Interactive 3D Force-Directed Edge Bundling on Clustered Edges

Daniel Zielasko Benjamin Weyers Bernd Hentschel Torsten W. Kuhlen
Virtual Reality Group, RWTH Aachen University*
JARA – High Performance Computing

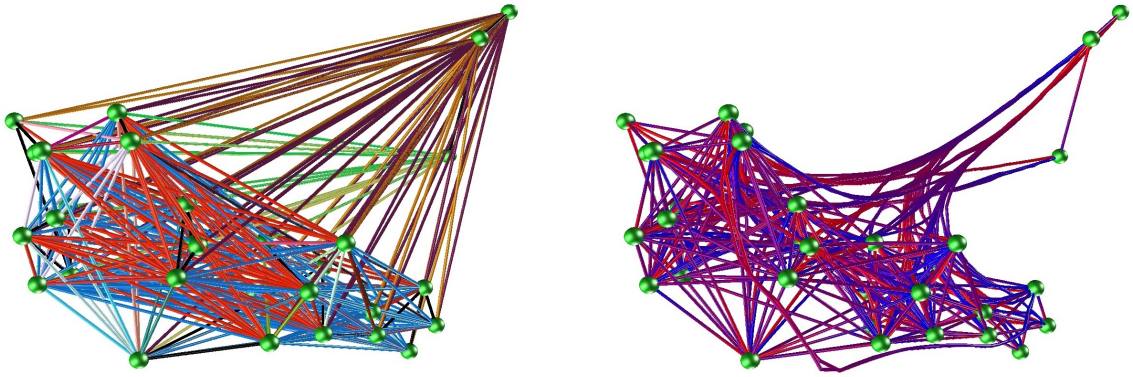

Figure 1: Node-link diagram of an almost fully connected, bidirectional graph, originating from a NEST simulation [4]. This images depicts 32 nodes each of which resembles a brain region. The edges are the regions' interconnectivity, whose weights are not considered here.
Left: original graph shown with 27 color-coded clusters consisting of similar edges; black edges are unclustered, i.e., not similar to any other.
Right: the same graph after edge bundling; the edges are directed from blue to red.

Abstract

Graphs play an important role in data analysis. Especially, graphs with a natural spatial embedding can benefit from a 3D visualization. But even more then in 2D, graphs visualized as intuitively readable 3D node-link diagrams can become very cluttered. This makes graph exploration and data analysis difficult. For this reason, we focus on the challenge of reducing edge clutter by utilizing edge bundling. In this paper we introduce a parallel, edge cluster based accelerator for the force-directed edge bundling algorithm presented in [5]. This opens up the possibility for user interaction during and after both the clustering and the bundling.

Index Terms: E.1 [Data Structures]: Graphs and networks I.3.8 [Computing Methodologies]: Computer Graphics—Applications

1 Introduction

Today, 2D graph visualization is very common and represents a broad field of past and current research. In contrast, 3D graph visualization is not that widely covered, yet. Nevertheless, especially graphs with a natural spatial embedding can benefit from a 3D visualization. One example is brain region connectivity data. But even more then in 2D, graphs visualized as intuitively readable 3D node-link diagrams can become very cluttered. This makes graph exploration and data analysis difficult. In this paper, we focus on the challenge of reducing edge clutter by utilizing edge bundling. In addition, we want to provide the user with tools for interactive manipulation, e.g., edge routing, parameter control and steering the bundling in the way she needs for her analysis. Thus, the respective algorithms have to run at interactive frame rates, i.e., the delay between a user input and the corresponding update of the visual representation should be at 100ms or below.

Holten et al. introduced force-directed edge bundling (FDEB) [5], a spring-based algorithm that is intuitive and generalizable to 3D. This algorithm has a runtime of $O(n^2)$, as for each edge, all other edges have to be visited. Although there are much faster methods available, e.g., Multilevel Agglomerative Edge Bundling (MINGEL)[1] [3] or Skeleton-Based Edge Bundling (SBEB)[2] [1], we decided to use FDEB as a basis as it produces good and comprehensible results. Moreover, it can be integrated into a broader force-based interactive system, as it is itself spring embedded. To speed up the calculation, we first cluster the edges by similarity, similar to the first step of the SBEB pipeline, and process the FDEB algorithm in parallel on these clusters.

2 Method

We implemented an initial, non-optimized version of FDEB with the addition that the visual graph representation is updated during the iterative bundling to give the user a direct feedback about the process, in the form of a smooth animation of the edges to their final position in a bundle. This gives the user the opportunity to follow corresponding edges in both representations. However, the ongoing update calculations of the visual representation further increase the runtime, which results in an unacceptable, non-interactive delay for a typical data set, e.g., a graph with 32 vertices and almost 600 edges (Figure 1).

Most of the delay is due to the calculations of the compatibility for every pair of edges and the force every single edge exerts on the current one. Notably, the compatibility and consequently the force usually are almost zero for most of the pairs, even in a graph as compact and dense as in the example.

*e-mail: {zielasko, weyers, hentschel, kuhlen}@vr.rwth-aachen.de

[1] $O(n\ log(n))$ in best case
[2] $O(C)$, with C cluster size

2.1 Edge Clustering

In order to alleviate these points, we preprocess the data by clustering edges with high compatibility. This results in subgraphs of similar edges, for which we can calculate FDEB in parallel. To measure edge similarity, we define some basic edge metrics. These metrics are inspired by the ones used to calculate the compatibility in FDEB and edge clustering in SBEB: gradient (angle), length (scale), and position. This leads to a seven-dimensional feature vector for every edge. These vectors are inserted into an R*-tree and clustered by DBSCAN [2], a density-based clustering algorithm, with a minimal cluster size of 2. Figure 1 (left) shows the result of a clustering run, where the different clusters are color-coded. The number of clusters, or the similarity of edges necessary to form a cluster is determined by the density parameter *eps*. It is a threshold for deciding if two data points, according to their Euclidean distance in parameter space, are close enough to be assigned to a common cluster. Furthermore, the subgraphs spanned by the resulting edge clusters are processed independently and in parallel by an unmodified FDEB algorithm. We use a single thread for each cluster, but share the edge position data among all of them. As every edge is assigned to at most one cluster, this allows a master thread to continuously, i.e., with every finished iteration of each FDEB thread, update the drawing of the graph during calculations.

As the clustering takes a large part of the edge similarity decisions of the FDEB algorithm, the latter's force calculations can be simplified to further reduce the runtime. This is part of our current work, although it was not factored into the first tests to get a direct comparison. In bundling the example graph, without any changes of our FDEB implementation and a number of 27 clusters, the clustered variant took 18.3 seconds and was thus 12 times faster than the basic version. Measurements were performed on an Intel Xeon E5540 2.53GHz processor running Windows 7 with 12 GB of RAM and a GeForce GTX480 graphics card. The used *eps* parameter was determined empirically by examining the visualized clusters (Figure 1, left) and the quality of the bundling (Figure 1, right). Although, smaller clusters lead to a lower runtime, over a certain number of clusters the bundling rapidly drops, which is not what we want to achieve. Finally it should be mentioned, that the animation is running smoothly at 60fps during the computation.

A positive side effect of the clustered edge bundling is that even if the runtime is not at interactive frame rates, the impression is another, as small clusters are bundled very fast (25 of 27 clusters in under a 1 second for the example) and the larger ones are just generating small changes during the remaining time. Therefore, the user is already able to explore a largely cleaned graph, while the bundling is being finished. However, a large variance in cluster sizes causes load-balancing issues in static thread-based parallelization. Additionally, while the dense structures of the orange and blue subgraphs (Figure 1, left) likely could not be cleaned up completely, there seems to be potential for a better bundling. This is addressed by interactive sub-clustering, which is described in the next section.

2.2 Interaction

We believe that user interaction is key for successful exploration and analysis of complex data spaces. Hence, a major goal of our work is to keep the used algorithms at interactive frame rates. An important part of our current work is the implementation of interactive tools, which enable steering, correcting, or manipulating of the clustering and the edge bundling.

For clustering, edge metrics like gradient, length, and position are dimensions to measure edge similarity on a structural level. Depending on the use case, several other semantical metrics are useful, such as edge weight. Therefore, the user will be able to add additional metrics, switch on/off each or even change the weight of every metric in the clustering process.

Computing the "right" density parameter for the clustering algorithm is not trivial. On the one hand, it could be very different for two graphs, on the other hand a good parameter selection strongly depends on the current analysis task. But since the clustering is fast, the user herself is put in the loop by choosing the value and instantly see a suitable visual representation of the resulting clusters (cf. Figure 1, (left) as a first prototype). This is supported by precomputing an interval of reasonable values, starting with a lower bound where most of the edges first become assigned to any cluster and an upper bound where the result is only one cluster.

In some cases, there is not one optimal *eps* value for every part of the graph, e.g., due to dense subgraphs. Therefore, the user will be able to select the appropriate cluster and initiate a further clustering within this subgraph.

An additional scenario is that the clustering is satisfying in principle, but there is a single edge the user wants to be assigned to another cluster or two clusters better be combined. Although, one solution in these cases might be to find another suitable density parameter, this will probably lead to other inadequacies. Therefore, the user will be equipped with the necessary tools to directly manipulate the clusters.

A common characteristic to all interaction tools described above is that we need to find an error-tolerant selection metaphor and a suitable visual representation of the clusters beyond the simple color coding shown in Figure 1 (left).

3 Summary and Future Work

In this paper, we have introduced a parallel, edge cluster based accelerator for the force-directed edge bundling algorithm. It opens up the possibility for a set of user interactions during and after the clustering, which in turn should heavily involve the user in the process of the overall graph layout. Thus, the user should be able to dynamically adapt the graph visualization the way she prefers for a specific analysis. We want to finish the implementation of the interactive clustering and extend the interaction to the bundling process for optimizing the parameter selection and to the bundling result by, e.g., giving the user the tools to bend edge bundles to other positions to optimize the result (cf. Edge Plucking [6]) or add global, local, or meshed sources of force to, e.g., inflate the graph or carve out structures. Furthermore, we would like to optimize the algorithm to be able to handle larger graphs and cluster sizes. Additionally, it could be interesting to examine the behavior of other clustering methods. Last but not least, there will an evaluation of the usability of the presented techniques.

Acknowledgements

The authors wish to thank the Helmholtz Portfolio – Supercomputing and Modeling for the Human Brain and the Human Brain Project for funding.

References

[1] O. Ersoy, C. Hurter, F. V. Paulovich, G. Cantareiro, and A. Telea. Skeleton-based edge bundling for graph visualization. *IEEE transactions on visualization and computer graphics*, 17(12):2364–73, Dec. 2011.

[2] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *KDD*, pages 226–231, 1996.

[3] E. R. Gansner, Y. Hu, S. North, and C. Scheidegger. Multilevel agglomerative edge bundling for visualizing large graphs. *2011 IEEE Pacific Visualization Symposium*, pages 187–194, Mar. 2011.

[4] M. Gewaltig and M. Diesmann. NEST (NEural Simulation Tool). *Scholarpedia*, 2(4):1430, 2007.

[5] D. Holten and J. J. van Wijk. Force-Directed Edge Bundling for Graph Visualization. *Computer Graphics Forum*, 28(3):983–990, June 2009.

[6] N. Wong and S. Carpendale. Supporting interactive graph exploration using edge plucking. *Visualization and Data Analysis*, 6495, 2007.