Fast Graph Drawing Algorithm Revealing Networks Cores

Romain Giot, Romain Bourqui

Fast Graph Drawing Algorithm Revealing Networks Cores

Romain Giot, Romain Bourqui
Labri, Univ. Bordeaux, France
romain.giot@labri.fr, romain.bourqui@labri.fr

Abstract

Graph is a powerful tool to model relationships between elements and has been widely used in different research areas. Size and complexity of newly acquired graphs prohibit manual representations and urge a need for automatic visualization methods. We are interested with the node-links diagram which represents each node as a glyph and edge as a line between the corresponding nodes. We present a novel layout algorithm that emphasizes the cores of very large networks (up to several hundred thousand of nodes and million of edges) in few seconds or minutes. Our method uses a hierarchical coreness decomposition of the graph and a combination of existing layout algorithms according to the clusters topologies. Area-aware drawing algorithms which produce node overlap-free drawings are used to reduce the visual clutter. Edges are bundled along the hierarchy of clusters to highlight the network communities and reduce edge visual clutter. We validated our approach by comparing our method against one of the fastest method of the state of the art on a benchmark of 23 large graphs extracted from various sources. We have statistically proved that our method performs faster while providing meaningful results.

Keywords— **Graph drawing, Hierarchical clustering, Network community visualization**

1 Introduction

Graphs play an important role in many research areas, such as biology, microelectronics, social sciences, data mining and computer science. Improvements in data acquisition techniques urge a need for automatic visualization methods, as the size and the complexity of such acquired graphs prohibit manual drawing. Therefore, the graph drawing and the information visualization communities focus on designing effective visualizations of such large graphs. Among the different visualization methods, we are interested in the node-links diagram which represents each node as a glyph (usually a circle or rectangle) and each edge as a line (straight or curved) between the corresponding nodes. For particular classes of graphs, such as trees, planar graphs or directed acyclic graphs, effective solutions have been found that give very good results in terms of time/space complexity and in terms of aesthetic criteria. However, real-world graphs from application domains usually do not belong to these classes. To find an algorithm that gives good results (in term of computation time, aesthetic criteria and information emphasized) for arbitrary large graphs is a very difficult problem.

We present a new layout algorithm, called *HCBL* (for *Hierarchical Coreness Based Layout*), that emphasizes the cores of very large networks (up to several hundred thousand of nodes and million of edges) in few seconds or minutes. It is based on a hierarchical coreness decomposition of the graph and a combination of existing layout algorithms according to the clusters topologies. It bundles edges along the hierarchy of clusters to highlight the network communities and to reduce edge visual clutter. We validated our approach by comparing *HCBL* against one of the fastest method [19] of the state of the art on a benchmark of 23 large graphs extracted from various sources, and have statistically proved that *HCBL* performs faster while providing meaningful results.

The strengths of the algorithm are: (i) it can lay out very large graphs in few seconds or minutes and outperforms one of the fastest force directed algorithm [19]; (ii) it emphasizes the cores of the network, and therefore eases the visual community detection task; (iii) it provides a layout minimizing visual clutter as node-node overlap cannot occur and as edge bundling is used to reduce edge clutter; (iv) it is easy to implement and could be parallelized in order to be even faster.

The paper is structured as follows. Section 2 provides the notations used within the paper. Section 3 reviews related work on large graph layouts. Section 4 details the different steps of our graph drawing algorithm and section 5 describes the benchmark protocol we have set up to evaluate *HCBL*. Section 6 presents the result and provides discussion. Finally, section 7 draws a conclusion and gives directions for future work.

2 Notations and Vocabulary

This section presents the notations and definitions used in the paper. Let $G = (V,E)$ be a simple graph, with V the set of vertices, $E \subseteq V \times V$ the set of edges. A free tree $T = (V,E)$ is a connected graph without cycles, while a

rooted tree is a tree with one vertex which is the root and directed edges from the root to the leaves. A clustered graph $CG = (G,T)$ corresponds to a graph G and a rooted tree T whose leaves are the nodes of G. Each node v of T represents a cluster of vertices of G that are the leaves of the subtree of T rooted at v. The height of the clustered graph is the height of the tree. A Quotient graph [10] $Q_G = (V_Q, E_Q)$ of a partition $(C_1,\ldots,C_k)$ of the nodes of $G = (V,E)$ is defined as follows: $V_Q = \{C_1,\ldots,C_k\}$ and $(C_i,C_j) \in E_Q$ if and only if $i \neq j$ and $\exists u \in C_i$ and $v \in C_j$ such that $(u,v) \in E$. For a clustered graph $CG = (G,T)$, to each level l of T corresponds a quotient graph $Q^l_G = (V^l_G, E^l_G)$ where V^l_G is the partition represented by the nodes of level l of T. We call *metanode* a node of the quotient graph that represents a cluster C_i in the original graph, and a *metaedge* links two metanodes. A subgraph H of G induced by a set $C \subseteq V$ is the *k-core* of G if and only if $\forall v \in C$, $degree_H(v) \geq k$ and H is the maximum subgraph with this property [30]. Obviously the *(k+1)-core* of a graph G is a subgraph of the *k-core* of that graph. The *coreness* of a node v of G is equal to c if it belongs to the *c-core* but not the *(c+1)-core*, that is the maximal core a node belongs to.

3 Previous Work

Large graph drawing has been widely studied during the last decades and different approaches have been proposed. These approaches could be classified into two categories: the multilevel one and the linear algebra one.

The force directed algorithms are the most popular in the multilevel category. They overcome the time complexity/computation time issue of classical force directed algorithm (e.g. [15, 16]) by computing a multilevel clustering of the graph and make a trade-off between computation time and aesthetics of the resulting layout. FM^3 [19] is one of the fastest methods of the literature while providing meaningful layouts in $O(|V|log|V| + |E|)$. The graph is recursively reduced by using rules which use an analogy with galaxies, until reaching a fixed amount of nodes. Then, the algorithm uses a force directed model with multipoles expansion to draw each level of the hierarchical clustering. Gajer and Kobourov [17] used a different approach with graph distances between nodes (called *Maximal Independent Set Filtration*) to iteratively elect representatives until only a very few nodes remain. Classical force directed algorithms are then used to lay out each level of representatives. Although the force directed algorithms can handle very large graphs, the resulting layouts suffer from the so-called *hairball* effect [18] (i.e. a large amount of clutter due to the number of edge crossings). For very complex networks, other methods based on detection of communities and/or particular topological structures may be more suited. In both [31] and [4], the authors used the biconnected components decomposition to extract the underlying tree structure of the network. In TopoLayout [6], Archambault *et al.* generalize that method and recursively extract topological features from the graph and then use drawing algorithms dedicated to them. In [22] and [13], the authors proposed similar methods based on the computation of a hierarchical decomposition and a hybrid space filling–force directed algorithm to draw in a bottom-up manner the entire network. The coreness [30] has also been the key point of some of these community detection methods as it provides interesting results while offering a good time complexity in $O(|E|)$. Baur *et al.* proposed in [9] to display the graph in 2.5D where the bottom layer contains the 1-core of the graph and the highest layer contains the highest core of the network. In that method, the subgraph with the highest core value is first drawn with a spectral layout algorithm and the lower cores are progressively inserted in the drawing using a force directed algorithm. Alvarez-Hamelin *et al.* [3] also use the coreness value as an information in order to draw a graph. The nodes are then lay out on nearly concentric circles using polar coordinates where the radius of the coordinate of a node depends on its coreness and its neighborhood.

Harel and Koren [20] provide an algorithm, named *HDE*, that first extracts m pivots from the graph corresponding to m different points of view. Then a distance matrix is built by computing the distances between all the nodes of the graph and these m pivots. The algorithm finally performs a dimensionality reduction using a classical Principal Component Analysis to obtain a 2 or a 3 dimensional representation of the network. In [23], the authors use a different approach, called *ACE*, based on the eigenvectors of the Laplacian matrix of the graph. To speed up the computation, the authors introduce the algebraic multigrid method which computes an approximation of the eigenvectors using a hierarchy of matrices. That method computes the eigenvectors of the highest level matrix and estimates the eigenvectors of lower and lower levels. To embed the graph in d dimensions, the authors use the d most representative eigenvectors. Such algorithms offer very good computation times but aesthetics of the results are satisfactory only on a "specific subset of general graph, many of which are grids" according to [6].

4 A Multilevel and Topology-Aware Drawing Algorithm

We propose a bottom-up multilevel layout algorithm. First, we compute a hierarchical clustering of the graph with the quotient graph of each level by using the coreness measure. This iterative process ends when the quotient graph contains only one cluster or is a tree. Then, the quotient graphs of the hierarchy are laid out in a bottom-up manner. It ensures that no node-node overlap can occur as the size of a metanode is set to the size of the bounding box of the underlying cluster layout. Finally, final absolute positions

of the nodes of the original graph are recursively computed by traversing the hierarchy tree in a top-down manner. We also use a variant of the technique of Holten [21] which bundles the original edges along the hierarchy tree (corresponding to the hierarchical clustering) to ease the visual identification of the network cores. Detailed explanations follow:

4.1 Hierarchical Clustering with the Coreness

HCBL computes the *coreness* [8] of each node of the graph and uses the connected components of each equivalent class as a partition of the nodes. That step produces a flat partition of the nodes where each set of the partition is connected and only contains nodes of equal *coreness*. HCBL then computes the quotient graph associated to that partition. This allows to obtain a first level of abstraction of the network. These steps are repeated on the resulting quotient graph until the partitioning step produces a single cluster to obtain a hierarchical partition of the network. Even if the number of iterations and thereby the number of levels of the hierarchy have not been theoretically bounded, our experiments (see section 6) show that it barely exceeds 10.

4.2 Area and Topology-Aware Layout of Clusters

HCBL lays out the quotient graphs of the hierarchy in a bottom-up manner with area-aware drawing algorithms which ensures that no node-node overlap happens at any level of the hierarchy. Inspired from the TopoLayout method [6], *HCBL* uses dedicated drawing algorithms depending on the clusters topology. It considers three cases for a cluster $C = (V_C, E_C)$: (i) if C is a *quasi-clique* with density greater than 0.8[1], it uses a circular drawing algorithm; (ii) if C is a free tree, it uses the area-aware version of Archambault *et al.* [5] of *Rings* [32] which is particularly suited for emphasizing isomorphic sub-trees as well as symmetries; (iii) otherwise, it uses a combination of FM^3 algorithm [19] followed by the *Fast Overlap Removal (FOR)* algorithm [14] (which has been, modified to consider circular nodes instead of squared ones).These three general cases are generic enough to be present in most graphs.

4.3 Metanode Expansion

Once the position of nodes and metanodes have been computed relatively to their parent metanode in the hierarchy, it is necessary to compute their final absolute coordinates. This is achieved by substituting in a top-down manner each metanode by its underlying cluster and by centering this cluster to its metanode coordinate. As (i) we have set the size of each metanode to the size of the bounding box of the underlying cluster, and (ii) we have used overlap free layout methods, such substitution cannot not create node-node overlap. Note that we consider each (meta)node is circular.

[1] *i.e.* $|E_C| \geq 0.8 \frac{|V_C|*(|V_C|-1)}{2}$

4.4 Edge Bundling

The final step of *HCBL* consists in reducing edge-edge clutter due to edges crossings. To do so, the edge bundling technique [21] is applied to all inter-clusters edges[2] which eases the visual detection of clusters in the overall drawing of the graph. By laying out the metanodes of the hierarchy, we have also built a nested representation of the hierarchy tree representing the hierarchical clustering. In that tree, leaves are the original nodes of the graph while internal nodes are the metanodes of the different quotient graphs. The original edge bundling method uses clusters center as bends to route the edges. We use a different novel approach where the bends are set on the border of the clusters. For each edge e, we first determine in the quotient graphs of each level of the hierarchy, the meta-edge M_e representing e. We then compute the two *middle* bends of e as the intersection points of M_e and its extremities. The remaining bends of e are computed by iteratively adding bends where the line between the previous bend and the center of the deeper cluster intersect.

5 Experimental Protocol

5.1 Materials

To perform the evaluation of *HCBL*, we used a benchmark of 23 real world graphs. The number of nodes varies from 4039 to 8797692 with a median of 60388.0 while the number of edges varies from 28980 to 68993773 with a median of 408102. Several graphs were downloaded from the SNAP database[3] [24–29]: *ca-AstroPh*, *ca-CondMat*, *ca-GrQc*, *ca-HepPh*, *ca-HepTh*, *cit-HepPh*, *cit-HepTh*, *cit-Patents*, *email-Enron*, *email-EuAll*, *facebook_combined*, *gplus_combined*, *soc-Epinions1*, *soc-LiveJournal1*, *soc-pokec-profiles*, *soc-Slashdot0811*, *twitter_combined*, *wiki-Vote*, *wiki-Talk* ; *pgraph* is a protein homology graph presented in [1]; *Cheswick-2005* is the internet tomography dataset generated in 2005 by Cheswick's Internet Mapping Project [11]; *hero-social-network* and *comic-hero-network* which represent relations between Marvel characters [2]. Table 1 summarizes them.

5.2 Baseline Algorithms and Analysis methodology

In that evaluation, we propose to compare *HCBL* to (i) FM^3 , (ii) FM^3 followed by our modified *FOR* (FM^3_{FOR}), and (iii) a variant of *HCBL* which only uses FM^3_{FOR} drawing algorithm (*HCBL* FM^3_{FOR}).

5.3 Technical Aspect

HCBL is implemented in C++11 with Tulip visualization framework 4.6 [7]. *FOR* and FM^3 algorithms were already available within the Tulip framework. The FM^3 implementation integrated in the Tulip framework is provided by Open Graph Drawing Framework [12] and implies

[2] *i.e.* edges having their extremities in different clusters

[3] `http://snap.stanford.edu/data/`

Table 1: Summary of the dataset and performances of each algorithm. $\emptyset$ is used to represent the cases impossible to compute.

Graph	Graph		Tree	Total times							Individual cumulative times			
	#nodes	#edges	height	HCBL (s)	FM^3(s)	Gain(%)	FM^3_{FOR}(s)	Gain(%)	HCBL FM^3_{FOR}(s)	Gain(%)	Lay.	Clust.	Exp.	Bundl.
cheswick-2005	190384	228354	3	**16.90**	122.28	623	422.01	2397	95.36	464.26	14.21	1.46	0.10	0.41
soc-Epinions1	112298	508837	8	**9.93**	73.65	642	225.05	2167	82.61	732.34	3.07	4.16	0.17	1.38
soc-Slashdot0811	154676	905468	8	**14.99**	108.02	620	324.19	2062	95.02	533.89	7.22	4.21	0.14	2.06
email-Enron	73384	367662	6	**6.20**	34.87	462	112.47	1714	44.67	620.72	3.11	1.68	0.07	0.76
email-EuAll	300069	420045	6	**25.65**	119.38	365	396.22	1444	1836.00	7057.89	12.04	8.50	0.49	2.12
ca-CondMat	46266	186936	5	**6.89**	22.80	230	79.92	1059	15.01	117.69	5.78	0.56	0.02	0.27
twitter_combined	81306	2420766	7	**14.90**	56.03	276	156.44	949	31.10	108.72	4.08	4.83	0.09	4.44
cit-HepTh	48239	352807	6	**9.12**	27.24	198	87.50	859	16.52	81.14	7.12	1.02	0.03	0.53
cit-HepPh	60388	421578	6	**12.12**	37.15	206	110.22	809	22.33	84.24	9.55	1.33	0.04	0.66
ca-HepTh	19754	51971	4	**2.58**	8.48	228	23.15	795	5.82	125.27	2.26	0.17	0.01	0.08
pgraph	30727	1206654	5	**5.80**	17.61	203	49.60	755	12.84	121.42	3.09	1.10	0.01	0.70
ca-AstroPh	37544	396160	6	**7.36**	21.30	189	61.33	732	12.43	68.82	5.63	0.82	0.02	0.55
hero-social-network	10469	178115	6	**1.14**	4.68	309	8.80	669	3.29	187.84	0.43	0.29	0.01	0.32
comic-hero-network	19286	96519	5	**2.64**	7.96	201	19.68	645	5.58	111.37	2.05	0.29	0.01	0.18
ca-HepPh	24016	237010	6	**4.50**	12.08	168	31.14	592	7.25	61.16	3.69	0.39	0.02	0.21
wiki-Vote	8491	103689	6	**1.17**	4.66	300	7.43	537	2.99	156.31	0.74	0.20	0.01	0.16
ca-GrQc	10484	28980	4	**1.20**	3.62	202	5.94	395	2.37	97.91	1.05	0.08	0.00	0.05
gplus_combined	107614	30494866	9	**152.00**	369.01	142	514.54	238	209.10	37.57	7.53	64.92	0.18	52.28
facebook_combined	4039	88234	4	**0.84**	1.56	84	1.77	109	1.30	54.12	0.64	0.08	0.00	0.09
Problematic graph for baseline only														
soc-LiveJournal1	8797692	68993773	8	6792.00	$\emptyset$	∞	$\emptyset$	∞	**1070.00**	−84.25	6069.20	404.97	4.05	313.78
cit-Patents	5348328	16518948	5	**3819.00**	12841	236.23	$\emptyset$	∞	5136.00	34.49	3530.40	210.81	1.46	20.69
wiki-Talk	2516783	5021410	8	**1841.00**	3939.20	113.95	$\emptyset$	∞	1948.00	5.81	1806.10	16.70	0.68	8.37
Problematic graph for baseline and proposal														
soc-pokec-profiles	1632803	30622564	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

a representation of the graph in both Tulip and OGDF datastructures, which is memory consuming. During the experimentation, we only captured the execution time of each drawing algorithm (neither graph loading nor rendering time were measured). The computation has been done on a laptop with an Intel® Core™ i7-3840QM CPU@2.80GHzx8, 32Gb of RAM, 64Gb of swap, running Ubuntu 14.04 64bits. All computations have been done only with one core of the CPU, and no tasks were run in parallel. Images are generated only with the biggest component of the graph (but computation is done with all nodes), the colors of each node depends on its coreness, and the transparency of the edges is proportional to the inverse of its length. The intercluster edges are displayed with cubic-b-splines which use the edge bends as control points.

6 Results and Discussion

Table 1 summarises the benchmark of graphs and the performance of each algorithm. We consider that an algorithm is unable to draw a graph if the program meets memory issues or after 36 hours of computation. No method is able to draw *soc-pokec-profiles*, FM^3 and FM^3_{FOR} are unable to draw *soc-LiveJournal1*, and the FM^3_{FOR} is unable to draw *wiki-Talk* and *cit-Patents* due to FOR. Note that these problematic graphs are among the largest ones. These 4 problematic graphs are not used in the following analysis in order to compare only graphs drawable by all the methods.

Computation times of *HCBL* are far better than the computation times of the baselines FM^3 and FM^3_{FOR}, as well as *HCBL* FM^3_{FOR} for all graphs (except for *soc-LiveJournal1* probably because drawing all the subgraphs with FM3 produces smaller clusters and reduce the probability to have to remove overlaps). In term of median, HCBL computes the layout in 6.89 seconds while FM^3, FM^3_{FOR} and *HCBL* FM_{FOR} respectively compute it in 22.80, 79.92 and 15.01 seconds. In order to statistically verify that *HCBL* is faster, we have computed the Wilcoxon signed-rank test on the total computation time with a significance level of $\alpha = 0.05$. The test confirmed that *HCBL* statistically performs better than FM^3, FM^3_{FOR} and *HCBL* FM^3_{FOR} (the 3 p-values are all close to $0.000132 \ll 0.05$).

In average, 5.9% of the time is used to hierarchically cluster the graph, 93.21% of the time to compute the layout of the graphs, 0.06% of the time to compute the final position of the nodes from the relative positions of the metanodes , and 0.81% of the time to build the edges bends.

On that benchmark, most of these structures are quasi-cliques. The performance gain of *HCBL* is explained by the low time complexity of the *Circular* drawing compared to FM^3 and FOR as it that runs in linear time.

Figure 1 shows a sample of graph drawings obtained by *HCBL*. One can easily notice that the three drawings presented in that figure are similar. This is due to the clustering step that oftenly produces a quotient graph with a tree topology for the last level of abstraction. We assume that this is one of the advantages of *HCBL* (in addition to good computation time) as it highlights the main cores of the network while keeping peripheral structures of the network away from the center of the drawing. However, the clusters separation is made at the expense of a large drawing area with empty area that can be solved in a visualization system with interaction tools. In comparison, the drawings produced by FM^3 and FM^3_{FOR} are unsatisfactory due the scale-free and

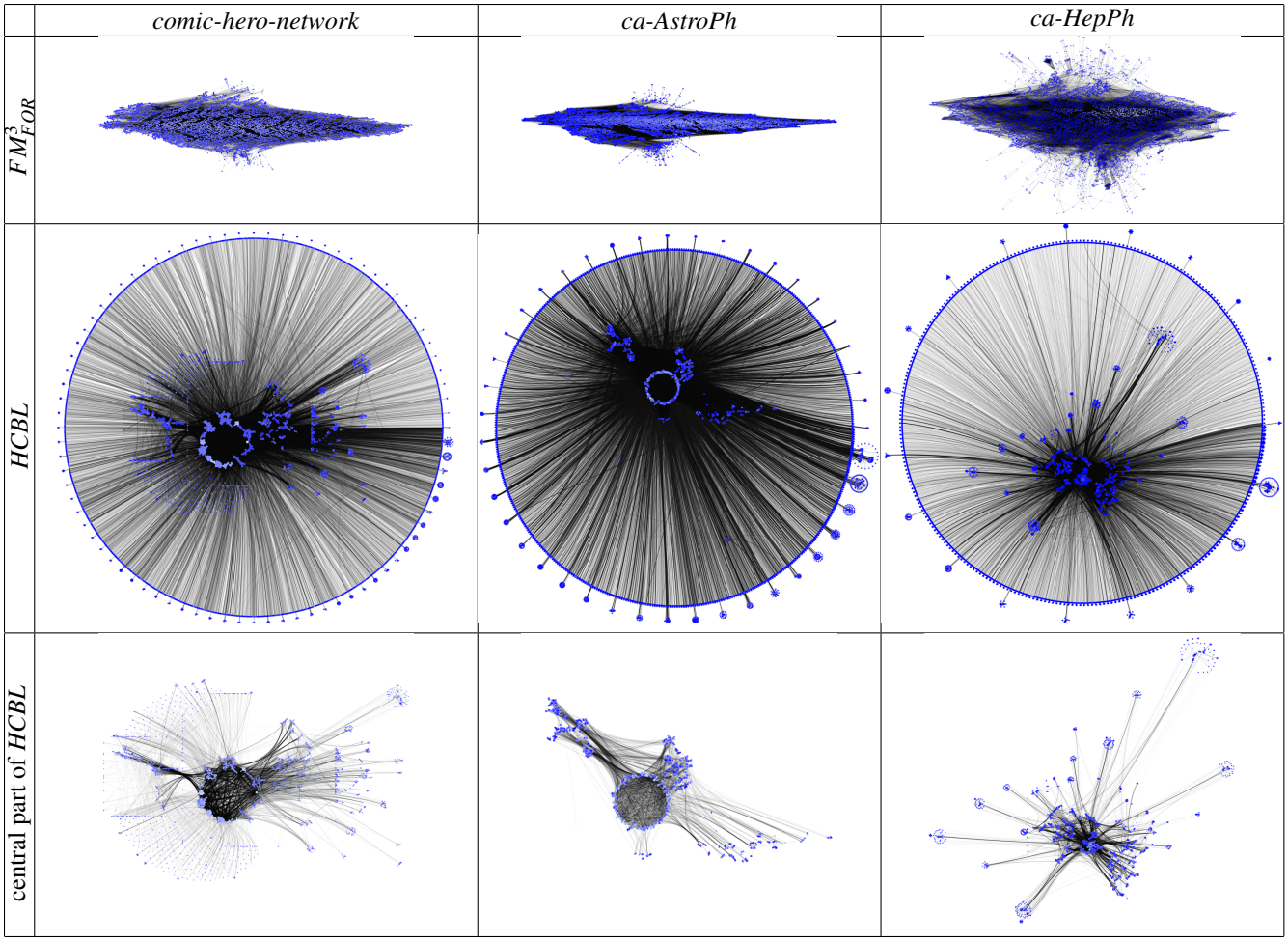

Figure 1: Drawing produced by FM^3_{FOR} baseline (on top), *HCBL* proposal (middle) for *comic-hero-network*, *ca-AstroPh* and *ca-HepPh* graphs and a zoomed view on the central part of these graphs (bottom).

small-world properties of these graphs. For such class of graphs, FM^3 (as well as any other force-directed method) usually produces drawing with an "hairball" effect. This makes the analysis of the graph difficult as it does not emphasize the topological structures of the network. Moreover, it also produces many node overlaps that may also hinder the analysis. As the number of overlaps is large, FM^3_{FOR} does not improve these drawings but rather distorts them.

7 Conclusion

This paper presents a computationally efficient method, *HCBL*, to compute the drawing of large graphs which uses a hierarchical clustering based on the coreness equivalence classes and their connectivities ; recursively computes the drawing of each hierarchical cluster with area and topology-aware drawing algorithms; and uses edge bundling to aggregate edges between nodes of different clusters and thus to reduce visual clutter.

We have compared it against one of the best methods of the state of the art (FM^3) and against an area aware version of it (FM^3_{FOR}) on a benchmark of 23 real-world large graphs. We have statistically verified that the proposed method performs faster than these baselines, and we have empirically verified that *HCBL* provides meaningful results. We have also evaluated a not topology-aware version (that only uses FM^3_{FOR}) which also performs better than the baselines.

We use a redundant data structure to obtain the best performance which may be memory consuming in case of deep hierarchy tree. A future direction is to define a data structure making a trade off between computation time and memory use in order to support the drawing of larger graphs (up to million of nodes and hundred of million edges). Another interesting direction is to constraint the hierarchy tree topology which could reduce the number of metanodes belonging to a quotient graph and the size of the input of each layout algorithm. Finally, it would be interesting to im-

prove the edge bundling step by defining a method avoiding node-edge overlap as well as cluster-edge overlap to better emphasize the hierarchical organization of the network.

Acknowledgments

This work was partially funded by the SpeedData project (PIAO17298-398711).

References

[1] A.T. Adai, S.V. Date, S. Wieland, and E.M. Marcotte. Lgl: creating a map of protein function with an algorithm for visualizing very large biological networks. *Journal Mol Biol*, 340(1):179–190, 2004.

[2] Ricardo Alberich, Joe Miro-Julia, and Francesc Rosselló. Marvel universe looks almost like a real social network. *arXiv preprint cond-mat/0202174*, 2002.

[3] José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. k-core decomposition: A tool for the visualization of large scale networks. *arXiv preprint cs/0504107*, 2005.

[4] D. Archambault, T. Munzner, and D. Auber. Smashing Peacocks Further: Drawing Quasi-Trees from Biconnected components. *IEEE Transactions on Visualization and Computer Graphics (Proc. Vis/InfoVis 2006)*, 12(5):813–820, 2006.

[5] Daniel Archambault, Tamara Munzner, and David Auber. Smashing peacocks further: Drawing quasi-trees from biconnected components. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):813–820, 2006.

[6] Daniel Archambault, Tamara Munzner, and David Auber. Topolayout: Multilevel graph layout by topological features. *Visualization and Computer Graphics, IEEE Transactions on*, 13(2):305–317, 2007.

[7] David Auber, Patrick Mary, Morgan Mathiaut, Jonathan Dubois, Antoine Lambert, Daniel Archambault, Romain Bourqui, Bruno Pinaud, Maylis Delest, Guy Melançon, et al. Tulip: a scalable graph visualization framework. In *Extraction et Gestion des Connaissances (EGC) 2010*, pages 623–624, 2010.

[8] V. Batagelj and M. Zaversnik. An o(m) algorithm for cores decomposition of networks. cs.DS/0310049, 2003.

[9] Michael Baur, Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. Drawing the as graph in 2.5 dimensions. In *Graph Drawing*, pages 43–48, 2005.

[10] Ralf Brockenauer and Sabine Cornelsen. Drawing clusters and hierarchies. In *Drawing graphs*, pages 193–227. Springer, 2001.

[11] B. Cheswick, H. Burch, and S. Branigan. Mapping and visualizing the internet. In *Proc. USENIX*, 2000.

[12] M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, and P. Mutzel. *The Open Graph Drawing Framework (OGDF)*, chapter 17. CRC Press, 2013.

[13] Walter Didimo and Fabrizio Montecchiani. Fast layout computation of hierarchically clustered networks: Algorithmic advances and experimental analysis. In *Information Visualisation (IV), 2012 16th International Conference on*, pages 18–23, 2012.

[14] Tim Dwyer, Kim Marriott, and Peter J Stuckey. Fast node overlap removal. In *Graph Drawing*, pages 153–164, 2006.

[15] A. Frick, A. Ludwig, and H. Mehldau. A Fast Adaptive Layout Algorithm for Undirected Graphs. In *Proc. Graph Drawing 1994 (GD'94)*, pages 388–403, 1994.

[16] T. M. J. Fruchterman and E. M. Reingold. Graph Drawing by Force-directed Placement. In *Software-Practice and Experience*, volume 21(11), pages 1129–1164. nov 1991.

[17] Pawel Gajer and Stephen G. Kobourov. Grip: Graph drawing with intelligent placement. In *Proceedings of the 8th International Symposium on Graph Drawing*, GD '00, pages 222–228, 2001.

[18] Helen Gibson, Joe Faith, and Paul Vickers. A survey of two-dimensional graph layout techniques for information visualisation. *Information Visualization*, 12(3-4):324–357, 2013.

[19] Stefan Hachul and Michael Jünger. Drawing large graphs with a potential-field-based multilevel algorithm. In *Graph Drawing*, pages 285–295, 2005.

[20] David Harel and Yehuda Koren. Graph drawing by high-dimensional embedding. In *Graph Drawing*, pages 207–219, 2002.

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

[22] Takayuki Itoh, Chris Muelder, and Kwan liu Ma. A hybrid space-filling and force-directed layout method for visualizing multiple-category graphs. In *In IEEE Pacific Visualization*, 2009.

[23] Yehuda Koren, Liran Carmel, and David Harel. Ace: A fast multiscale eigenvectors computation for drawing huge graphs. In *Information Visualization, 2002. INFOVIS 2002. IEEE Symposium on*, pages 137–144, 2002.

[24] Jure Leskovec, Daniel Huttenlocher, and Jon Kleinberg. Signed networks in social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1361–1370, 2010.

[25] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, 2005.

[26] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2, 2007.

[27] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[28] Julian J McAuley and Jure Leskovec. Learning to discover social circles in ego networks. In *NIPS*, volume 272, pages 548–556, 2012.

[29] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust management for the semantic web. In *The Semantic Web-ISWC 2003*, pages 351–368. Springer, 2003.

[30] Stephen B Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.

[31] J. M. Six and I. C. Tollis. A framework for circular drawings of networks. In *Proc. Graph Drawing 1999 (GD'99)*, pages 107–116, 1999.

[32] Soon Tee Teoh and Ma Kwan-Liu. Rings: A technique for visualizing large hierarchies. In *Graph Drawing*, pages 268–275. Springer, 2002.