Improving the Layout for Text Variant Graphs

Stefan Jänicke[1], **Marco Büchler**[2], **Gerik Scheuermann**[1]
[1] Image and Signal Processing Group, Institute for Computer Science, Leipzig University, Leipzig
[2] Goettingen Centre for Digital Humanities, University of Goettingen, Goettingen
{stjaenicke,scheuermann}@informatik.uni-leipzig.de, mbuechler@gcdh.de

Abstract

Sentence Alignment Flows are visualizations for Text Variant Graphs that show the variations between different editions of texts. Although the resultant graph layouts are a substantial improvement to standard tools that are used in the corresponding Digital Humanities research field, the visualization is often cluttered due to large amounts of edge crossings and the occlusion of edges and vertices. In this paper, we present methods for the layering of vertices, the bundling of edges and the removal of overlaps between edges and vertices to reduce clutter, and therefore, to improve the readability for such graphs. Finally, we present the results of our survey with participants from the humanities and computer science, who had the task to compare the readability of Sentence Alignment Flows to the layouts generated by our improved method.

Keywords: Sentence Alignment Flows, Text Variant Graphs, Graph Layout, Directed Acyclic Graphs

1. Introduction

One of the substantial tasks in the field of textual criticism is called collation, which is the cautious comparison of various editions of one and the same text. The traditional approach of a humanities scholar is to put the texts of several editions next to each other and mark the differences among the textual entities (e.g., sentences, sections, chapters). Since this is an extremely laborious approach many Digital Humanities projects investigate tools that support the humanities scholars with computational methods.

To solve the collation task automatically, a great variety of algorithms was developed that compute the alignment between different text editions. CollateX (Dekker and Middell, 2011) is a standard tool used in the Digital Humanities that implements such alignment algorithms as well as provides a static visualization for Text Variant Graphs. The interactive interface Stemmaweb (Andrews and Macé, 2013) extends the CollateX graph to allow for user-driven annotation and modification of the graph structure. Both tools only provide plain horizontal layouts for Text Variant Graphs without highlighting its essential features. This makes it hard to visually follow subsequent tokens.

A recently published article proposes so called Sentence Alignment Flows (Jänicke et al., 2014) as a visualization solution for Text Variant Graphs. In the spirit of Wattenberg's Word Tree (Wattenberg and Viégas, 2008), it generates a proper overview that allows to follow how the words of an individual text edition disseminate in the graph. Therefore, it utilizes font size to highlight the number of occurrences of individual tokens and it uses horizontal links and vertical aligned splines to connect subsequent tokens. Figure 1 shows a Sentence Alignment Flow example. However, as discussed in Section 2.3. the layout algorithm is designed the way that occlusions of distinct splines and overlaps of splines and vertices often occur. Especially, when working with text editions that comprise lots of variation, these artifacts hamper the readability of the resultant Text Variant Graph layouts.

The purpose of this paper is to improve the readability for Text Variant Graphs based upon Sentence Alignment Flows. In particular, we provide the following methods to reduce visual clutter within the resultant graph layout:

- **Vertex Layering by Edition**: The goal is to place the subpaths of the Text Variant Graph that are passed by the same edition closely to each other to reduce the overall height of vertical connections to be drawn.
- **Improved Edge Routing**: This method aims to simplify the visual separation between different paths by bundling links with the same source or destination and removing occlusions of similarly routed links.
- **Overlap Removal**: In contrast to Sentence Alignment Flows, we suggest removing all occlusions between edges and vertices to avoid misinterpretations when observing the graph.

To obtain an objective evaluation, we finally conducted a survey with 53 participants. Their task was to compare the readability of the graph layouts computed with the Sentence Alignment Flow and our improved layout for Text Variant Graphs.

2. Related Work

For the purpose of modelling the differences and similarities between various editions of the same text, Schmidt et. al (Schmidt and Colomb, 2009) proposed so called Text Variant Graphs, which are directed acyclic graphs that emphasize such overlapping textual structures. Lots of research has been done in developing algorithms for directed

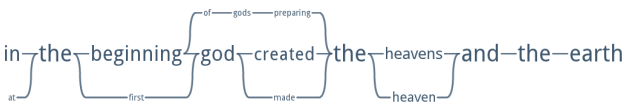


Figure 1: Sentence Alignment Flow for seven various English translations of the first Bible verse

acyclic graphs. In this section, we want to discuss traditional methods of the Graph Drawing community and the required steps in adopting some of the presented ideas for Text Variant Graphs. Furthermore, we want to consider methods developed for the dedicated research field in the Digital Humanities. Finally, we discuss Sentence Alignment Flows: a recently published layout algorithm for Text Variant Graphs.

2.1. Layout Algorithms for Directed Acyclic Graphs

Layered graph drawing as introduced by Sugiyama is the common drawing style used for directed acyclic graphs (Sugiyama et al., 1981). Typically, the vertices are placed on equally spaced horizontal (or vertical) layers and the edges are routed downwards (or rightwards) between the layers. Sugiyama's approach as well as many of its variations (Gansner et al., 1993; Cole, 2001; Utech et al., 1998; Eiglsperger et al., 2004) need to be adapted for the purpose of visualizing Text Variant Graphs, because only single vertices of one path are usually placed on one layer. This complicates the required vertical alignment of synonyms consistent of various amounts of tokens (e.g., "swarmed" and "brought forth abundantly" in *Genesis 1:21*). Additionally, the width of the vertices of a Text Variant Graph vary, so that a placing on vertical layers of equal width would further increase the distance between adjacent tokens.

To remove the occuring clutter for layered graph drawings with lots of edges, some approaches bundle edges to improve the readability of the resultant layouts (Eppstein et al., 2007; Pupyrev et al., 2011). Sentence Alignment Flows as well as its extension presented in this paper utilize this idea to compute well readable layouts for Text Variant Graphs.

2.2. Text Variant Graph Visualizations

The comparison of textual editions is a common task in textual criticism. Various research projects in the Digital Humanities focus on providing digitized editions of text to the collaborating humanities scholars.

Büchler proposed a horizontal alignment (Büchler et al., 2010) for the visualization of a Text Variant Graph for two similar text passages. One edition is used as a main branch and the variations to the second edition are highlighted in form of sub-branches in a certain color.

Several web-based tools were developed that also utilize the Text Variant Graph model to support the work with multiple digital text editions in the web browser (Dekker and Middell, 2011; Andrews and Macé, 2013). These tools compute horizontally aligned directed acyclic graphs with a plain design. The vertices are labeled with equally sized text tokens, which makes it hard to visually compare the number of occurrences of tokens. Directed edges labeled with the corresponding edition identifiers connect subsequent vertices. The layout often creates a very wide graph that contains edges routed opposed to the reading direction. These circumstances deteriorate the readability of the graph and make it hard for the observer to follow the route of a certain edition.

2.3. Sentence Alignment Flows

Within a Digital Humanities project, Sentence Alignment Flows were developed for visualizing Text Variant Graphs (Jänicke et al., 2014). In collaboration with humanities scholars experienced in the field of textual criticism, design principles were elaborated to facilitate the readability of the generated graphs, e.g., using vertex labels of different size to reflect the number of occurrences for a token, or horizontal layering of the vertices to improve the coherance of individual editions in the layout and the vertical alignment of synonyms. However, Sentence Alignment Flows produce well readable layouts for Text Variant Graphs if the extent of variation between the texts is limited. For examples with more complex variation structures, the resultant graph layouts are often cluttered due to the following incidents that are not treated by the algorithm:

Edge splines crossing text vertices: If connections are drawn between vertices of layers that are not adjacent to each other, occlusions between splines and vertices of the intermediate horizontal layers are possible (e.g. the edge between "helper" and "as" crosses the vertex "helpmate" in Figure 2). An improved vertex placement as described in Section 3.1. reduces the amount and length of vertical links, and therefore, the number of potential edge/vertex overlaps. A further strategy (Section 3.3.) is attached to remove occuring overlaps.

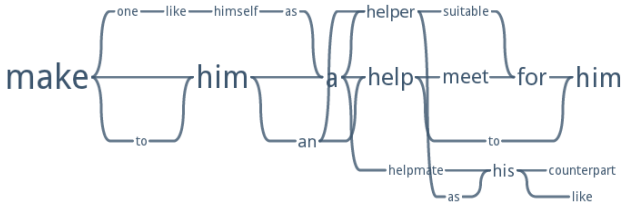


Figure 2: Edge/vertex overlaps in *Genesis 2:18*

Occlusion of similarly routed edges: An edge $\{t_1, t_2\}$ routed between text vertices of different layers consists of a spline starting from t_1, a horizontal line between the layers, and a spline ending at t_2. Thereby, it often happens that multiple horizontal lines occlude each other, so that it is hard to determine the destination of an edge. As shown in Figure 3, it is not clear if the vertex labeled "down" is linked to "as", to "like" or to both vertices. The method described in Section 3.2. solves this problem.

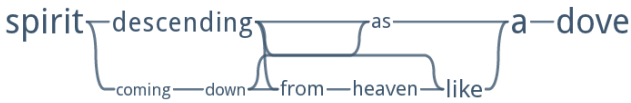


Figure 3: Edge overlaps in *John 1:32*

Occlusion of multiple splines: For strongly varying text passages lots of splines need to be drawn in the same area (see the token "adam" in Figure 4). This produces a large number of edge crossings and makes it difficult to identify the individual paths easily. The edge bundling approach proposed in Section 3.2. removes the crossings of a vertices' incoming and outgoing edges.

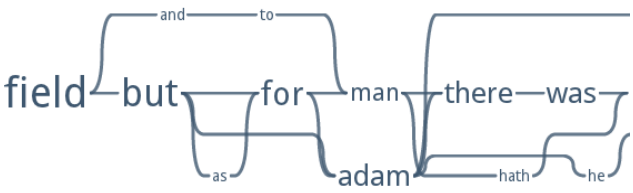


Figure 4: Edge spline occlusions in *Genesis 2:20*

3. Improved Text Variant Graph Layouts

Text Variant Graphs are constructed through various text editions transformed into paths, so that the graph basically contains multiple edges between the same vertices (like in Figure 5). Since these are routed side by side, it is sufficient to treat the readability issues mentioned in Section 2.3. for the "general case" with aggregate edges to be drawn.

3.1. Vertex Layering by Edition

When layers are assigned to vertices in the Sentence Alignment Flow algorithm, initially, the corresponding vertices for a dedicated edition e_0 are placed on the center layer 0. Iteratively, the shortest paths $\{t_1, ..., t_n\} \in G$ are determined with assigned layers for t_1 and t_n and missing layers for the vertices $\{t_2, ..., t_{n-1}\}$. The result is that the subpaths of an edition are treated at unpredictable iteration steps of the algorithm. Thus, the layers for consecutive subpaths of an edition may drift apart, so that some connections need to cross the intermediate layers. This can lead to frequent layer changes for individual editions, additional edge crossings and overlaps between vertices and edges.

We modify this process to place the subpaths of an edition as close as possible to each other by still achieving compactness of the graph. We also start with putting the vertices of e_0 on layer 0. Afterwards, we iteratively determine the next edition e_i with most vertices already assigned to layers. If multiple editions reach the same score, we choose the edition with vertices assigned to layers with lower absolute indices. For all subpaths of e_i to be inserted, we subsequently calculate the corresponding layer using the Sentence Alignment Flow method. This slightly modified approach treats the subpaths of e_i in a row, and thereby, keeps the extent of layer changes for e_i at a low level. An example can be seen in Figure 5. It uses the edition drawn in red color as e_0. The Sentence Alignment Flow algorithm (Figure 5 (a)) inserts the subpaths by increasing length in the order "so to jehova", "to yahweh your" and "to the lord your". Our proposed method (Figure 5 (b)) inserts the subpaths by edition in the order "so to jehova" (brown edition), "to the lord your" (green), and finally "to yahweh your" (orange). This reduces the total of layer changes among all editions.

3.2. Improved Edge Routing

To avoid cases such as shown in Figure 3, we insert a path layer p_i above each vertex layer l_i that reserves enough space to route horizontal links without overlaps. Furthermore, we remove overlaps of distinct vertical links. Between parallel routed edges, we require a minimum, configurable gap. The following four steps describe our improved edge routing procedure.

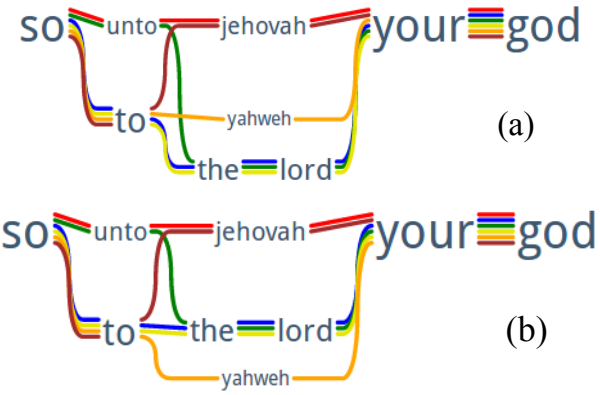


Figure 5: Vertex layering for part of *Deuteronomy 12:4*

3.2.1. Initialization of different edge types

We initialize the type for each edge $e = \{t_l, t_r\}$ dependent on the corresponding layers l_l and l_r of the connected vertices t_l and t_r. We separate three different edge types (see Figure 6):

type 0: If $l_l = l_r$ and there is no other vertex placed on l_l between t_l and t_r, e is drawn as a straight horizontal line.

type 1: If $l_l = l_r$ and there are vertices placed on l_l between t_l and t_r, a path is routed above l_l, consistent of an upward vertical link v_l, a horizontal link h on path layer p_l and a downward vertical link v_r.

type 2: If $l_l \neq l_r$, a path consistent of an upward (or downward) vertical link v_l, a horizontal link h on path layer p_l (or p_{l+1}) and an upward (or downward) vertical link v_r. We always put the horizontal link on the corresponding path layer with the higher absolute index.

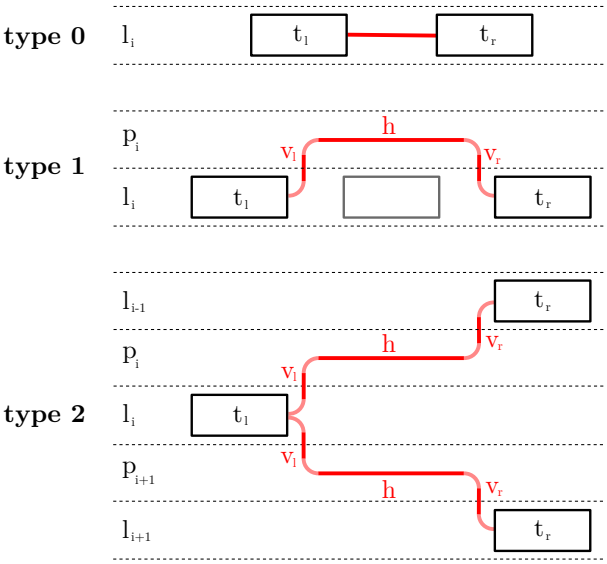


Figure 6: Different edge types

To smooth the graph layout, we connect each vertical link to its adjacent vertex and horizontal link using bends with radius r_b. Since four bends are required to draw edges of type 1 and 2, all adjacent vertices receive a minimum gap of $4 \cdot r_b$.

3.2.2. Bundling horizontal links

After all edges are initialized, we receive a list of horizontal links $h_1, \ldots, h_n$ with $h_i = \overline{t_l t_r}$ for each path layer p. We begin with constructing bundles $B = b_1, \ldots, b_n$ of horizontal links for all edges with the same left-hand vertex t_l and for all edges with the same right-hand vertex t_r. Thus, all horizontal links occur twice over all bundles. Afterwards, we sort B by decreasing number of horizontal links within the bundles. Iteratively, we insert the first bundle b_1 of B onto p. If b_1 overlaps with other already inserted bundles, we merge all these bundles into an overlap group. Then, we remove the duplicates of the horizontal links of b_1 from the remaining bundles of B and sort B again by decreasing number of horizontal links.

After placing all bundles onto p, we order and adjust the bundles of each overlap group parallel to each other. Thereby, we try to keep the number of edge crossings as minimal as possible. We perform this step iteratively by decreasing number of bundles in the overlap groups. Once a bundle that is part of multiple overlap groups is adjusted, it remains fixed for further ordering iterations. An example ordering is shown in Figure 7(a).

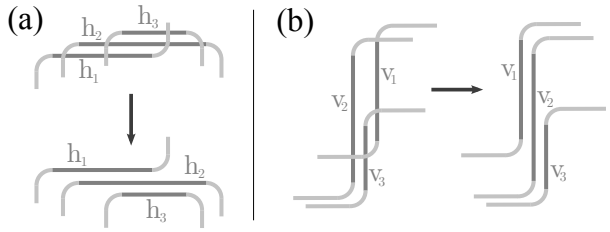


Figure 7: Ordering links of overlap groups

3.2.3. Bundling vertical links

For each vertex that is linked to neighbors with edges of type 1 and 2, we create a total of four bundles for its upward/downward incoming vertical links and its upward/downward outgoing vertical links. Since bundles of distinct vertices can be too close or even overlap each other, we perform the following two steps to keep the graph layout uncluttered. Firstly, we insert the bundles stepwise by increasing x-value into the graph layout. If a minimum gap to bundles that are already inserted is not given, we merge these bundles into overlap groups. Secondly, after all bundles are inserted, we order the vertical links of each overlap group, so that the number of edge crossings is minimal as it is shown in Figure 7(b). Finally, we test whether the required gaps between each vertical link v of the group and its subsequent glyph (right-hand vertical link v_r or vertex t_r) is still guaranteed. If this is not the case, we slightly shift all subsequent edges and vertices of v to the right so that the requirement is fulfilled.

3.2.4. Converting edges with type 2

To improve the readbility of the graph layout, we try to simplify edges of type 2 by removing one vertical link, and thereby, two of the four bends. Figure 8 illustrates an example of our approach. The upper connection $\{v_{l1}, h_1, v_{r1}\}$ between t_l and t_{r1} is replaced by $\{v_{new}, h_{new}\}$, because

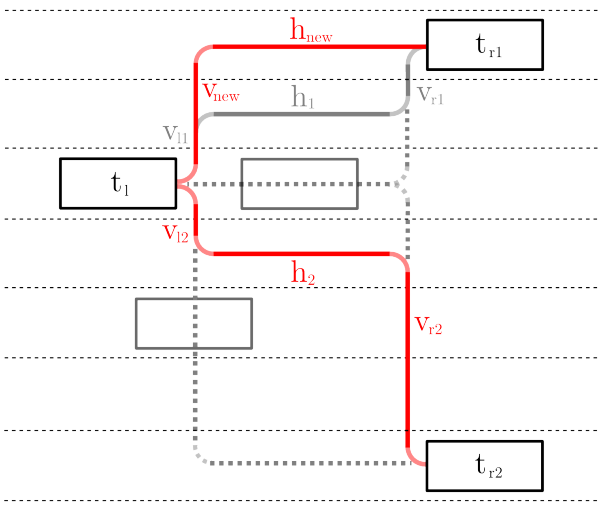


Figure 8: Converting edge type 2

neither v_{new} nor h_{new} cause an overlap with a vertex of the layout. Since this is the case for the lower connection $\{v_{l2}, h_2, v_{r2}\}$ between t_l and t_{r2}, it cannot be replaced.

There are two possibilities for each edge conversion, either the left hand vertical connection v_l gets removed and the right hand vertical connection v_r gets replaced by v_{new} or vice versa. If no overlaps are produced in both cases and $|l_l| > |l_r|$, we remove v_l and replace v_r. Otherwise, we remove v_r and replace v_l.

3.3. Removing overlaps between vertices and edges

Although the graph is designed the way the observer follows the spreading of an edition in horizontal direction, potential overlaps between a vertical link v that crosses an intermediate vertex layer l_m and a vertex t placed on l_m may hamper the readability of the graph. In such a case, we check if t can be moved horizontally without overlapping bends or other vertices by keeping the minimal required gaps to its neighbors.

An example can be seen in Figure 9(a). A leftward move-

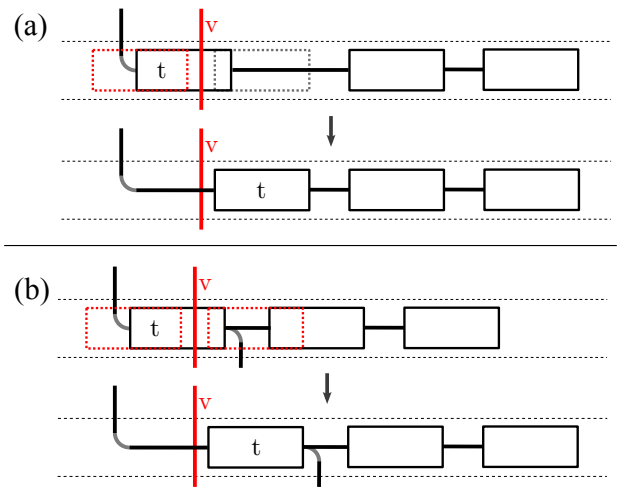


Figure 9: Overlap removal examples

ment of t is preferred, since the final position would be closer to its current position. Because this attempt fails, we move t to the right. If a horizontal movement of t is not possible (see Figure 9(b)), we move t and its subsequent edges and vertices, so that the overlap gets removed.

3.4. Improvement Validation

The benefit of the improved vertex layering method can be seen for the *Genesis 2:18* example. In comparison to the Sentence Alignment Flow layout in Figure 2, the vertical links in the modified layout (Figure 10) are shorter and rarely cross intermediate vertex layers. Furthermore, the overlap removal procedure moves the vertex labeled "helpmate" leftwards, so that the overlap with the connection between "helper" and "as" gets removed.

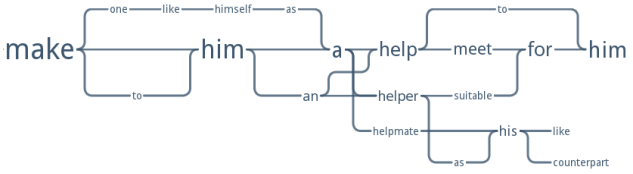


Figure 10: Removed edge/vertex overlaps in *Genesis 2:18*

Figure 11 shows the same example as Figure 3 with our new edge routing technique. The horizontal adjustment step unveils that the token "down" is connected to "as".

Figure 11: Removed edge overlaps in *John 1:32*

Finally, the advantage of bundling vertical links is visible in Figure 12. Compared to the splines drawn in Figure 4, the large number of crossings for the incoming and outgoing edges for the vertex "adam" is reduced.

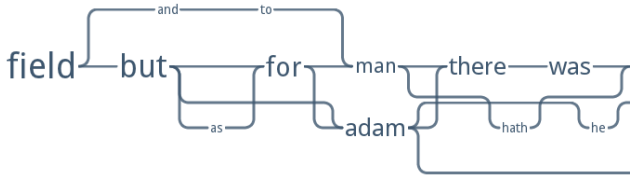


Figure 12: Removed edge spline occlusions in *Genesis 2:20*

4. Evaluation

To evaluate the benefit of the improvements for Text Variant Graph layouts described in Section 3., we conducted a survey with 53 participants. We decided against a mathematical comparison of both methods, because the resultant visualization for Text Variant Graphs needs to be understood and analyzed by humanities scholars working in textual criticism. Thus, our attitude was to let the target group decide whether our modifications are expedient or not. We further wanted to confirm our own expectations by computer scientists that are used to work with graphs. Dependent on the corresponding research fields, we could devide the participants roughly into three groups:

- **14 humanities scholars**, experienced in the field of textual criticism and partially involved in Digital Humanities projects
- **16 computer scientists**, working in Digital Humanities projects and partially dealing with issues in textual criticism
- **23 computer scientists**, most of them focused on research in visualization (partially experienced in graph visualization) and natural language processing

Within our project, we provide an interface to the humanities scholars that highlights the multiple colored edges on interaction (like in Figure 5), which further supports the understanding of the underlying structure. But for the purpose of discussing the graph readability issues, we confined the survey to the "general" case with aggregate edges.
The survey had a plain structure. The participants had to compare the readability of both layouts generated for six different examples. Each of the examples was an alignment of the same verse from seven various English translations of the Bible. The participants were asked to assess, whether they prefer Layout A (the resultant Sentence Alignment Flow), Layout B (our improved method), or if the readability of both layouts is similar. Furthermore, the reasons for the taken decisions were requested. Most of the participants had never seen any of both visualizations before the survey, so we could expect unbiased results.
We selected three of the Bible verses that contain the different structures with varying complexity that occur for typical Text Variant Graph use cases. Another three examples were randomly chosen from the remaining 28,629 verses. We wanted to assure that Text Variant Graphs benefit from our modifications generally, and not only a minor list of examples. The resultant layouts for the three selected verses are shown in Figure 14.
The results for all participants and the three different groups are juxtaposed in Figure 13. From a total of 318 comparisons, the Sentence Alignment Flow was preferred 36 times (11.3%), our improved layout 232 times (73%), and 49 times both layouts readability was assessed similarly (15.7%). When we take a look at the various groups, the

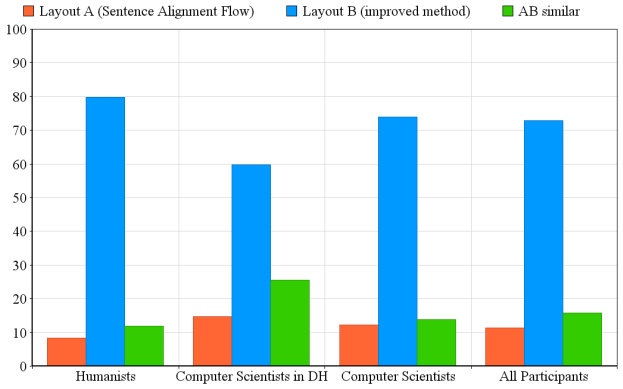
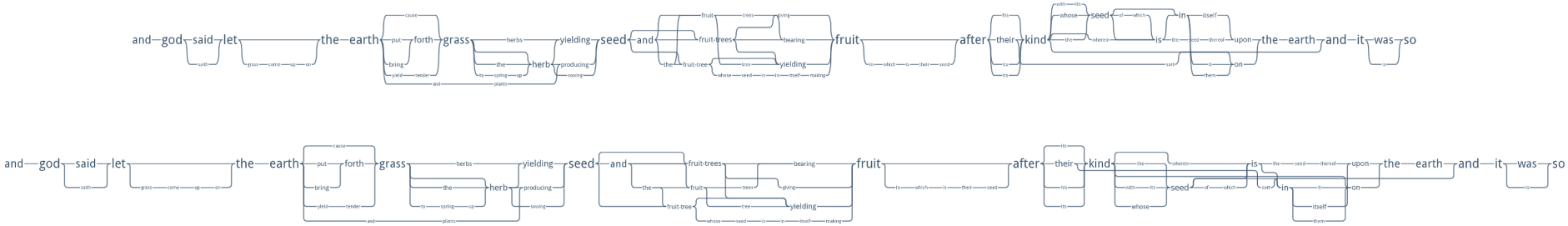
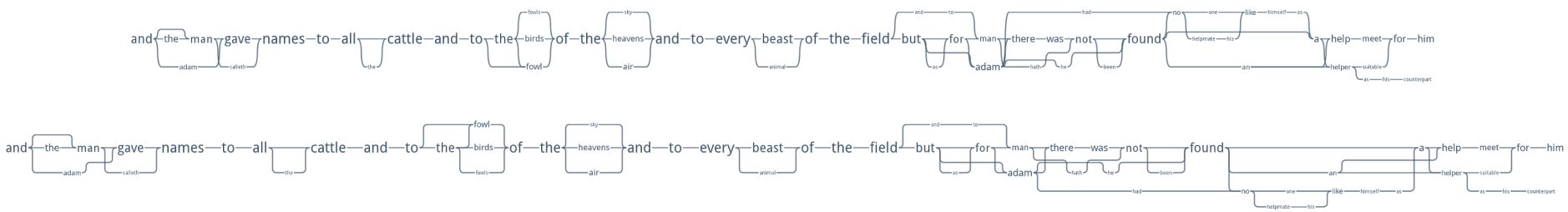


Figure 13: Preferred Layouts by participant group (in %)

Genesis 1:11



Genesis 2:20



Luke 17:1

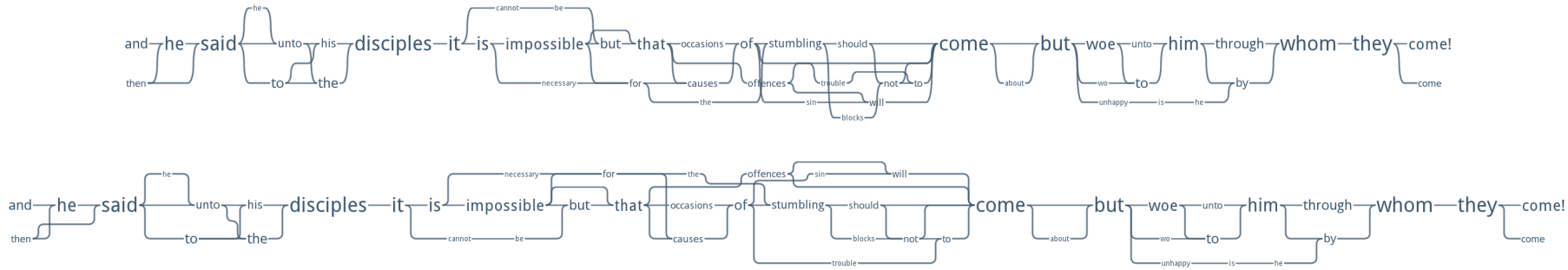


Figure 14: The three selected verses used within the survey. The upper layout is the result of the Sentence Alignment Flow, the bottom layout is the result of our method.

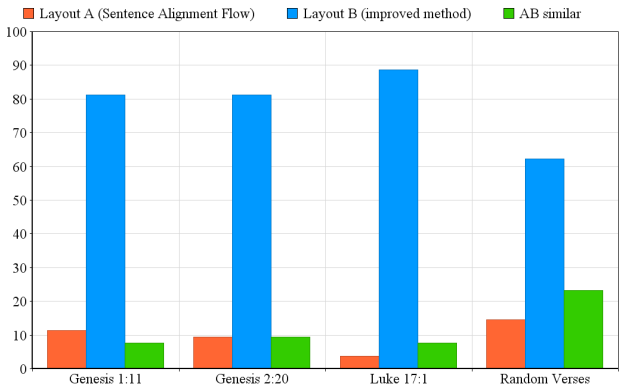


Figure 15: Preferred Layouts by verses (in %)

preferences vary only slightly. Especially satisfying was the evaluation of the dedicated target group from the humanities. In 79.8% of all cases (67 of the 84 presented layouts) the improved method of this paper was preferred over the Sentence Alignment Flow visualization.
Figure 15 shows the results for the individual examples. Our improved method turned out to be the participants' preference for the three selected verses. The reasons given for the decisions were congruent to the purpose of our modifications. Enclosed, some of the participants' comments:

- "Layout A seems to have more crossings and even lines through words which is disturbing."
- "Layout B has less clutter, no overlaps and crossings."
- "The line crossings in Layout B are orthogonal and fewer."

Particularly, one of the discussed issues of the Sentence Alignment Flow – the edge spline occlusions in *Genesis 2:20* starting from "adam" – was often perceived as problematic, e.g.:

- "In case of Layout A, the slightly vertical paths are harder to follow (e.g. at adam there is confusion of paths)."
- "At some point (e.g. after "adam"), it is easier to follow the text in example B."

For nearly one fourth (23%) of the randomly chosen verses the readability of both layouts was assessed as similar. Figure 17 shows one of the typical examples for these cases. Both layouts show less variation and clearly visible paths and the overall number of edge crossings is apparently reduced compared to the selected examples with a higher complexity. The Sentence Alignment Flow was favored in 14.5% of the cases over our new method. Often, the verses were short and properly visualized and with few or without edge crossings. An example is given in Figure 16. The participants preferring Sentence Alignment Flows argued like:

- "Compact Layout A for small sentence; easier to follow in one view."
- "Less spacing for Layout A."

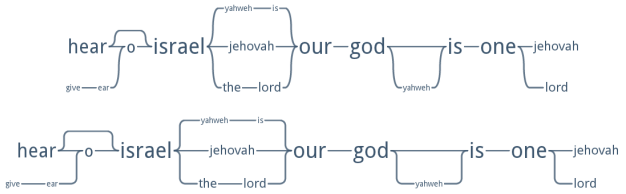


Figure 16: Sentence Alignment Flow preferred for *Deuteronomy 6:4*

- "... and Layout A is shorter."

Due to the various edge types and the required spacing between adjacent vertices, the width of the layouts computed by our method is constantly larger compared to the Sentence Alignment Flows. Although some participants favoring our method mentioned the too small spacings between subsequent vertices in Sentence Alignment Flows, the resultant compactness seems to improve the readability for some of the participants.
But our method was still mostly preferred (62%) also for the random verses. The slightly decreased percentage is attributable to the averagely lesser complexity of the random verses. Finally, some of the participants' comments justifying their decisions when favoring our improved method:

- "The lines are easier to follow – less twisty."
- "The circular edge bending looks better structured than the stretched/elliptical, which removes orthogonal feeling."
- "The word spacing and line curvature of the B layouts made them more readable to me."
- "I just like the angular style!"
- "Layout B seems to have less edge crossings and is more balanced (same amount of text above and below center line)."
- "Though the sentences in B are a lot longer/wider ... they are a lot clearer as there are no lines in the way of words while you are trying to read."

In conclusion, we determined that our method was constantly preferred when the corresponding examples contained lots of variation, and therefore, a larger amount of edge crossings. This fact confirms the benefit of the improvements proposed in this paper.

5. Conclusion

In this paper, we presented three methods to improve the readability for Text Variant Graphs compared to Sentence Alignment Flows that often contain clutter in form of edge crossings and occlusions between splines and text vertices. The first method improves the layering of the vertices that are inserted onto the layers in dependency to the corresponding editions, so that the resultant layering keeps vertices of one edition close to each other. This procedure shortens vertical links, and thereby, minimizes the amount of edge crossings and potential overlaps with other vertices.

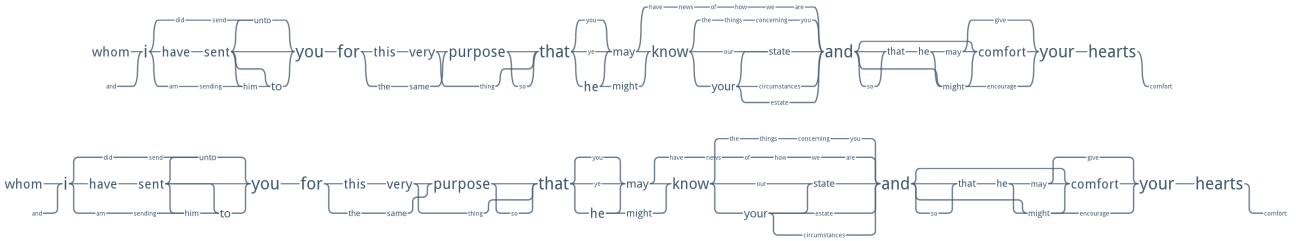


Figure 17: Similar readability of layouts for *Colossians 4:8*

As a second method, we proposed a four-step-approach to improve the routing of the edges, so that each path of the layout can be traced easily by the observer. Therefore, we separate different edge types dependent on the layers of the corresponding vertices, bundle links that share the same source or destination and route groups of overlapping links parallel to each other to avoid occlusions.

Finally, we provide a solution to remove all overlaps between vertical links and vertices. Either the corresponding vertex can be moved horizontally, or the vertex and all its successors are moved rightwards to avoid overlaps.

To evaluate our method, we conducted a survey with researchers from humanities and computer science. Their task was to compare the layouts for Text Variant Graphs produced by Sentence Alignment Flows and our method. For several examples, the participants could choose the preferred layout. Independent from the background, the majority judged the layout generated by our method as better readable, especially for examples with long texts, more complex variations and many edge crossings. For examples with minor variations and lesser edge crossings the readability of both layouts was often assessed similarly. In some cases, also the Sentence Alignment Flow was preferred.

One issue for the kind of visualization presented in this paper remains. When several editions vary the way that whole blocks of text are put in a different order, the width of the resultant graph increases rapidly, since only small parts of the texts can be aligned in form of a directed acyclic graph. To keep the graph layouts still compact and readable when highlighting such structures is one of the great challenges for the future developement of visualizations for Text Variant Graphs.

6. Acknowledgements

The authors like to thank Christian Heine for fruitful suggestions, Muhammad Faisal Cheema and Thomas Reimann for proof reading and the 53 participants of the survey for their time. This research was funded by the German Federal Ministry of Education and Research within the project eTRACES (project number: 01UA1101A).

7. References

Andrews, T. L. and Macé, C. (2013). Beyond the tree of texts: Building an empirical model of scribal variation through graph analysis of texts and stemmata. *Literary and Linguistic Computing*.

Büchler, M., Geßner, A., Eckart, T., and Heyer, G. (2010). Unsupervised Detection and Visualisation of Textual Reuse on Ancient Greek Texts. *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, 1(2).

Cole, R. (2001). Automated Layout of Concept Lattices Using Layered Diagrams and Additive Diagrams. In *Proceedings of the 24th Australasian Conference on Computer Science*, ACSC '01, pages 47–53, Washington, DC, USA. IEEE Computer Society.

Dekker, R. H. and Middell, G. (2011). Computer-Supported Collation with CollateX: Managing Textual Variance in an Environment with Varying Requirements. *Supporting Digital Humanities 2011*.

Eiglsperger, M., Siebenhaller, M., and Kaufmann, M. (2004). An Efficient Implementation of Sugiyama's Algorithm for Layered Graph Drawing. In *Proceedings of the 12th International Conference on Graph Drawing*, GD'04, pages 155–166, Berlin, Heidelberg. Springer-Verlag.

Eppstein, D., Goodrich, M. T., and Meng, J. Y. (2007). Confluent Layered Drawings. *Algorithmica*, 47(4):439–452.

Gansner, E. R., Koutsofios, E., North, S. C., and phong Vo, K. (1993). A Technique for Drawing Directed Graphs. *IEEE Transactions on Software Engineering*, 19(3):214–230.

Jänicke, S., Geßner, A., Büchler, M., and Scheuermann, G. (2014). Visualizations for Text Re-use. In *GRAPP/IVAPP*, pages 59–70.

Pupyrev, S., Nachmanson, L., and Kaufmann, M. (2011). Improving Layered Graph Layouts with Edge Bundling. In Brandes, U. and Cornelsen, S., editors, *Graph Drawing*, volume 6502 of *Lecture Notes in Computer Science*, pages 329–340. Springer Berlin Heidelberg.

Schmidt, D. and Colomb, R. (2009). A Data Structure for Representing Multi-version Texts Online. *Int. J. Hum.-Comput. Stud.*, 67(6):497–514, June.

Sugiyama, K., Tagawa, S., and Toda, M. (1981). Methods for Visual Understanding of Hierarchical System Structures. *Systems, Man and Cybernetics, IEEE Transactions on*, 11(2):109–125, Feb.

Utech, J., Branke, J., Schmeck, H., and Eades, P. (1998). An Evolutionary Algorithm for Drawing Directed Graphs. In *Proceedings of the International Conference on Imaging Science, Systems and Technology*, pages 154–160. CSREA Press.

Wattenberg, M. and Viégas, F. B. (2008). The Word Tree, an Interactive Visual Concordance. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1221–1228, November.