Hierarchical Graphs on Mobile Devices: A Lane-based Approach

Tom Horak
Interactive Media Lab
Technische Universität Dresden
Dresden, Germany
horakt@acm.org

Raimund Dachselt
Interactive Media Lab
Technische Universität Dresden
Dresden, Germany
dachselt@acm.org

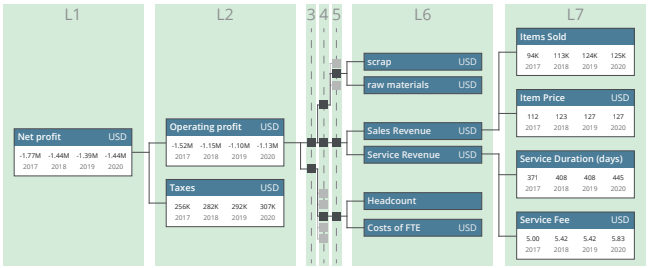

Figure 1: We propose to optimize hierarchal graphs for mobile devices with visualization and interaction concepts based on *lanes* (i.e., the different hierarchy levels; here numerated and highlighted in green), which can be expanded, minimized, or collapsed.

MobileVis '18 Workshop at CHI 2018, April 21, 2018, Montreal, QC, Canada.
https://mobilevis.github.io/

Abstract

In this paper, we propose a lane-based approach for displaying hierarchical graphs on mobile devices such as smartphones and tablets. We emphasize the hierarchical levels of the graph through lanes, in order to ease the data exploration in a mobile interface. From a visualization perspective, we adapt the appearance of nodes per lane as well as reduce non-relevant levels by collapsing the corresponding lanes. From an interaction perspective, the lanes introduce constraints that allow us to simplify the interaction vocabulary and add guidance for the user. We examine our concept by applying it to a business data analysis scenario using value driver trees (VDT). For this scenario, we further propose additional interface elements and functionalities that support the user during the data exploration as well as basic data simulations. We believe that our lane-based approach is a concept that is able to ease the visual exploration of hierarchical graphs on mobile devices.

Author Keywords

Mobile visualization; hierarchical graphs; data exploration; value driver trees; mobile devices.

ACM Classification Keywords

H.5.2 [Information interfaces and presentation]: User Interfaces.

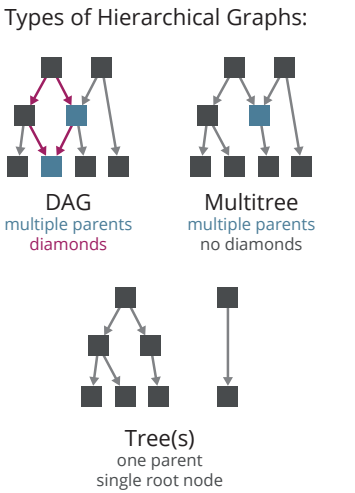

Figure 2: In directed acyclic graphs (DAG) and multitrees [6], nodes can have multiple parents, however, multitrees do not allow for diamonds (i.e., multiple paths from one node to another). In normal trees, each node has only one parent node, and only one root node exists

Introduction & Background

Hierarchical graphs are particularly interesting as they come with special layout requirements. Formally, they are directed acyclic graphs (DAG), which are laid out via a hierarchical graph drawing approach (see Figure 2). While graphs in general can often be freely altered, the importance to recognize existing hierarchy levels and how nodes in the lower levels influence nodes in the upper levels lead to certain layout restrictions. For instance, in organization charts of companies it is essential to identify in which level a certain person is, who is assigned to this person, and who are the supervisors. This fixed structure of hierarchical graphs is especially challenging when adapting them for small screens. In context of the emerging need for visual exploration tools on mobile devices, such as tablets or smartphones, this issue is getting more prominent again and requires both new visualization and new interaction concepts.

Especially in the management level of companies, the need for monitoring a companies' performance *on the go* is increasing. While providing selected single data points is straightforward, enabling more complex business data analysis on mobile devices remains challenging. An example for such a more complex analysis are value driver trees (VDT): they represent characteristics as well as parameters of a business and how these influence each other [1, 13], in most cases a company's value drivers and their impact on key performance indicators (KPIs). Value drivers are specific adjustable parameters of a business, e.g., number of sold items, item price, or number of employees, while KPIs are common business indicators, e.g., net profit, operating profit, or gross sales. Importantly, the value drivers often affect multiple other nodes, i.e., nodes can have multiple parents. Thus, VDTs can be DAGs or multitrees [6] and must not necessarily be trees (cf. Figure 2). Typically, the

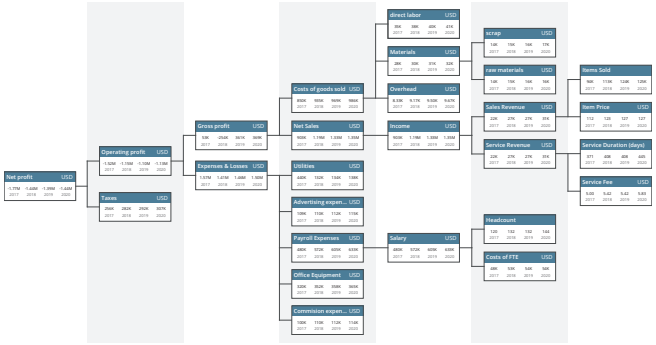

Figure 3: Value driver trees represent multiple relevant parameters of a business and how these affect each other. In many cases, a company's KPIs (at the higher levels) and value drivers (at the lower levels) are of most interest.

nodes of a VDT show the value of a given attribute for multiple years, allowing recognizing trends over time. While the size of VDTs depends heavily on the company and their analysis strategy, the example graph given in Figure 3 is representative for small and midsize companies.

Besides general research on how to design hierarchical visualizations [4, 14], there also exists work on navigating in trees and hierarchical graphs. Already 15 years ago, two similar concepts for exploring larger trees were proposed: SpaceTree [12] and Degree-of-interest Trees [3, 8]. Both try to show only as many nodes of the tree as fit into the available screen space, while the remaining subtrees are indicated through triangle representations at the parent node. In a similar fashion, McGuffin and Balakrishnan [11] proposed different layout strategies for multitrees as well as interaction mechanisms to expand multiple levels of collapsed subtrees. For more complex hierarchical

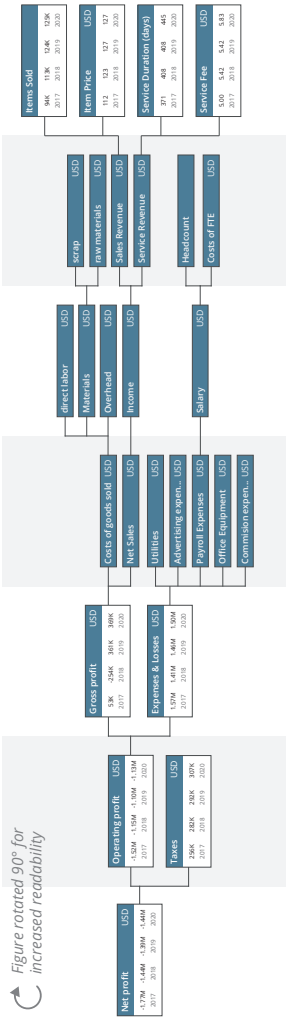

Figure 4: For lanes with many nodes, the nodes can be displayed in a minimized way.

graphs, Gladisch et al. [7] proposed a split view interface, where the hierarchical structure was used to adjust the level of abstraction of nodes. To reduce the complexity of hierarchical graphs, Lee et al. [10] developed the Treeplus interface allowing to explore such a graph sequentially in a tree-like fashion. While these concepts are able to ease the exploration, they (i) are not focused on mobile devices, and (ii) assume that users follow a top-down approach, i.e., expand a certain subtree in order to see more details. In contrast, in VDTs the leaf nodes are often representing determining factors for the data analysis, hence hiding them by collapsing whole subtrees is unfavorable. Horak et al. [9] considered these aspects in an earlier conceptual work, but focused on concepts of semantic zooming and embedded visualizations.

Lane-based Graph Layout

When working with hierarchical graphs, in most cases some specific levels (e.g., top and bottom level) or node types (e.g., leaf nodes) are of particular interest. This is similar for VDTs: the value drivers are typically at the bottom levels of the graph while the KPIs are located in the top levels [13]. As a result, not all levels are of the same interest to the user. We want to exploit this aspect by proposing a lane-based visualization and interaction approach: each lane represents a hierarchy level and can be individually controlled. More specifically, these lanes allow for two things: (i) applying different visualization adaptations to each lane, and (ii) adding constraints to user interactions in order to simplify navigation and manipulation while guiding the user. Following the typical layout of VDTs, the lanes are in a horizontal order. The nodes themselves are visualized as wide rectangles as this is common in VDTs. Although other forms could be beneficial in certain situations (e.g., tall rectangles, squares), we will not consider them in this paper.

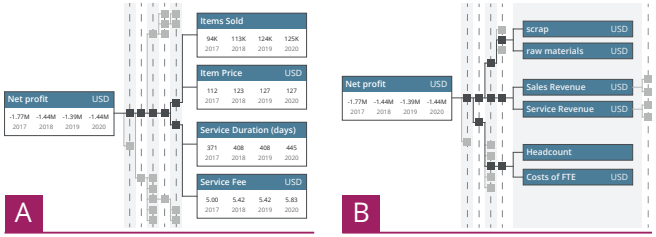

Figure 5: Lanes can be collapsed (dotted lines) in favor of a few selected lanes. Swiping horizontally allows navigating between lanes (a to b).

When bringing these visualizations to mobile devices, the limited display space is the most prominent challenge. As the tree is growing from left to right, a landscape orientation allows for the highest number of displayed lanes at the same time. However, as a lane can contain many nodes, it is possible that not all nodes fit in the available vertical space. To reduce the need for vertical scrolling inside a lane, we propose to shrink the nodes of these lanes to a minimized version where the non-essential details are removed (Figure 4). If still not all nodes can be displayed at the same time, vertical scrolling is enabled for this lane; the other lanes are not affected. In order to scroll all lanes in parallel, a vertical two-finger drag can be used. In some situations it can be required to manually adapt the level of detail, which equals a constrained semantic zooming (e.g., expanding or minimizing nodes of a lane by pinch) or a details-on-demand operation (e.g., tapping a node).

The horizontal spread is also limited, i.e., in many situations there is not enough space to show all lanes. Therefore, inspired by Elmqvist et al. [5] and Butscher et al. [2], we propose to collapse lanes that are of less interest (i.e., the lanes in the middle). As shown in Figure 1 and Fig-

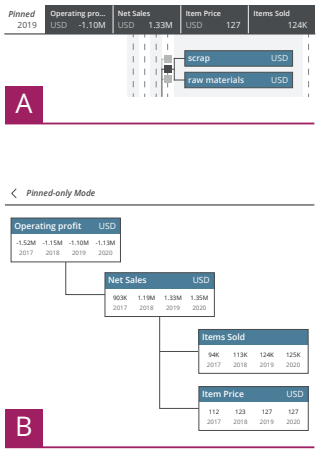

Figure 6: (a) Nodes can be pinned to a top bar; (b) a pinned-only mode provides a special view on these nodes.

ure 5, these collapsed lanes are displayed as dotted lines. All nodes and links are indicated in light gray and visually emphasized if they are part of a path between one of the nodes in the expanded lanes. To display the name of such an (indicated) node of a collapsed lane on demand, a tap could be used. For navigation, we propose that a horizontal swipe on one of the expanded lanes allows switching to a neighboring lane, i.e., swiping right collapses the current lane while expanding the next lane on the left (Figure 5a to b) and vice versa. In order to quickly access a collapsed lane (i.e., dotted line), double tapping on this lane could directly expand it in favor of the closest, currently expanded lane. Regarding tablet devices, we consider a number of four to five lanes—depending on the size and resolution—as optimal to be able to display the nodes and their content in an easily readable way. On smartphones, we propose an optimal number of only two lanes, as they are smaller and mostly held in portrait mode. However, it should be possible to manually adapt the number of expanded lanes to match different user preferences, for instance, through a horizontal pinch gesture spanning multiple lanes. As it is possible that not all lanes fit into the display space, horizontal scrolling could be supported by a two-finger drag.

As a result, the different adaptations of lanes (collapsed, minimized, expanded) represent a degree of interest from the user in these lanes, where multiple focus points (i.e., lanes) can exist. This lane-based approach is in contrast to, e.g., the degree-of-interest trees (DOITrees) [3, 8], in which the focus is on single nodes. While our adaptations are saving a significant amount of space—similar to DOITrees—they also preserve the visibility of specific lanes (i.e., hierarchy levels) at the same time. For larger graphs with a high number of lanes or nodes per lane, our approach faces clear limitations. In these cases, both horizontal and vertical scrolling can get necessary, which increases the effort for keeping track of all nodes. In these situations, additional techniques such as extended semantic zooming [9], collapsing larger graph parts [12, 3], or alternative representations could be incorporated. However, in context of our described application case, most of the VDTs are manageable in size.

Pinning of Nodes

Besides the lane-based graph layout, we propose to provide further interface elements and interaction flows for the presented application case on a mobile device. As outlined before, users are particularly interested in analyzing the effects of a changed value driver on certain KPIs or other nodes. However, these nodes may not always be located in the same lane. For instance, in our example graph, the driver nodes are not exclusively in the last lane but in the last two, while the KPI nodes are distributed across three different lanes. Therefore, we propose to allow the pinning of nodes.

By performing a long tap on a node, a copy of this node is pinned at a bar at the top of the screen. In the top bar, nodes are displayed in a simplified way showing the name, unit and value for a particular year (Figure 6a). This representation is decoupled from the graph view, thus no links are shown. On smartphones, the bar can naturally hold four to five nodes; when adding more nodes the bar should become scrollable. We propose that the pinned nodes also serve as a quick access: tapping on a node navigates to the corresponding lane, i.e., opening and expanding the lane if it is not already in focus. By again long tapping on the node, it could be unpinned.

In some situations, the user wants to focus purely on the pinned nodes. Therefore, we propose to offer a *pinned-only mode*, which is a simplified presentation of the pinned

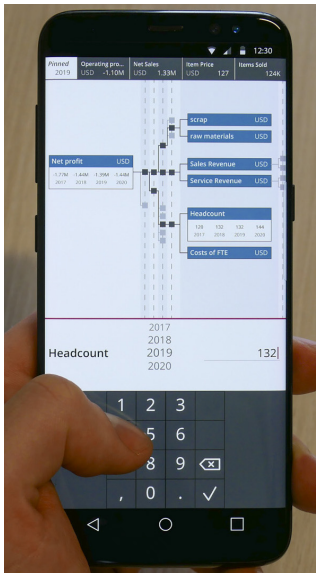

Figure 7: Mock-up of editing a value on a Samsung Galaxy S8 smartphone with opened keyboard.

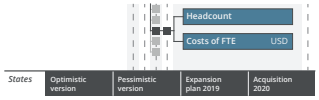

Figure 8: A bottom bar allows storing and accessing different states resulting from value manipulations.

nodes (Figure 6b). The nodes are connected with links; however, no other nodes are shown or indicated. The layout here is similar to an indented tree. The mode can be activated or deactivated by a long tap on the top bar.

Simulating different States

In order to actually manipulate a node and start a simulation, it must be possible to edit values. By double tapping on a node, an edit view with a date picker for the year and an input field for the value could be offered (Figure 7). When confirming the changed value, the graph is updated and the changes in other nodes are displayed. We propose a split view layout while editing; both the graph and the edit view should be visible at the same time. However, it must be considered that the virtual keyboard of mobile devices requires a certain amount of space, as visible in Figure 7.

After adjusting the value drivers, the user may want to store this simulation for later use. By performing a long tap on an empty area of any lane, a menu allows to store the current state with a short description. Analogous to the top bar with pinned nodes, the stored sets are displayed in a bottom bar and can be restored by tapping (Figure 8). This allows the user to create and compare different scenarios (e.g., optimistic vs. pessimistic) by quickly switching between these states. For both the top bar and the bottom bar, the bar title on the left serves as a toggle to hide the remaining part of the bar, which allows to save display space if required.

Discussion & Conclusion

In this paper, we propose a lane-based approach for visualizing hierarchical graphs and interacting with them on mobile devices (Figure 7 and 9). The lane concept allows us to introduce some constraints in favor of an improved mobile experience. For instance, instead of freely panning and zooming in parallel, we separate and limit these interac-

Figure 9: Mock-up of the envisioned mobile interface on a Samsung Galaxy S3 tablet.

tions. Users can swipe horizontally to switch between single lanes or scroll the viewport horizontally with a two-finger drag; vertical scrolling can be performed by a one-finger drag for a single lane or by a two-finger drag for all lanes in parallel (if scrollable). The zooming is limited to two states, a minimal node layout and a default layout, as well as encapsulated for each lane.

While such constraints can have negative implications, we believe that these constraints can improve the user experience especially on mobile devices. Furthermore, the lanes provide a degree of responsiveness allowing to adapt the visualization to different devices and screen estates. In combination, we believe that our concepts allow for an easy exploration of value driver trees on the go: managers are able to quickly view the most important values while still perceiving the overall interplay between them. Mainly, this is supported by emphasizing the top and bot-

tom lanes—leveraging the implicit knowledge about the graph's structure—instead of forcing a top-down navigation approach, as it is often the case in tree visualization interfaces.

Of course, our concepts heavily depend on the application case as well as the visualization and their properties. In our case, the hierarchical graphs come with a clear and strict layout; in other layouts, e.g., force-directed ones, our lane-based approach does not directly apply. However, as future work it could be interesting to look into such application scenarios and how our ideas could adapt (e.g., constraining both interaction and visual adaptations onto clusters). For now, we plan to implement our described concepts in a prototype and study our ideas in more depth.

Acknowledgments

This work was supported in part by DFG grant DA 1319/3-3 (GEMS 2.0).

REFERENCES

1. Rama Akkiraju and Ruoyi Zhou. 2012. Measuring Service Solution Quality in Services Outsourcing Projects Using Value Driver Tree Approach. In *2012 Annual SRII Global Conference*. IEEE. DOI: http://dx.doi.org/10.1109/srii.2012.30
2. Simon Butscher, Kasper Hornbæk, and Harald Reiterer. 2014. SpaceFold and PhysicLenses: Simultaneous Multifocus Navigation on Touch Surfaces. In *Proceedings of the 2014 International Working Conference on Advanced Visual Interfaces*. ACM, 209–216. DOI: http://dx.doi.org/10.1145/2598153.2598177
3. Stuart K. Card and David Nation. 2002. Degree-of-interest Trees: A Component of an Attention-reactive User Interface. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM, 231–245. DOI: http://dx.doi.org/10.1145/1556262.1556300
4. Niklas Elmqvist and Jean-Daniel Fekete. 2010. Hierarchical Aggregation for Information Visualization: Overview, Techniques, and Design Guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16, 3 (may 2010), 439–454. DOI: http://dx.doi.org/10.1109/tvcg.2009.84
5. Niklas Elmqvist, Nathalie Henry, Yann Riche, and Jean-Daniel Fekete. 2010. Mélange: Space Folding for Visual Exploration. *IEEE Transactions on Visualization and Computer Graphics* 16, 3 (may 2010), 468–483. DOI:http://dx.doi.org/10.1109/tvcg.2009.86
6. George W. Furnas and Jeff Zacks. 1994. Multitrees: enriching and reusing hierarchical structure. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. ACM, 330–336. DOI: http://dx.doi.org/10.1145/191666.191778
7. Stefan Gladisch, Heidrun Schumann, and Christian Tominski. 2013. Navigation Recommendations for Exploring Hierarchical Graphs. In *Advances in Visual Computing*. Springer Berlin Heidelberg, 36–47. DOI: http://dx.doi.org/10.1007/978-3-642-41939-3_4
8. Jeffrey Heer and Stuart K. Card. 2004. DOITrees Revisited: Scalable, Space-constrained Visualization of Hierarchical Data. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM, 421–424. DOI: http://dx.doi.org/10.1145/989863.989941

9. Tom Horak, Ulrike Kister, and Raimund Dachselt. 2017. Improving Value Driver Trees to Enhance Business Data Analysis. In *Poster Program of the 2017 IEEE Conference on Information Visualization (InfoVis)*. 2.

10. Bongshin Lee, Cynthia S. Parr, Catherine Plaisant, Benjamin B. Bederson, Vladislav D. Veksler, Wayne D. Gray, and Christopher Kotfila. 2006. TreePlus: Interactive Exploration of Networks with Enhanced Tree Layouts. *IEEE Transactions on Visualization and Computer Graphics* 12, 6 (2006), 1414–1426. DOI: http://dx.doi.org/10.1109/tvcg.2006.106

11. Michael J. McGuffin and Ravin Balakrishnan. 2005. Interactive visualization of genealogical graphs. In *IEEE Symposium on Information Visualization*. IEEE, 16–23. DOI: http://dx.doi.org/10.1109/infvis.2005.1532124

12. Catherine Plaisant, Jesse Grosjean, and Benjamin B. Bederson. 2002. SpaceTree: supporting exploration in large node link tree, design evolution and empirical evaluation. In *IEEE Symposium on Information Visualization.* IEEE, 57–64. DOI: http://dx.doi.org/10.1109/infvis.2002.1173148

13. Chunhua Tian, Rongzeng Cao, Wei Ding, Hao Zhang, and Juhnyoung Lee. 2007. Business Value Analysis of IT Services. In *IEEE International Conference on Services Computing (SCC 2007)*. IEEE. DOI: http://dx.doi.org/10.1109/scc.2007.36

14. Tatiana von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J. van Wijk, Jean-Daniel Fekete, and Dieter W. Fellner. 2011. Visual Analysis of Large Graphs: State-of-the-Art and Future Research Challenges. *Computer Graphics Forum* 30, 6 (2011), 1719–1749. DOI:http: //dx.doi.org/10.1111/j.1467-8659.2011.01898.x