

Fig. 2. (***a***) Original graph; (***b***) graph with a color schema; (***c***) collapse of two activities; (***d***) collapsing of the agent's activities; (***e***) graph *c* after another collapse; and (***f***) temporal filter (Color figure online)

3.2 Collapses and Filters

Our tool provides a vertex collapse feature to aid in the analysis of the graph and allows a manual collapse of selected vertices in order to compact the graph size, grouping the selected vertices together in a single summarized vertex. No information is lost in this process and it can be reverted by the user. Figure 2c shows an example of collapsing activity vertices. The grey markings represent vertices before the collapse and the grey

arrow represents where they were collapsed to. Another usage of the collapse feature is to group activities related to the same agent, allowing the user to see all the influences and changes that the agent did throughout his tasks. Figure 2d shows such an example. The size of the collapsed vertex is bigger than the rest due to the number of vertices in the collapse group (and is proportional to the number of vertices). Furthermore, the shape is the same as that of an agent vertex because there is an agent vertex in the collapse group.

The summarized information is displayed as follows: For String values, it shows all different values separated by a comma (*e.g.*, *String_Value1*, *String_Value2*, *String_Value3*). For attributes with numeric values in a collapse group composed of two vertices, the tool shows the average value for that attribute followed by the minimum and maximum values. Otherwise (collapse group containing more than two vertices), the tool displays the average value followed by the five-number summary (minimum value, 1st quartile, median, 3rd quartile, maximum value).

Similar edges (*i.e.*, same type) that have the same target and type are also grouped together when collapsing vertices. The collapsed edge's information (*i.e.*, color, thickness, and value) is computed by summing or averaging the values of the participating edges, depending on their type. For example, *Prov Viewer* can use the *sum* function for edges representing expenses and the *average* function for edges representing percentage. However, the user needs to parameterize each edge. Otherwise the tool will always use the default *sum* function. Figure 2e shows an example of collapsing edges that occurred when collapsing another group of vertices after collapsing the vertices in Fig. 2c. Note that the colors for each edge changed after the collapse due to the new maximum value (from the sum of the collapsed edges).

The tool also offers another simple vertex filter based on temporal information. The user defines the desired temporal range (*e.g.*, start time and end time) for visualization and the tool hides all vertices that are outside the selected range. Figure 2f shows an example of the temporal filter, hiding vertices from Fig. 2a with time (which can be seen by the rows) greater than four and less than two. *Prov Viewer* also has an edge filter, which filters edges by context (*i.e.*, label) or by the type of relationship.

3.3 Graph Merge

Our provenance visualization tool also proposes a feature based on [16] to merge two provenance graphs in order to generate a single unified provenance graph. The merge process combines the current displayed graph with another graph (chosen by the user) to generate a single unified graph for visualization. The merge process is composed of four steps: (1) vertex matching, which selects pairs of vertices from the graphs; (2) similarity verification between vertices, which receives two vertices from the first step and informs if they are similar. Vertices are considered similar if they belong to the same vertex type and have the same properties with similar numeric values within a configurable margin of error; (3) merge vertices that were considered similar in the previous step; and (4) creation of the unified graph for visualization, which only occurs after the matching process is over. The resulting graph can be exported using the PROV-N notation for future usage.

Figure 3 illustrates the graph merge of two distinct graphs from the same domain. Red vertices in the merged graph (Fig. 3c) belong exclusively to the first graph (Fig. 3a), while grey vertices represent common vertices (*i.e.*, merged vertices) from both graphs, and green vertices belong exclusively to the second graph (Fig. 3b). This graph merge feature is useful when analyzing multiple sessions or trials by detecting common sections. Merged vertices from this feature also provide similar summarized information using the five-number summary.

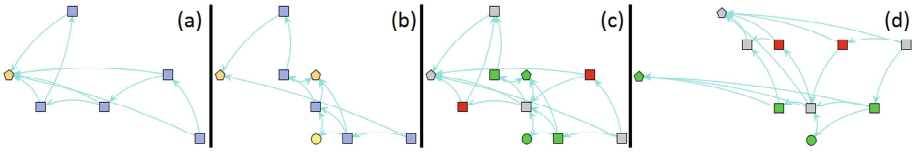

Fig. 3. Two graphs (***a*** and ***b***) merged into a single graph (***c***) and with a temporal layout (***d***) (Color figure online)

3.4 Graph Layouts

Our visualization tool allows the user to interactively change the graph layout to better visualize the result. We created two provenance graph visualization layouts: temporal and spatial.

The **temporal layout** organizes the graph in a chronological order similar to a timeline (or spreadsheet) for each agent. Thus, each timeline (or line) of the graph groups activities of the same agent and each column in the graph represents the passage of time. This makes easier to know the entity or agent responsible for executing each activity by just looking at the agent responsible for that line. Invalid temporal values are considered as zero for the purpose of positioning them in the graph. Thus, the graph positions the vertices in the x axis according to the chosen scale. Figure 3d illustrates an example of our temporal layout, displaying the graph similarly to a spreadsheet and organizing the vertices by their chronological order. Note that now it is much easier to identify the agent responsible for each activity as the leftmost nodes and their chronological order by looking at the activity's placement in the graph. The horizontal position represents the time axis (passage of time) and the vertical position represents the agent axis (the responsible for the activity).

The **spatial layout** organizes the vertices in the graph by their spatial coordinates and can be used for spatial or georeferencing the data. The layout also supports the usage of an orthographic image or maps taken from *Google Maps* and *OpenStreetMaps* as background for the graph. This is particularly useful for corresponding elements with other graphical representations, such as a map of a city or a game scene. When dealing with real world maps, *Prov Viewer* automatically transform the latitude and longitude to pixel coordinates. This automatic process requires only three input values to correctly align the image with the graph and is described in the tool's documentation. When using the spatial layout in conjunction with a background image, the user sees where each event occurred just by looking at the graph's placement in the image.

3.5 Automatic Collapse

Prov Viewer offers an automatic collapse feature to highlight the relevant information in the provenance graph by summarizing the tracked data. This automatic collapse groups sequential information that has similar values or represents the same state, allowing the user to quickly identify relevant information or state transitions. This process is similar to data deduplication, which hides duplicate copies of repeating data. Currently, this form of collapse uses only one vertex attribute to define similarity. However, unlike normal deduplication, the process in *Prov Viewer* is reversible without any kind of information loss since it occurs only in the visualization process.

The collapse process compares each vertex with its neighbor to omit similar states. If the vertices' values are similar for the specific attribute being analyzed, then they can be collapsed into a single vertex. Vertices are considered similar if they are neighbors and their values for the specified attribute are within one standard deviation difference. Since the goal is to combine all similar states, it is necessary to go beyond the vertex direct neighbor. If any of the vertices in the collapse group have an edge to a vertex outside of the collapse group and that vertex has a similar attribute value to those in the group, then it is added to the collapse group. Thus, the collapse group will keep growing until a significant change of state is detected. Figure 4 illustrates an example of our similarity collapse for the gathered provenance data from a racing car doing multiple laps in a racing-track. Sequential vertices that had similar speeds were collapsed into a single vertex. Note that the collapse preserves the notion of different laps (differently from the merge feature) and that the collapsed vertices (with varying sizes according to the number of collapsed vertices) have similar speeds. This allows the analyst to detect behavior patterns and locations where the driver had more difficulties.

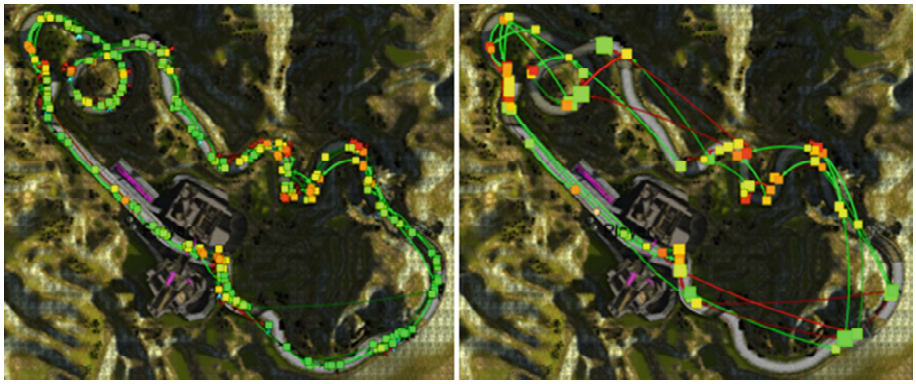

Fig. 4. Similarity collapse (right) according to the speed of the car from the original graph (left) (Color figure online)

4 Case Studies

As discussed in the introduction section, we adopted *Prov Viewer* in two different provenance domains: digital games and urban data. In the game domain, our visualization tool was used for the analysis of game sessions of five different games (*SDM*, *Super Mario World*, *Unity's Tower Defense*, *Unity's Angry Bots*, and *Unity's Car Tutorial*) and for the urban data domain our tool was used to analyze bus traffic data from the city of Rio de Janeiro. More than thirty people, including researchers and students, used *Prov Viewer* to analyze provenance data, where 29 were students analyzing data gathered from a serious game [14]. In the following sub-sections we present two case studies of our visualization tool.

4.1 Game Session Provenance

The first case study is the *Angry Bots* game, an open-source demo from Unity asset store. The provenance graph used in this example contains 1275 vertices and 2976 edges. Figure 5 illustrates one of the possible visualizations of the provenance graph from the game using our visualization tool, which was captured using the PinG approach. The tool was configured to include the background rendering from the scene.

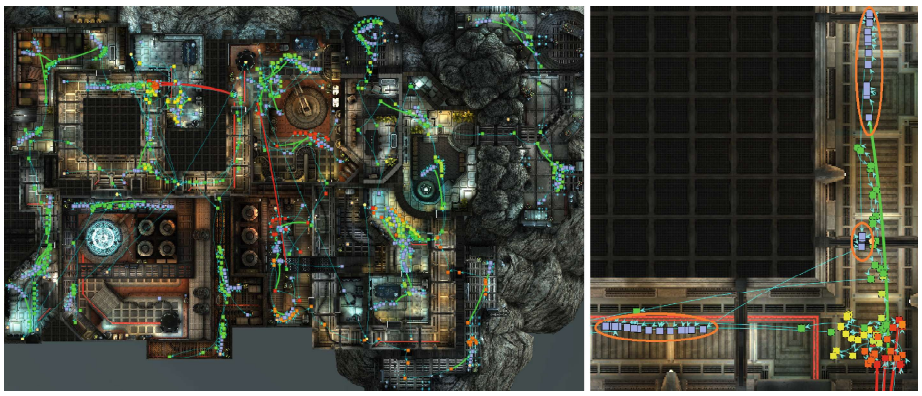

Fig. 5. Provenance graph (left) rendered with *Prov Viewer* from the *Angry Bots* game using spatial layout. The second graph (right) is a zoomed section rotated by 90º. (Color figure online)

The displayed graph is using the vertex visualization schema for the player's health attribute values (vertex color using traffic light schema: green, yellow, and red). The green and red edges respectively represent the influences that changed the player's experience points (awarded when defeating an enemy) and moments when the player died (red vertex from the edge) and was moved to the restoration unit to be revived (green vertex from the edge). Blue vertices represent activities executed by other characters in the game (*i.e.*, enemies) and blue edges represent chronological order of events or state change in scene objects (*e.g.*, terminals, doors). By analyzing the

visualization from Fig. 5, we can see the chronology of events, regions visited by the player, sections where more action happened, where the player engaged in battle, when he/she suffered heavy health loss (vertex color change), and the player's health during each section. For instance, we can infer that the cause of the player death in the middle of the map (near the circle-shaped object that represent a storage cylinder) was because he rushed through the level without waiting to recover health from his small health regeneration trait (sequence of orange followed by red vertices) after most engagements. Other three deaths (three "respawn" red edges in the bottom, leaving the second graph of Fig. 5) were caused by three different engagements (red circles) with the same enemy (blue vertices) in close quarters, leaving the player without enough room to maneuver to dodge the explosive attacks. In this game, we used a screenshot of the scenario with the dimensions of 4,280 × 3,208 for the background, which precisely matches the locations where the events (*i.e.*, activities). This high resolution allows for a higher detail of the game scene when zooming the graph during analysis.

4.2 Bus Traffic Provenance

The second case study is based on bus traffic data analysis in the city of *Rio de Janeiro*. The data used in this research, which includes geographic location tracked from the buses' GPS, is obtained from *DataRio*[2]. *Prov Viewer* is being used in this context to render the data for analysis, allowing the research team to understand the wealth of tracked information. Our tool allows for filtering the data to focus on specific buses or relate the bus delays with ongoing events in the city through their geographic location, speed in the region, and timestamp.

Figure 6 illustrates one of the possible visualizations of the provenance data using *Prov Viewer*. The graph contains 601 vertices and 600 edges. The displayed graph is using a color schema based on the bus speed. Therefore the vertices, which represent on-line GPS information tracked from buses at every minute interval, are colored from red to green according to their (instantaneous) speed at that specific moment, while the blue edges link these vertices in a chronological order. Note that the displayed graph is showing bus data from nine different busses from the same route within a period of two hours.

In the graph from Fig. 6, we can see the buses routes through the city and their respective speeds along the way. Furthermore, we can see that the traffic is better in the region near "*Botafogo*" due to the high concentration of green vertices than "*Urca*" and "*Copacabana*". This type of graph visualization allows the user to quickly indentify the streets where the buses moved slower due to traffic by finding regions in the graph with reddish vertices by looking at the vertex location in the map. Moreover, it is also possible to better understand the extension of the traffic jam and the affected areas by crossing the displayed graph with a graph from another route (*e.g.*, merge the graphs) that also use segments of the same street.

[2] *DataRio*: http://data.rio/dataset.

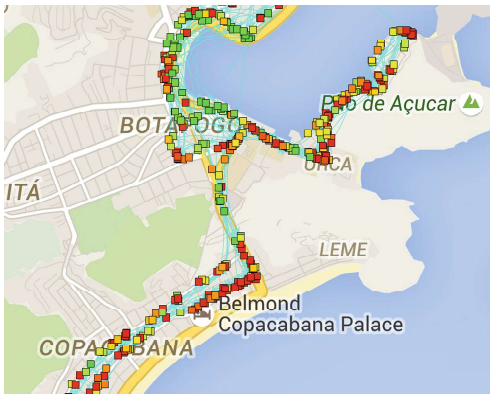

Fig. 6. Provenance Graph rendered on *Prov Viewer* from collected bus traffic data (Color figure online)

5 Conclusions

Graph visualization strategies bring problems related to scalability when dealing with provenance datasets beyond a few hundred nodes. Traditional node-link diagrams can easily become too visually cluttered when dealing with huge provenance data, limiting the user's ability to thoroughly analyze and explore the data. To deal with this problem, *Prov Viewer* offers collapse options that can generate different levels of detail and graph layouts to sort the data and reduce node clustering. *Prov Viewer* also has some basic automatic collapses based on vertex similarity and graph merges, allowing users to omit data and combine different graphs for analysis. Other contributions include the merging of PROV-N data from different files and georeferencing capabilities for provenance information.

Our tool can be configured and used by different provenance applications as a general-purpose provenance visualization tool as it supports graphs that use the PROV-N notation. *Prov Viewer* also supports pre-processing steps, which can be done outside the tool, as long as the final data format is compatible with the PROV-N notation. We showed two case studies from different domains: analysis of digital game sessions and bus traffic data. In both cases, the graph had more than one thousand artifacts (vertices and edges combined). However, the performance starts to degrade after rendering graphs with more than two thousand visible artifacts (*i.e.*, long loading times). Future work includes more complex algorithms to analyze the provenance data and suggest which information can be omitted to reduce the graph to acceptable sizes; more types of graph visualization techniques; summarization techniques; and more graph layouts, including a support for dynamically loading new layouts, or handcrafted ones, in the tool at run-time. We are also working on optimizing the tool to be able to handle rendering huge graphs more smoothly and reducing loading time.

Acknowledgment. The authors thank CAPES, CNPq, and FAPERJ for the financial support.

References

1. Altintas, I., et al.: Kepler: an extensible system for design and execution of scientific workflows. In: Proceedings of 16th International Conference on Scientific and Statistical Database Management, 2004, pp. 423–424 (2004)
2. Bastian, M., et al.: Gephi: an open source software for exploring and manipulating networks. In: Third International AAAI Conference on Weblogs and Social Media (2009)
3. Biton, O., et al.: Querying and managing provenance through user views in scientific workflows. In: IEEE 24th International Conference on Data Engineering, ICDE 2008, pp. 1072–1081 (2008)
4. Callahan, S.P., et al.: VisTrails: visualization meets data management. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, pp. 745–747. ACM, New York (2006)
5. Cheung, K., Hunter, J.: Provenance explorer – customized provenance views using semantic inferencing. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) ISWC 2006. LNCS, vol. 4273, pp. 215–227. Springer, Heidelberg (2006)
6. Diehl, S.: Software Visualization: Visualizing the Structure, Behaviour, and Evolution of Software. Springer, Heidelberg (2007)
7. Ebden, M., Huynh, T.D., Moreau, L., Ramchurn, S., Roberts, S.: Network analysis on provenance graphs from a crowdsourcing application. In: Groth, P., Frew, J. (eds.) IPAW 2012. LNCS, vol. 7525, pp. 168–182. Springer, Heidelberg (2012)
8. Ellson, J., et al.: Graphviz and dynagraph — static and dynamic graph drawing tools. In: Jünger, M., Mutzel, P. (eds.) Graph Drawing Software, pp. 127–148. Springer, Berlin Heidelberg (2004)
9. Hoekstra, R., Groth, P.: PROV-O-Viz - understanding the role of activities in provenance. In: Ludaescher, B., Plale, B. (eds.) IPAW 2014. LNCS, vol. 8628, pp. 215–220. Springer, Heidelberg (2015)
10. Hull, D., et al.: Taverna: a tool for building and running workflows of services. Nucleic Acids Res. **34**(suppl 2), W729–W732 (2006)
11. Jacob, L., et al.: A non-intrusive approach for 2D platform game design analysis based on provenance data extracted from game streaming. In: 2014 Brazilian Symposium on Computer Games and Digital Entertainment, pp. 41–50 (2014)
12. Kohwalter, T.C., Clua, E.G., Murta, L.G.: Game Flux analysis with provenance. In: Reidsma, D., Katayose, H., Nijholt, A. (eds.) ACE 2013. LNCS, vol. 8253, pp. 320–331. Springer, Heidelberg (2013)
13. Kohwalter, T., et al.: Provenance in Games. In: Brazilian Symposium on Computer Games and Digital Entertainment, SBGAMES, pp. 162–171 (2012)
14. Kohwalter, T., et al.: Reinforcing software engineering learning through provenance. In: 2014 Brazilian Symposium on Software Engineering, SBES, pp. 131–140 (2014)
15. Kohwalter, T., et al.: SDM – an educational game for software engineering. In: Brazilian Symposium on Games and Digital Entertainment, SBGAMES, pp. 222–231 (2011)
16. Koop, D., et al.: Visual summaries for graph collections. In: IEEE Pacific Visualization Symposium (Pacific Vis 2013), pp. 57–64. IEEE (2013)
17. Moreau, L., Missier, P.: PROV-DM: The PROV Data Model. http://www.w3.org/TR/prov-dm/
18. Del Rio, N., da Silva, P.P.: Probe-It! Visualization support for provenance. In: Bebis, G., et al. (eds.) ISVC 2007, Part II. LNCS, vol. 4842, pp. 732–741. Springer, Heidelberg (2007)
19. Seltzer, M.I., Macko, P.: Provenance map orbiter: interactive exploration of large provenance graphs. In: TaPP (2011)