First publ. in: Proceedings of the 12th International Symposium Graph Drawing (GD ´04) (LNCS 3383), 2004, pp. 43-48

Drawing the AS Graph in 2.5 Dimensions*

Michael Baur[1], Ulrik Brandes[2], Marco Gaertler[1], and Dorothea Wagner[1]

[1] University of Karlsruhe, Department of Computer Science,
76128 Karlsruhe, Germany
{baur,gaertler,dwagner}@ilkd.uni-karlsruhe.de
[2] University of Konstanz, Department of Computer & Information Science,
78457 Konstanz, Germany
Ulrik.Brandes@uni-konstanz.de

Abstract. We propose a method for drawing AS graph data using 2.5D graph visualization. In order to bring out the pure graph structure of the AS graph we consider its core hierarchy. The k-cores are represented by 2D layouts whose interdependence for increasing k is displayed by the third dimension. For the core with maximum value a spectral layout is chosen thus emphasizing on the most important part of the AS graph. The lower cores are added iteratively by force-based methods. In contrast to alternative approaches to visualize AS graph data, our method illustrates the entire AS graph structure. Moreover, it is generic with regard to the hierarchy displayed by the third dimension.

1 Introduction

Current research activities in computer science and physics are aiming at understanding the dynamic evolution of large and complex networks like the physical internet, World Wide Web, peer-to-peer systems and the relation between autonomous systems (AS). The design of adequate visualization methods for such networks is an important step towards this aim. As these graphs are on one hand large or even huge, on the other hand evolving, customized visualizations concentrating on their intrinsic structural characteristics are required.

In this paper we propose a layout method that brings out the pure structure of an autonomous systems (AS) graph. More precisely, we focus on the core hierarchy of AS graphs. A 2D layout is obtained by first choosing a spectral layout to display the core with maximum value and then adding the lower cores iteratively by force-based methods. Using 2.5D graph visualization, we then represent the core hierarchy by stacking the induced 2D layouts of the k-cores for increasing k on top of each other in the third dimension. Visualizations in 2.5D have been proposed frequently for network data, for example to display other graph hierarchies [6, 9] or evolving graphs over time [4].

A few samples of visualizations of AS graphs are already available. However, they either focus on the geographic location of the AS [8], on the routing structure seen from a selected AS [2, 7] or on a high level view created by clustering

* The authors gratefully acknowledge financial support from DFG under grant WA 654/13-2 and BR 2158/1-2, and from the European Commission within FET Open Projects COSIN (IST-2001-33555) and DELIS (contract no. 001907).

the nodes [13]. In contrast, our method displays the entire AS graph structure without using external information. Previous attempts to analyze the structure of the AS graph propose the existence of meaningful central nodes that are highly connected to a large fraction of the graph [11]. It seems that this structural peculiarity is interpreted very well by the notion of k-cores [14, 1]. This concept is already rudimentary used for initial cleaning in [12]. Accordingly, our approach is based on the hierarchical core decomposition of the AS graph. Moreover, other kind of hierarchies can be used instead.

We consider AS graphs from different dates between 2001 and 2003 to demonstrate the usefulness of our method as means for analyzing the relation between ASes. Also graphs obtained by the Internet Topology Generator INET 3.0 [15] are consulted.

The new 2.5D visualization method for AS graphs is explained in Section 2. In Section 3 we present and discuss the results obtained for various AS graph data sets and Section 4 gives the conclusions.

2 Layout Method

Layout Paradigm. We assume a hierarchical decomposition based on the k-core concept. The *k-core* of a graph is defined as the unique subgraph obtained by recursively removing all nodes of degree less than k. A node has *coreness* ℓ, if it belongs to the ℓ-core but not to the $(\ell + 1)$-core. The *ℓ-core layer* is the collection of all nodes having coreness ℓ. The *core* of a graph is the k-core such that the $(k + 1)$-core is empty. In general, the core decomposition can result in disconnected parts. For the AS graph, all k-cores stay connected which is an advantage of the core hierarchy.

However, abstraction to the levels of hierarchy is normally accompanied by a loss of information that should be avoided. Therefore, we establish the following layout paradigm: First, all nodes and edges are displayed, second, the levels of hierarchy are emphasized, and third, the inter- and intra-level connections are made clear.

We propose an incremental algorithm to produce a 2D layout satisfying our layout paradigm. This layout is afterwards transformed into 2.5D in a canonical way using the core hierarchy. First a generic method to generate a 2D layout of a hierarchical decomposition of the graph is introduced, followed by the specification of parameters that can be chosen to fulfill certain requirements and requests induced by the structure of AS graphs.

Generic Algorithm. The first step of the algorithm constructs a spectral layout for the highest level of the hierarchy. Then, iteratively, the lower levels are added using a combination of barycentric and force-directed placement. Algorithm 1 gives a formal description of this procedure based on the core hierarchy.

Preliminary studies indicate that a spectral placement does not lead to a satisfactory layout of the AS graph as a whole. However, the results improve for increasing core value. We therefore choose a spectral layout as initial placement for the core of the graph. Then, for the iterative addition of the other level of

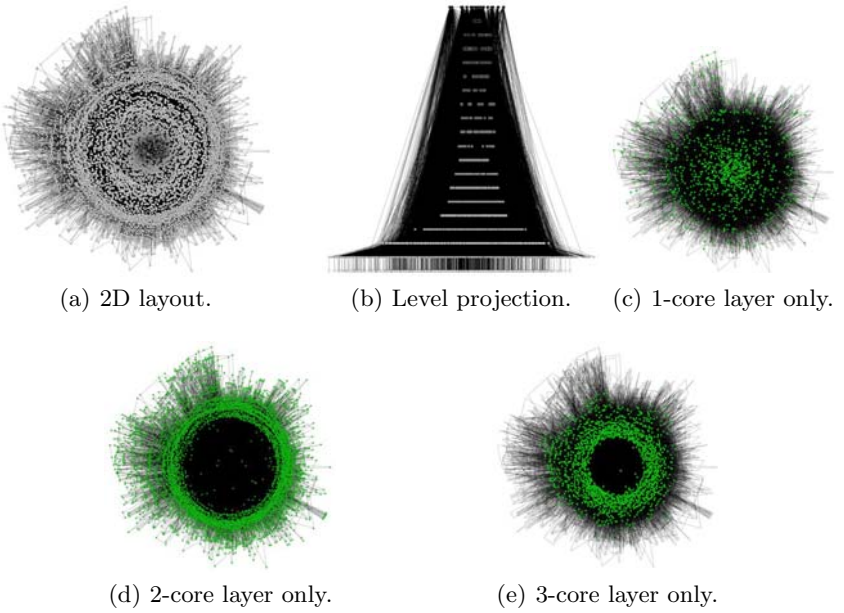
Algorithm 1: Generic AS layout algorithm.

```
Input: graph G = (V, E)
let k ← maximum coreness, G_l ← the l-core, C_l ← l-core layer
calculate spectral layout for G_k
for l ← k − 1, ..., 1 do
    if C_l ≠ ∅ then
        calculate barycentric layout for C_l in G_l, keeping G_{l+1} fixed
        calculate force-directed layout for C_l in G_l, keeping G_{l+1} fixed
        calculate force-directed layout for G_l
```

hierarchy, we first calculate a barycentric placement in which all new nodes are placed in the barycenter of their neighbors in this level. Unfortunately, barycentric layouts also have a number of drawbacks. Firstly, nodes that are structurally equivalent in the current subgraph are assigned to the same position. Secondly, all nodes are placed inside the convex hull of the already positioned nodes. In particular this means that the outermost placed nodes are those having highest coreness which is clearly contradictory to the intuition of importance. To overcome these difficulties, we use the barycentric layout as an initial placement for a subsequent force-directed refinement step, where only newly added nodes are displaced. In addition, a force-directed approach is applied for all nodes in order to relax the whole graph layout. However, the number of iterations and the maximal movement of the nodes is carefully restricted not to destroy the previously computed layout. A special feature of this relaxation step is the use of non-uniform natural spring lengths $l(u, v)$, where $l(u, v)$ scales with the smaller core value of the two incident nodes u and v. Thus, the effect of a barycentric layout is modeled, since edges between nodes of high coreness are longer than edges between nodes of low coreness. Accordingly, these springs prevent nodes with high coreness from drifting into the center of the layout.

Fitting the Parameters. Beside the choice of the hierarchical decomposition, the algorithm offers a few more degrees of freedom that allow an adjustment to a broad range of applications. Our choice of parameters are originated from the core structure of the AS graph. For the spectral layout we propose a modified Laplacian matrix $L' = 1/4 \cdot D - A$ [5]. Our experiments showed that the normalized adjacency matrix results in comparably good layouts while the standard Laplacian matrix performs significantly worse.

The force-directed placement is computed by a variant of the algorithm from [10]. Unlike the original algorithm, we calculate the displacement only for one vertex at a time and update its position immediately. Furthermore, we use the original forces but with non-uniform natural edge lengths $l(u, v)$ proportional to $\min\{\text{level}(u), \text{level}(v)\}^2$. For the local refinement step we perform at most 50 iterations and for the global roughly 20 iterations.



(a) 2D layout. (b) Level projection. (c) 1-core layer only.

(d) 2-core layer only. (e) 3-core layer only.

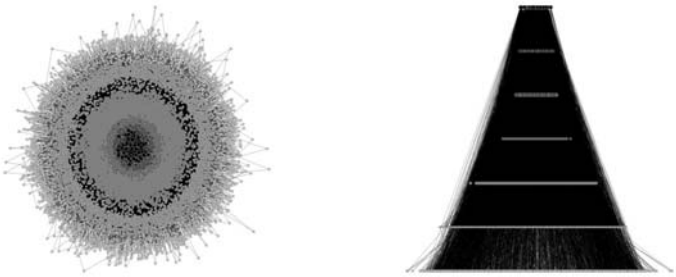
Fig. 1. 2D layout and level projection of the AS graph (06/01/02).

3 Results

We illustrate the results of our method for real AS data sets as well as for generated graphs. For a more detailed discussion, we also refer to [3]. The section is concluded by techniques to aid the human perception.

Our real world data consist of three AS graphs collected by the Oregon Routeview Project (`http://www.routeviews.org`) on different dates, i.e June, 1st 2001 (11,211 nodes, 23,689 edges, 19 levels), June, 1st 2002 (13,315 nodes, 27,703 edges, 20 levels), and June, 1st 2003 (15,415 nodes, 34,716 edges, 25 levels). In addition, we used INET 3.0 to generate artifical graphs that should exhibit a similar topology. We discuss two different two-dimensional types of figures, the 2D layout produced by Algorithm 1 and the projection of the 2.5D layout into one of the full dimensions, also referred to as *level projection*. Nodes are represented by ellipses of size decreasing according to the coreness and with colors fading from black to white. Edges are always drawn as straight lines.

Real AS Graph. The 2D layouts are dominated by the nodes with small coreness leading to a huge periphery (Fig. 1(a)). On the other hand, most nodes with higher coreness are contained in the convex hull of the core, which is apparent in Figure 1(b) and documents the relation between importance and coreness. A closer examination reveals three almost separated radial areas around the center. The first one mainly contains the 3-core layer, while the 2-core layer forms the second and third area that are distinguished by their density (see Fig. 1(c)–1(e)).



(a) 2D layout. (b) Level projection.

Fig. 2. Layouts of the generated graph with 11,211 nodes.

This reflects the heterogenous importance distribution within these areas. In contrast, a large part of the 1-core layer is attracted to the central region. These properties can be observed for all three instances. The well-known growth of the AS graph affects especially the 2- and 3-core layers. We observe that the spatial distances of these two layers decreases over time.

Generated Graphs. There are significant differences of the generated graphs to the real AS graphs, e.g. in the number of edges (35,300 vs. 23,700) and core levels (8 vs. 19). An obvious difference of the generated graph is the more uniform distribution of cardinalities of the core layers (Fig 2). Accordingly, the separation of the different core layers is less visible in the layout.

Supporting Perception. There are several means for visual aid in 2.5D layouts, i.e. choice of perspective (in 3D), additional geometric objects emphasizing the levels of hierarchy, and colors. The choice of perspective is very powerful. We have already used this feature when presenting only the 2D layout and the level projection respectively. More general, a user can focus on individual aspects, i.e. a global oriented view, a hierarchical version, or a mixture of both. A beneficial consequence might be that unintended information is automatically masked out by the perspective. In order to simplify navigation in the three dimensional space, one can also introduce additional objects that mark the levels of hierarchy, i.e. rectangles, discs, or planes. Transparency or filters might even increase their effectiveness. Color can be used in various ways, to highlight nodes and edges of special interest, to code the levels of hierarchy, or to improve the overall perception. We used transparent rectangles that absorbed light to draw layers and colored the nodes accordingly to their coreness. The color of the edges are determined by a linear interpolation of their incident nodes' color (see Fig. 3).

4 Conclusion

Core based 2.5D visualizations of the AS graph support the recognition of its detailed hierarchy. Especially, it emphasizes the characteristics of the lower core

(a) Small example. (b) AS graph (06/01/03).

Fig. 3. Visual support features.

layers and their connections with the highest layers. The evolution of the AS graph has an observable effect on the layout. Also there is a significant difference in the layouts of real AS graphs and generated ones.

References

1. V. Batagelj and M. Zaveršnik. Generalized cores. Preprint 799, University of Ljubljana, 2002.
2. G. Di Battista, F. Mariani, M. Patrignani, and M. Pizzonia. BGPlay: A System for Visualizing the Interdomain Routing Evolution. In *Proc. of Graph Drawing, GD'03*, volume 2912 of *Springer LNCS*, pages 295–306. Springer, 2004.
3. M. Baur, U. Brandes, M. Gaertler, and D. Wagner. Drawing the AS Graph in Two and a Half Dimensions. TR 2004-12, Informatics, University Karlsruhe, 2004.
4. U. Brandes and S. Corman. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization*, 2(1):40–50, 2003.
5. U. Brandes and S. Cornelsen. Visual ranking of link structures. *Journal of Graph Algorithms and Applications*, 7(2):181–201, 2003.
6. U. Brandes, T. Dwyer, and F. Schreiber. Visual understanding of metabolic pathways across organisms using layout in two and a half dimensions. *Journal of Integrative Bioinformatics*, 0002, 2004.
7. CAIDA. Walrus – graph visualization tool, 2002.
8. CAIDA. Visualizing internet topology at a macroscopic scale, 2003.
9. P. Eades and Q. Feng. Multilevel visualization of clustered graphs. In *Proc. of Graph Drawing*, volume 1190 of *Springer LNCS*, pages 113–128. Springer, 1996.
10. T. Fruchtermann and E. Reingold. Graph drawing by force-directed placement. *Software – Practice and Experience*, 21(11):1129–1164, 1991.
11. M. Gaertler and M. Patrignani. Dynamic analysis of the autonomous system graph. In *IPS 2004 – Inter-Domain Performance and Simulation*, 2004.
12. C. Gkantsidis, M. Mihail, and E. Zegura. Spectral analysis of internet topologies. In *IEEE Infocom 2003*, 2003.
13. G. Sagie and A. Wool. A clustering approach for exploring the internet structure. In *Proc. 23rd IEEE Conv. of Electrical and Electronics Engineers in Israel*, 2004.
14. S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.
15. J. Winick and S. Jamin. Inet-3.0: Internet topology generator. Technical Report UM-CSE-TR-456-02, EECS, University of Michigan, 2002.