Edge Bundling without Reducing the Source to Target Traceability

Fabian Beck* Martin Puppe Patrick Braun Michael Burch† Stephan Diehl‡

University of Trier and VISUS, University of Stuttgart, Germany

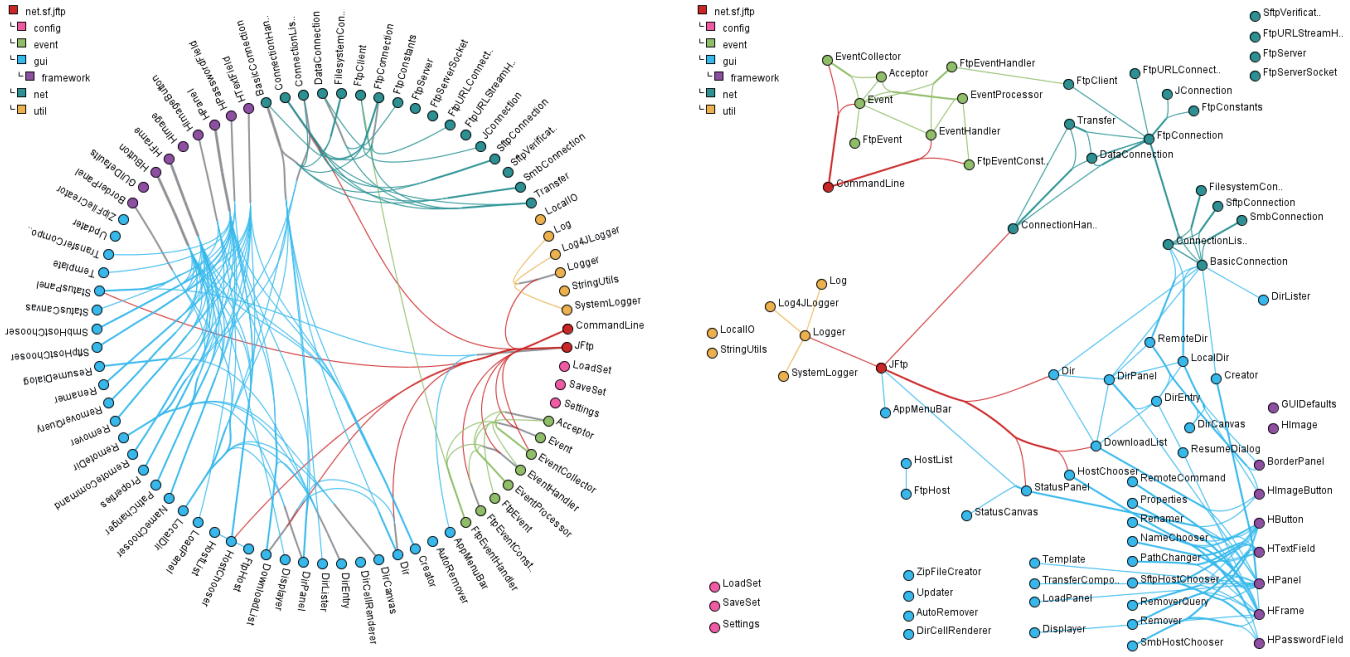

Figure 1: Two variants of the traceable edge bundling approach applied to the class dependencies of the JFtp project.

Abstract

Bundling edges improves the readability of graph visualizations by grouping similar edges together. We propose and explore an edge bundling approach that is less invasive than existing approaches and preserves the traceability of edges. Bundling is restricted to edges that start or end at the same node. The approach is applicable to directed graphs in an arbitrary node layout.

1 Introduction

Drawing edges in a graph is like travelling. A person, Alice, travelling from X to Y metaphorically represents an edge connecting the nodes X and Y. If another person, Bob, wants to travel in a similar direction—from X' near X to Y' near Y— Alice and Bob may travel parts of their way together, for instance, in the same train. This is probably more economical like it is often more economical to bundle edges in a graph to reduce visual clutter. But the problem here is, when Alice, Bob, and a bunch of other people are deboarding the train, we usually cannot retrieve where they were originally coming from—from X, X', or another location near X? Edge bundling, hence, often obscures the exact starting point of an edge but only provides rough directions. In this paper, we explore a less invasive edge bundling approach that allows retrieving the exact source and target of each edge. Metaphorically speaking, Alice and Bob only share transport if they begin their journey at the same starting point or head towards the same destination.

In general, edge bundling approaches use simplified structures to route the edges of a graph. For instance, Holten [2] uses a global hierarchy to determine how to group the edges into bundles; Cui et al. [1] propose to route the edges along a mesh structure. These approaches have in common that they bundle edges of various sources or targets. As metaphorically reported, the problem of such an approach is that the route of single edges, especially its exact source and target, usually cannot be retrieved. The bundling approach that we will present in the following is more related to drawing flow maps [3].

2 Edge Bundling Technique

The idea of our approach is to only bundle edges that have the same source or the same target. Similar to the flow map approach by Phan et al. [3], we use hierarchies to determine the bundling. We will present two edge bundling variants. While the first variant requires a radial node layout, the second variant can handle arbitrary layouts.

The graph that we use as an example is the dependency structure between the classes of a small software system (JFtp, written in Java, 78 classes). In such a system, the package structure provides a natural hierarchy on the nodes. Figure 1 shows the data set in two layouts resulting from the two variants of the approach. The package structure is visualized as the color of the nodes.

*e-mail: beckf@uni-trier.de
†e-mail: michael.burch@visus.uni-stuttgart.de
‡e-mail: diehl@uni-trier.de

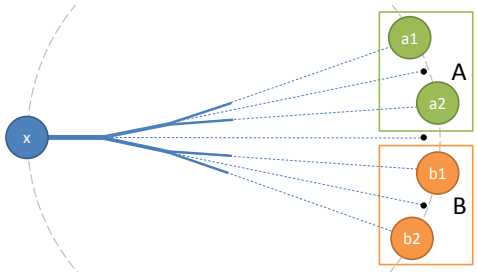

Figure 2: Descriptive example of how to create a bundle of the outgoing edges of a node x in a radial layout.

2.1 Radial Layout

A simple, but quite effective layout for graphs is a radial layout where the nodes are linearly arranged on the circumference of a circle. The first variant of our bundling approach requires such a layout and assumes that there exists a global hierarchical structure on the graph nodes. If such a hierarchy is not provided, it can be easily computed by hierarchically clustering the nodes.

The bundling algorithm first bundles the outgoing edges. For each node x, all outgoing edges are bundled like it is illustrated in Figure 2: In this example, node x is connected to four other nodes, a1, a2, b1, and b2. The hierarchical structure of the target nodes, consisting of elements A and B, is now mirrored into the bundle. In Figure 2, the bundle splits into two smaller bundles, one for hierarchy element A, one for hierarchy element B. These subbundles are then themselves split according to the contained elements. A bundle of edges always points to the barycenter of its target nodes (black dots in Figure 2). The lines are, however, not completed but stop at a certain fraction of the total distance—dashed lines are not visible in the final visualization.

The bundle of outgoing edges forms a small tree diagram for each node. Drawing these bundles is repeated for the incoming edges of each node. Finally, each node has two tree bundles, one for outgoing edges and one for incoming edges. The loose ends of these trees just need to be connected according to the graph structure to complete the diagram.

We use curved lines to make the edges easier to trace. The thickness of the line grows logarithmically with the number of the bundled edges. To underline the origin of the edges we use the color of the starting node as the color of the lines. This is, however, only possible for bundled outgoing edges, where all edges have the same origin. Thus, for the bundles of the incoming edges, which potentially summarize edges from different sources, we switch to neutral grey line colors. To avoid links overlapping nodes, we furthermore moved all split points somewhat to the center of the diagram.

2.2 Arbitrary Layouts

When switching from radial node layouts to arbitrary layouts, it is no longer justified to start with a single bundle because the target nodes may be scattered all around the diagram. In an arbitrary layout, the nodes of the same part of the hierarchy are not necessarily placed next to each other. Adapting the hierarchy so that this constraint becomes valid again, however, does not solve the problem: If a node is placed in the center of the elements of the same part of the hierarchy, targets in this part are still placed in different directions. Hence, we replaced the global hierarchy through node specific hierarchies depending on the geometric positions of the target nodes.

A resulting bundling is depicted in Figure 1 (right) for the JFtp data set. In this example, the node layout is created manually. But since the bundling approach works on arbitrary layouts, also any automatic algorithm could have been applied to compute the node positions.

The basic idea for the second bundling variant is that we define a maximum constraining the angle between two target nodes belonging to a bundle. Hence, the bundle is split at the point where this maximum value is reached. The process starts at the source node with all target nodes assigned to one bundle. At this point, the angle between two arbitrary targets in one bundle is often already above the maximum angle. Consequently, the bundle will be split directly at the source node until the constraint is valid again. In each splitting step, we split the set of targets at the largest angle of two neighboring targets (neighboring from the perspective of the source node). For each of the resulting bundles, we recursively compute the next splitting point:

1. For each pair of targets included in the bundle, compute the two points where the angle between the two targets is equal to the maximum angle. These two points are intersection points of a line from the starting point to the barycenter of the two targets and of a circle through the two targets. The radius of the circle is determined by $R = \frac{\overline{AB}}{2\sin\gamma}$, with $\overline{AB}$ being the distance between the targets A and B, and γ being the maximum angle.

2. The point among these candidates that is closest to the starting point becomes the next split point of the bundle. The bundle is split into two subbundles. Again the largest angle between two neighboring targets determines the partition of the targets.

3. Recursively continue to compute the next split point for each of the subbundles, now with the current split point as the starting point. The recursion stops when a bundle has only a single target.

Like in the radial layout, the split points are finally connected by curves. Color and strength of the lines can still be used to visualize the origin of a bundle and the number of aggregated targets.

3 Discussion

In contrast to the hierarchy-based flow map layout of Phan et al. [3], our work focuses on displaying relational information for many nodes instead of visualizing the flow relation of a single source or a small set of sources. The two algorithms that we proposed seem to be simpler to implement. New aspects of our work are the concurrent bundling of outgoing and incoming edges and the geometry-based bundling algorithm in arbitrary layouts.

Comparing the introduced bundling approach to other bundling approaches, we observe that the effect of reducing visual clutter is less. But though bundled, source and target of a particular edge can still be retrieved, which seems to be harder in other bundling approaches. These observations are, however, not empirically tested. A thorough evaluation will be a major part of our future work.

4 Conclusion

We introduced an edge bundling approach that focuses on the traceability of edges. We see its main advantage in its reduced invasiveness: Source and target are not obscured while reducing the visual complexity of the diagram.

References

[1] W. Cui, H. Zhou, H. Qu, P. C. Wong, and X. Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14:1277–1284, 2008.

[2] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[3] D. Phan, L. Xiao, R. Yeh, P. Hanrahan, and T. Winograd. Flow map layout. In *INFOVIS '05: Proceedings of the 2005 IEEE Symposium on Information Visualization*, Washington, DC, USA, 2005. IEEE Computer Society.