Graph Waves

James Abello*
Daniel Nakhimovich*
abelloj@cs.rutgers.edu
d.nak@rutgers.edu
Rutgers University
New Brunswick, New Jersey

Abstract

We present efficient algorithmic mechanisms to partition graphs with up to 1.8 billion edges into subgraphs which are fixed points of degree peeling [3]. Similarly to [1], we are able to process graphs with tens of millions of edges in seconds and hundreds of millions of edges in minutes. For fixed points that turn out to be larger than a desired interactivity parameter we further decompose them with a novel (to our knowledge) linear algorithm into what we call "graph waves and fragments". This decomposition is used to create spanning views of fixed points that we call "DAG Covers". We illustrate these decompositions by presenting intuitive and interactive visualizations of the meta-structures of a variety of publicly available data sets including social, web, and citation networks [18].

Keywords

graph waves, kcore, graph decomposition, graph visualization

1 Introduction

How far can we simplify computation and visualization in order to maintain "large scale" macro structural graph features without losing the ability to interactively extract more detailed connectivity? Some progress in this direction appears in [3]. In general, macro structural graph views can be obtained by using some form of vertex or edge aggregation that can be represented as hierarchy trees [4, 8, 12, 23, 29]. The choice of suitable representations that facilitate smooth visual interaction is a subject of active research [13, 19, 20, 22]. In [11] an edge stratification technique is proposed that is applied to a given layout of a graph with a few hundred vertices and edges. All of these techniques require algorithms with running time greater than linear in the number of graph elements. Our focus is on graph decompositions that are applicable to graphs of several orders of magnitude larger for which a direct layout is unavailable due to memory constraints or impractical due to screen size contraints.

Some hierarchical decompositions have faster techniques. Major approaches include SPQR-trees, k-core and nuclei based decompositions [1, 5, 10, 15, 17, 24] or even simple stochastic sampling [21, 27]. k-connectivity decomposition techniques have been explored in [4, 6, 16] and terrain metaphors have been explored in [2] and [28]. For large graphs, such techniques are limited because there is no known canonical hierarchical decomposition for k-connectivity. The notion of cores was first introduced in [25] and their utiliy was further identified in [5]. One of the main advantages of core based vertex decompositions is that they can be computed efficiently [10] and they provide an arguably "natural" hierarchical view of any network. The vertex core decomposition has been used to obtain edge partitions where each set of edges in the partition is a maximal edge subgraph that is a fixed point of degree peeling. The maximality of these decompositions is desirable because it provides bounds on the connectivity between the decomposed subgraphs [3]. The pragmatic challenge of computing this iterative edge "layer" decomposition for graphs with several hundred million edges in a few minutes has been undertaken in [1]. However, if a particular layer in the decomposition is large, the navigation and visualization of such big fixed points is still a major task. For example, in the Friendster social network there are four fixed points with over 100 million edges. We address this issue by introducing a novel and efficient algorithm that we call the "wave decomposition". The "waves" produced by the decomposition can be naturally separated into connected sub-waves which can in turn be refined into a collection of "fragments". For large fragments we propose using maximal matching based contractions combined with graph rendering and clustering techniques similar to [9, 30]. Our notion of fragments is quite different to that used in [7]. The ultimate goal of this work is to obtain a humanly-interpretable, hierarchical description of any graph.

First we introduce fixed points and waves in sections 2 and 3. In sections 4 and 5 we show sample visualizations and runtime results. Lastly we discuss our contributions and future work in section 6.

2 Fixed Points and Degree Peeling

We deal with **undirected graphs** $G = (V, E)$ with vertex set $V(G)$ and edge set $E(G)$. We denote by n the number of vertices in V, m the number of edges in E, and the degree of a vertex $u \in V$ by deg(u).

Definition 1. ***(Peel Value)*** *The peel value of a vertex* $u \in V(G)$ *denoted* $peel_G(u)$ *is the largest* $i \in [1, deg(u)]$ *such that* u *belongs to a subgraph of* G *of minimum degree* i.

Definition 2. ***(Graph Core)*** *The core of* G, $core(G)$, *sometimes called the k-core of* G, *is the subgraph induced by the maximal subset of vertices of* G *whose peeling value is maximum.*

*Both authors contributed equally to this research.

Cores can be intuitively thought of as dense subgraphs. Although, they are not necessarily maximally dense in any known way, [26] shows that k-cores are 2-approximate locally dense subgraphs.

In a previous work, [3] utilizes k-cores to produce an iterative decomposition of a graph's edges into layers of fixed points (which we redefine below) with the same peel value. Since the decomposition forms a partition of edges, we call vertices that appear in more than one fixed point, clone vertices. The notion of clone vertices can be used to define a diversity measure and are useful for detecting community overlap [3].

Definition 3. ***(Fixed Point)*** *A graph F_k is a fixed point of degree peeling k if $core(F_k) = V(F_k)$ and the peel value of each vertex in F_k is k.*

From proposition 1 in [3] it is easy to verify that a fixed point of peel value k must have minimum degree k and average degree strictly less than 2k. In Figure 1 we show the distribution of the average degrees of fixed points in the Patents citation and Friendster social networks, along with two black lines indicating the minimum and supremum average degree possible.

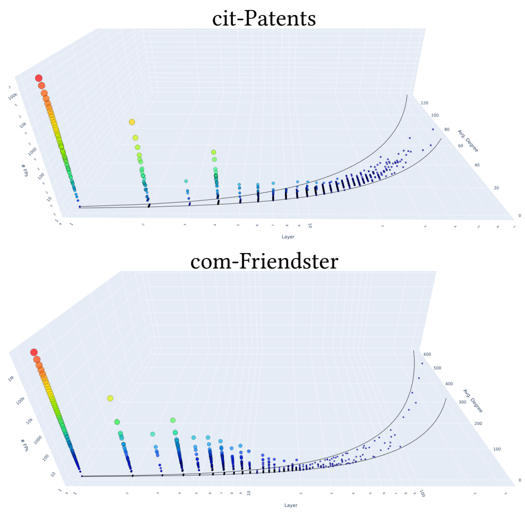

Figure 1: Plots of average degree of fixed points (y-axis) by peel value (x-axis) with frequency of fixed points of a given peel value and average degree along the z-axis being emphasized by color (blue to red) and point size. Note that the x-axis and z-axis are shown on a log scale.

What we call fixed points of degree peeling k, fixed points of peel value k, or simply fixed points, are in fact extensions of forests called k-dense forests in [14]. In that paper the authors first formulate the notion of k-dense cycles as connected graphs where each vertex is of degree > k. From there it is natural to define a k-dense forest as a graph without k-dense cycles. Using any algorithm to find the k-core of a graph, one can find what [14] calls a k-elimination order of a fixed point of degree peeling k. Thus, by Lemma 2.9 in [14] the definition of a fixed point of degree peeling k and the definition of a k-dense forest are equivalent. The importance of noticing that fixed points are generalizations of trees is that it motivates a decomposition of fixed points that works by iteratively peeling "leaf-like" fragments (subsection 3.1).

3 Waves and Fragments

Waves are introduced in order to decompose Fixed points. Previous works do not address decompositions of fixed points if they are large. To further decompose fixed points we introduce a "wave decomposition" which forms an edge partition by iteratively removing "leaf-like" fragments.

3.1 Wave Decomposition

Definition 4. *We define the* ***boundary*** *of a vertex set $S \subset V$ as $\partial S = \{v \in V : \exists\ (u, v) \in E,\ u \in S,\ v \notin S\}$. We define the* ***k-boundary*** *of S as $\partial_k S$ to be the vertices in ∂S with degree less than k restricted to the graph induced by $V \setminus S$. We extend these boundary definitions to a collection of disjoint sets by taking their union.*

Definition 5. ***(Graph Fragment)*** *Given a* ***seed set*** *of vertices $S \subset V$ the* ***fragment*** *generated by S, denoted frag(S), is the set of edges, (u, v), such that $u \in S$.*

In Figure 2 (right) we show a decomposition of a fixed point into fragments. Next, we use graph fragments to introduce the notion of graph waves of fixed points.

Definition 6. ***(Graph Waves of Fixed Points)*** *A* ***wave*** *of a fixed point of peel value k is the union of fragments in the sequence $\{frag(S_j)\}_{j=0}^{m}$ with S_0 being the set of vertices of minimum degree k and each subsequent $S_{j+1} = \partial_k(\cup_{i=0}^{j} S_i)$.*

Notice that a fixed point could consist of only one wave and a wave could consist of only one fragment (i.e. k-regular graphs).

Due to space limitations pseudo code of the wave decomposition, Alg.1, is in the appendix. If we let $k = 1$ then F_1 is a forest and the first fragment is the neighborhood of the leaves of that forest. For $k > 1$, the wave decomposition mimics the iterative removal of leaves in a forest by removing collections of k-leaves (a.k.a fragments) from k-dense forests (a.k.a fixed points of degree peeling) [14]. The example in Figure 2 shows a coloring corresponding to the edge partition formed by applying the wave decomposition. The wave decomposition was heavily inspired by the method of computing k-cores in [10] and much like the implementation of that algorithm the wave decomposition can be computed in linear time and space with respect to the number of edges in a graph.

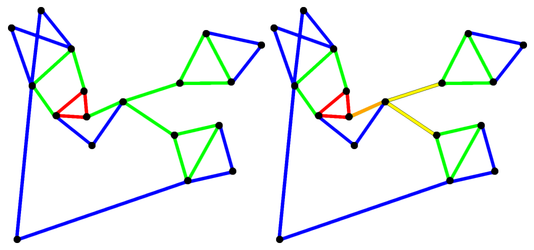

Figure 2: (Left): A fixed point of peel value 2 whose edges are colored based on the wave decomposition (wave 1: blue, wave 2: green, wave 3: red). (Right): The same graph sub-divided into fragments: wave 1 with 1 blue fragment, wave 2 with 3 fragments (green, yellow, orange), and wave 3 with 1 red fragment.

3.2 DAG Cover of Waves and Fixed Points

The concept of fragments is powerful in its own right. Consider the ordered collection of vertex sets $S = \{S_0, S_1, ..., S_j\}$ as defined in Def. 6. These sets form a vertex partition of the wave vertices. We construct a directed acyclic graph (DAG) representation of the wave decomposition by directing thoses edges (x, y) with $x \in S_i$ and $y \in S_{i+1}$ for $0 \leq i < j - 1$. We call this a **DAG cover of a wave** because it contains (or covers) all the vertices of the wave plus the edge cuts between consecutive sets in S. We create a **DAG cover of a fixed point** by using its ordered wave decomposition. Similarly we could create a DAG cover for an entire graph by using its fixed point decomposition. Moreover, we use this DAG cover in conjunction with other techniques to produce graph drawings. In particular, we next illustrate how we use the DAG cover to visualize stratification of fixed points and their waves.

3.2.1 *Visualization of Fixed Points with DAG Cover*

We use three different techniques – color, force manipulation, and edge sparsification – driven by the DAG cover to provide better interpretability of the structure of fixed points.

The most visually striking technique is to use the DAG cover to color the vertices and edges of a graph. First we assign a color to each vertex in the DAG cover based on the sets in S, then we color the edges of the original graph as a function of the color of their end points (e.g. hue average). Using a spectrum color scheme from blue to red, we show a few examples of fixed points (Figure 3) for which the DAG coloring reveals more clearly a stratification structure than an uncolored drawing with the same layout.

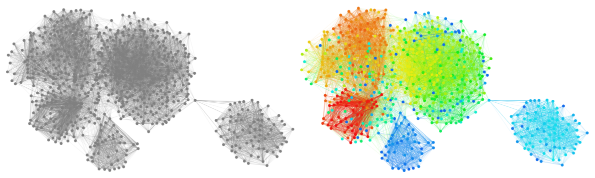

Figure 3: (Left): A fixed point of peel value 22 from the Patents citation network with 742 vertices and 12985 edges shown using a basic force-directed layout in grey. (Right): The same fixed point shown with the DAG coloring.

Another technique we use is to manipulate the forces in a force-directed layout using the information of the DAG set ordering. In a standard force-directed layout, there is a repulsive force between each pair of vertices and an attraction between pairs of vertices connected by an edge. In addition to those forces, we apply unique radial forces to each of the sets of vertices in S centered at increasingly larger radii based on their order in S. Furthermore, we remove forces along the edges which are not part of the DAG cover. This tends to separate out the vertex sets in the layout and helps untangle hairball looking fixed points when the standard force-directed layout does not. Figure 4 shows a fixed point with four sets displayed using a standard force-directed layout and a DAG induced force-directed layout. Unlike the standard layout, the DAG induced layout helps to easily identify independent (or “near-independent”) sets of vertices to highlight a “bipartite-like” structure even without using color.

For a typical, interactive force-directed layout drawing application there is a maximum number of edges that can be rendered before making the system unresponsive. For such cases we could push the system to draw larger fixed points by only rendering the

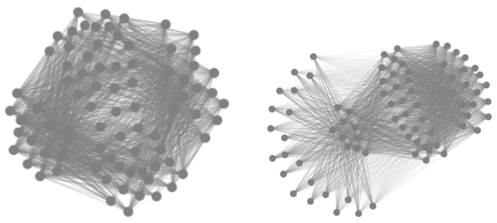

Figure 4: A fixed point of peel value 30 from the Patents citation network with 80 vertices and 1547 edges drawn with a force-directed layout (Left) and with forces applied to the DAG sets (Right).

edges in the DAG cover. Figure 5 shows a graph rendered before and after filtering by the DAG cover. Note that the full graph took on the order of a minute to render on a machine without a dedicated GPU whereas the DAG cover visualization rendered nearly instantly and remained interactive. The fraction of edges in the DAG cover is clearly less than or equal to the number of edges in the original graph and in practice the reduction often exceeds a factor of 2. Despite not having all the edges of the original graph, the graph drawing of just the DAG cover edges in combination with the DAG coloring and DAG set forces often still shows the general graph stucture.

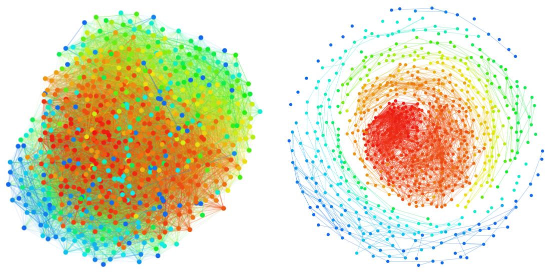

Figure 5: A fixed point of peel value 32 from the Friendster social network with 752 vertices and 21,055 edges drawn with a force-directed layout (Left) and with only the 2,686 DAG cover edges (Right).

4 Visual Exploration

Using the iterative edge core and wave decompositions we have designed an interactive graph visualization tool capable of exploring large graph data up to 1.8 billion edges. The system offers multiple high level views including a spiral view and a layer view that, in our experience, aid in meaningful graph exploration. These structures allow the user to make intuitive decisions as to which subgraphs (created by our decompositions) to zoom in to for deeper analysis. We define interactivity parameters I_e and I_v to be the maximum number of edges and vertices respectively that a user allows to be drawn on their screen. These parameters are predominantly dependent on screen size, GPU power, and user preference for interactivity versus displaying more data points/links at a time.

4.1 Fixed Points Spiral View

To offer users an overall view of the entire collection of fixed points appearing in a very large graph we tabulate the fixed points of degree peeling found through the iterative edge core decomposition. We then create buckets containing connected fixed points. These connected fixed points are grouped together into buckets according to the number of edges. Each i^{th} bucket has fixed points of size s such that $log^{i-1}(m) < s \leq log^{i}(m)$. This bucketing scheme is visually represented by a spiral of boxes that we call the spiral view. It provides direct access to any group of similarly sized fixed points in the graph data set.

4.1.1 *Spiral View*

The spiral view consists of boxes of two types corresponding to the type of fixed points the bucket contains. If a bucket has many fixed points we show a polar bar graph of frequencies of fixed points per unique size. If a bucket has only one fixed point larger than the interactivity parameter, I_e, we show a 2D wave map as described in subsubsection 4.2.1. Notice how in (Figure 6) the size of the spiral view of the larger Friendster network (1.8 billion edges) is about the same as that of the smaller Patents citation network (only 0.16 billion edges). Since the size of the spiral view scales at most logarithmically with graph size it works well for navigating graphs at almost any scale.

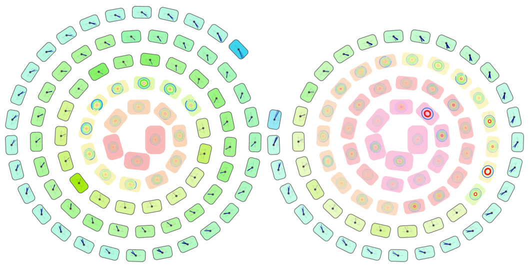

Figure 6: The spiral view of the Patents citation network with 3,774,768 vertices and 16,518,947 edges (Left) and the Friendster social network with 65,608,366 vertices and 1,806,067,135 edges (Right). The colors indicate the relative size of the buckets (number of edges increasing from blue to red) and with stronger opacity indicating higher density.

4.1.2 *Layer View*

The layer view (Figure 7 top) shows a rectangular representation of buckets. The size of each sub-rectangle encodes the overall number of fixed points in the bucket. When a bucket is clicked, either the fixed points in the bucket are displayed sorted by peel value and size, or a 2D wave map (see subsubsection 4.2.1) is shown if there is only one very large fixed point (i.e. size greater than I_e) in the bucket. Actually, the system only shows one graph for each fixed point of the same degree distribution in a bucket and explicitly indicates the multiplicity of such graphs. For example, if a fixed point is annotated as x16 it means that there are 16 fixed points with the same degree distribution as the one being shown. Furthermore, the user can filter the displayed fixed points by peel value using a layer ribbon (Figure 7 bottom left).

4.2 Visualizing Waves and Fragments

For fixed points larger than the interactivity parameter I_e we use the wave decomposition of the fixed point to partition its edges into smaller chunks which can be viewed individually. To have an

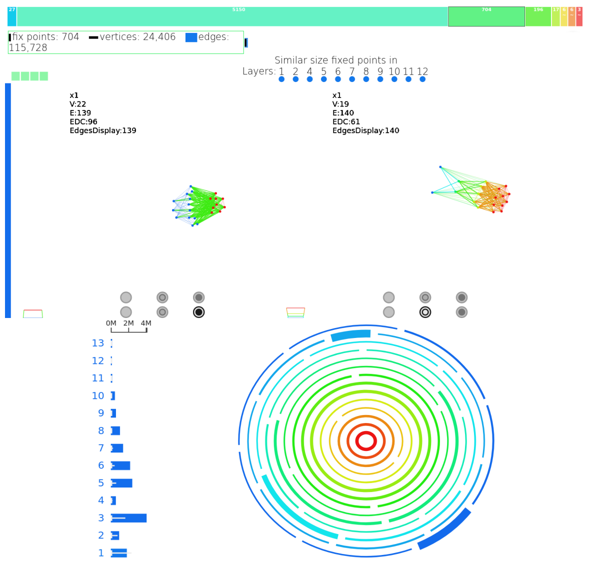

Figure 7: (Top): The layer view of a bucket smaller than I_e. (Bottom Left) The first 13 layers of the layer ribbon. (Bottom Right): The 2D wave map of a bucket larger than I_e which consists of a fixed point with 10,585 vertices and 103,101 edges. All of these are produced from the Patents citation network

overview of the whole fixed point we use 2D and 3D wave maps, that give a sense of the distribution of wave sizes in a fixed point.

4.2.1 *2D Wave map*

A 2D wave map (Figure 7 bottom right) is a collection of concentric rings. Each ring represents a wave and is made of multiple arcs representing connected components (or sub-waves) of a wave. The thickness of any arc represents the number of fragments in the sub-wave and the arc length is proportional to the number of edges in that sub-wave. The arcs are color coded by increasing wave number, from blue to red.

4.2.2 *3D Wave map*

A 3D wave map (Figure 8) is a "vase" made of multiple conic frustums, each representing a wave. The volume of each frustum is proportional to the number of edges in that wave and the area of the intersection of two frustums represents the number of shared vertices between those two waves. Recall, waves form an edge partition of the fixed point so any two waves may share common vertices, however only the shared vertices of adjacent waves are represented in the 3D wave map. Users can either select a wave or individual fragments of wave larger than I_e to view the corresponding subgraph. The color coding of the frustums and the color bar is the same as for the 2D wave map (see subsubsection 4.2.1).

5 Implementation and Results

5.1 Procedure

The process we use to perform the iterative edge-core decomposition is taken from [1]. The implementation of the wave decomposition was partly inspired by the implementation for computing the k-core of a graph in [10]. The connected components implementation used was from the Boost Graph Library. The visualization

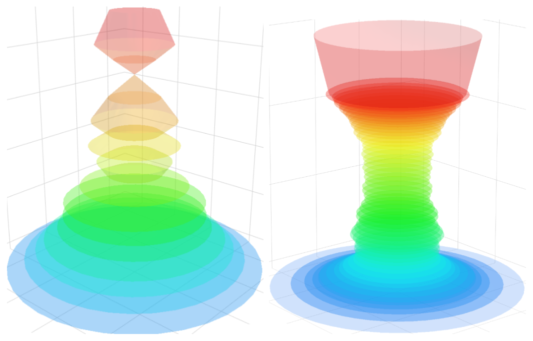

Figure 8: (Left): The 3D wave map of a fixed point of peel value 11 with 18,208 vertices and 151,196 edges from the Patents citation network. (Right): The 3D wave map of a fixed point of peel value 71 with 126,835 vertices and 8,339,470 edges from the Friendster social network.

was developed using D3, three.js, chart.js, plotly.js. To produce our results we performed the following steps:

(1) **Input**: a graph $G = (V, E)$ represented using an edge list.
(2) **Pre-processing:** each edge $e \in E$ is reversed (i.e. $e = (u, v)$ becomes $e = (v, u)$) and appended to the original edge list. The modified edge list is then sorted in increasing order removing duplicates. Additionally we remove self loops, i.e edges of the type $e = (u, u)$.
(3) **Iterative edge-core decomposition:** we use the parallelized implementation of the algorithm in [1] to compute the peel layers of the graph. This assigns a layer value to each edge in the graph. The metadata, like that shown in Table 1 is written to a separate output log.
(4) **Connected Components:** the connected components of the entire graph, as well as of each layer are computed using the Boost Graph Library. The metadata of this calculation (time, number of components) is logged as well as the metadata for each component (size of component).
(5) **Waves/Fragments:** Using the metadata from the previous steps we compute the wave decomposition of fixed points with more than $I_e = 2^{16}$ edges. We also compute the connected components for each wave. The metadata of this calculation (time, number of waves) is logged as well as the metadata for each wave (size, number of components, fragment distribution). Results for the Patents citation network waves are shown in Table 2 and for the Friendster social network in Table 3

We chose a wide range of of graphs, varying in both size and domain (e.g. social, geographic, hyperlink, and co-occurrence networks) [18]. We performed most of our experiments on a single computer equipped with an Intel® Core™ i7-8750H CPU clocked at 2.20GHz with 32GB of RAM. The com-friendster data set however was processed on a server with an Intel® Xeon® CPU E5-2620 v2 clocked at 2.10GHz with 126GB of RAM. Results of the iterative edge-core decomposition are reported in Table 1, which includes the graph that is decomposed, its vertex and edge count, its highest degree, the number of layers and connected component in each graph, the highest peel value and number of waves from the decomposition, and the algorithm compute time and I/O time averaged over 5 runs. Notice that the computation times of the iterative edge core decomposition for graphs with at least 1 million edges grows as $|E|\sqrt{|V|}$. Due to space limitations we include the results of the wave decomposition on the Patents citation and Friendster networks in Table 2 and Table 3 of the appendix.

For our experiments (see Table 2 and Table 3) we computed the wave decomposition on all the fixed points in a layer simultaneously as this was actually quicker than filtering by connected components.

Although not included in Table 2 and 3, we note that layer 1 of the Patents citation network has 534,601 trees totaling 2,370,043 vertices and 1,835,442 edges out of a total 3,774,768 vertices and 16,518,947. What is surprising is that about 63% of vertices in this network are part of a forest. In the Friendster network, layer 1 has 11,222,669 trees totaling 43,053,668 vertices and 31,830,999 edges. So similarly about 65% of vertices in Friendster are part of trees.

5.2 Handling Large Fragments

The sheer size of very large networks is usually dealt with some form of iterative vertex or edge decomposition. The level of granularity of the decomposition may be driven by storage or computational resources coupled with graph structure. The iterative edge core decomposition first introduced in [3] has been used in [1] as a useful graph abstraction to make sense of very large graphs. In this work, we go two steps beyond and decompose "large" fixed points into waves and if they are still "large" we decompose them further into fragments (see subsection 4.2). It is worth mentioning that the overall approach is basically the same, i.e., iterative removal of vertices that satisfy certain degree conditions. Fragments can be viewed as the most atomic graph types obtained by plain vanilla iterative degree removal methods. However, the main limitation is that these fragments can be "large" too and require specialized methods beyond the degree peeling based approaches.

In these situations we propose a variation of the maximal matching edge contraction approach first suggested in [4]. Namely, we iteratively select a random maximal matching $\{e_1, e_2, ..., e_j\}$ and contract its edges until the set of vertices remaining have cardinality equal to the interactivity parameter, I_v. This contracted graph can then be visualized as a metagraph, where each vertex represents the subgraph of contracted edges. The combination of contractions and matchings is an area of study that deserves further research.

6 Summary and Future Work

Our approach presents a high level computational overview of "large graphs" as sequences of "waves". This is fundamentally different to previous approaches in the sense that it is amenable to a visual representation (i.e. wave maps) that is closely tied to "intuitive" hierarchical navigation and exploration. Our next step will be to perform user experiments to evaluate the usability of the proposed system. To our knowledge this is the first type of system that offers a high level representation of graphs at the billion edge scale that can be visually explored at different levels of connectivity within a global context.

We believe that graph partitioning processes will become useful tools to address massive graph computations in a divide and conquer manner. This line of thinking opens up research to investigate which types of massive graph problems can be solved by composing their local fixed point and wave solutions.

Table 1: Results of performing the iterative edge-core decomposition across a number of different graphs varying in size and domain sorted by number of edges. (CC: number of connected components MP: max peel value, L: number of layers, MW: max number of waves)

Graph Name	|V|	|E|	Max Deg	CC	MP	L	MW	Time (s)	I/O Time (s)
Gnutella P2P network (p2p-gnutella31)	62586	147892	95	12	6	5	9	0.05	0.05
Astro Physics (ca-astroph)	18771	198050	504	289	56	47	10	0.14	0.10
Amazon co-purchasing (amazon0601)	403394	2443408	2752	7	10	10	22	1.00	0.98
California road network (roadNet-CA)	1965206	2766607	12	2638	3	3	63	1.34	1.33
Berkeley-Stanford web (web-BerkStan)	685230	6649470	84230	676	201	88	314	5.37	3.50
Patents citation network (cit-Patents)	3774768	16518947	793	3627	64	41	47	18.37	13.33
Pokec (soc-pokec)	1632803	22301964	14854	1	47	29	45	15.62	14.86
LiveJournal network (LiveJournal)	3997962	34681189	14815	1	360	119	49	54.04	35.73
Orkut (com-orkut)	3072441	117185083	33313	1	253	91	88	81.62	56.90
Friendster (com-friendster)	65608366	1806067135	5214	1	304	72	213	3085.42	2351.32

We close by mentioning that being able to efficiently compute the proposed wave decomposition in semi-external memory settings or in a streaming fashion will enhance in a major way the applicability of our approach not only to "large graph visualization" but to massive graph computation in general.

A video demonstrating our current prototype can be accessed at: https://dl.dropboxusercontent.com/s/a4kpwv5609op73w/graphwaves_bigvis.avi. Other formats are available at: https://www.dropbox.com/sh/bowfhdrr82ti18u/AAAAFwUZwkq5pQD63Kqyh-Ela?dl=0.

Acknowledgments

We thank Qi Dong for helping to develop graph drawing software, Yi-Hsiang Lo for interface design, and Haodong Zheng for interface design and system integration. This work was partially supported by NSF grants IIS-1563816 and IIS-1563971.

References

[1] James Abello, Fred Hohman, Varun Bezzam, and Duen Horng Chau. 2019. Atlas: Local Graph Exploration in a Global Context. In *Proceedings of the International Conference on Intelligent User Interfaces*. ACM.

[2] James Abello and Shankar Krishnan. 2000. Navigating graph surfaces. In *Approximation and Complexity in Numerical Optimization*. Springer, 1–16.

[3] James Abello and François Queyroi. 2013. Fixed points of graph peeling. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. ACM, 256–263.

[4] James Abello, Frank Van Ham, and Neeraj Krishnan. 2006. Ask-graphview: a large scale graph visualization system. *IEEE Transactions on Visualization and Computer Graphics* 12, 5 (2006), 669–676.

[5] J Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. 2006. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in Neural Information Processing Systems*. 41–50.

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

[7] Alessio Arleo, Oh-Hyun Kwon, and Kwan-Liu Ma. 2017. GraphRay: Distributed pathfinder network scaling. In *2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV)*. IEEE, 74–83.

[8] Benjamin Bach, Nathalie Henry Riche, Christophe Hurter, Kim Marriott, and Tim Dwyer. 2017. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Transactions on Visualization & Computer Graphics* (2017), 1–1.

[9] Vladimir Batagelj, Franz J Brandenburg, Walter Didimo, Giuseppe Liotta, Pietro Palladino, and Maurizio Patrignani. 2011. Visual analysis of large graphs using (x, y)-clustering and hybrid visualizations. *IEEE transactions on visualization and computer graphics* 17, 11 (2011), 1587–1598.

[10] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O(m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).

[11] Emilio Di Giacomo, Walter Didimo, Giuseppe Liotta, Fabrizio Montecchiani, and Ioannis G Tollis. 2014. Techniques for edge stratification of complex graph drawings. *Journal of Visual Languages & Computing* 25, 4 (2014), 533–543.

[12] Tim Dwyer, Nathalie Henry Riche, Kim Marriott, and Christopher Mears. 2013. Edge compression techniques for visualization of dense directed graphs. *IEEE transactions on visualization and computer graphics* 19, 12 (2013), 2596–2605.

[13] Dezhi Fang, Matthew Keezer, Jacob Williams, Kshitij Kulkarni, Robert Pienta, and Duen Horng Chau. 2017. Carina: interactive million-node graph visualization using web browser technologies. In *International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 775–776.

[14] Gianni Franceschini, Fabrizio Luccio, and Linda Pagli. 2006. Dense trees: a new look at degenerate graphs. *Journal of Discrete Algorithms* 4, 3 (2006), 455–474.

[15] Christos Giatsidis, Fragkiskos D Malliaros, Nikolaos Tziortziotis, Charanpal Dhanjal, Emmanouil Kiagias, Dimitrios M Thilikos, and Michalis Vazirgiannis. 2016. A k-core Decomposition Framework for Graph Clustering. *arXiv preprint arXiv:1607.02096* (2016).

[16] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the national academy of sciences* 99, 12 (2002), 7821–7826.

[17] Humayun Kabir and Kamesh Madduri. 2017. Parallel k-core decomposition on multicore platforms. In *IEEE International Parallel and Distributed Processing Symposium Workshops*. IEEE, 1482–1491.

[18] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[19] Zhiyuan Lin, Nan Cao, Hanghang Tong, Fei Wang, and U Kang. 2013. Interactive multi-resolution exploration of million node graphs. In *IEEE Conference on Visual Analytics Science and Technology, Poster*.

[20] Peng Mi, Maoyuan Sun, Moeti Masiane, Yong Cao, and Chris North. 2016. Interactive graph layout of a million nodes. In *Informatics*, Vol. 3. Multidisciplinary Digital Publishing Institute, 23.

[21] Quan Hoang Nguyen, Seok-Hee Hong, Peter Eades, and Amyra Meidiana. 2017. Proxy graph: visual quality metrics of big graph sampling. *IEEE transactions on visualization and computer graphics* 23, 6 (2017), 1600–1611.

[22] Robert Pienta, Fred Hohman, Alex Endert, Acar Tamersoy, Kevin Roundy, Chris Gates, Shamkant Navathe, and Duen Horng Chau. 2018. VIGOR: interactive visual exploration of graph query results. *IEEE Transactions on Visualization and Computer Graphics* 24, 1 (2018), 215–225.

[23] Loïc Royer, Matthias Reimann, Bill Andreopoulos, and Michael Schroeder. 2008. Unraveling protein networks with power graph analysis. *PLoS computational biology* 4, 7 (2008), e1000108.

[24] Ahmet Erdem Sariyuce, C Seshadhri, Ali Pinar, and Umit V Catalyurek. 2015. Finding the hierarchy of dense subgraphs using nucleus decompositions. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 927–937.

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

[26] Nikolaj Tatti. 2019. Density-friendly graph decomposition. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 13, 5 (2019), 54.

[27] Wouter van Heeswijk, George HL Fletcher, and Mykola Pechenizkiy. 2016. On structure preserving sampling and approximate partitioning of graphs. In *Proceedings of the 31st Annual ACM Symposium on Applied Computing*. ACM, 875–882.

[28] Yang Zhang, Yusu Wang, and Srinivasan Parthasarathy. 2017. Visualizing attributed graphs via terrain metaphor. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1325–1334.

[29] Hong Zhou, Panpan Xu, Xiaoru Yuan, and Huamin Qu. 2013. Edge bundling in information visualization. *Tsinghua Science and Technology* 18, 2 (2013), 145–156.

[30] Michael Zinsmaier, Ulrik Brandes, Oliver Deussen, and Hendrik Strobelt. 2012. Interactive level-of-detail rendering of large graphs. *IEEE Transactions on Visualization and Computer Graphics* 18, 12 (2012), 2486–2495.

Appendix

Alg. 1: Wave Decomposition (O(m))

```
Input: $F_k = (V, E)$, a fixed point of peel value k.
Output: $M = \{W_1, W_2, W_3, ..., W_m\}$ where each $W_i \subset M$ are waves.
Output: $S = \{S_0, S_1, S_2, ..., S_n\}$ where each $S_j \subset V$.
function waves($F_k$):
    $M \leftarrow \emptyset$
    $S \leftarrow \emptyset$
    $i \leftarrow 1$
    $j \leftarrow 0$
    while $E \neq \emptyset$ do
        $W_i \leftarrow \emptyset$
        $S_j \leftarrow \{v \in V : \mathrm{deg}(v) = k\}$
        while $S_j \neq \emptyset$ do
            $S \leftarrow S \cup \{S_j\}$
            $E \leftarrow E \setminus \mathtt{frag}(S_j)$
            $W_i \leftarrow W_i \cup \mathtt{frag}(S_j)$
            $S_{j+1} \leftarrow \{v \in \partial S_j \mid \mathrm{deg}(v) < k\}$
            $j \leftarrow j + 1$
        end
        $M \leftarrow M \cup \{W_i\}$
        $i \leftarrow i + 1$
    end
end
```

Table 2: Results of performing the wave decomposition on layers 2 through 13 of the Patents citation network. These are only the layers that contain a fixed point of size at least 2^{16} edges. (L: layer, F_k: number of fixed points, W: number of waves, F: number of fragments). The highlighted row corresponds to the layer containing the largest fragment of 1,356,327 edges.

L	|V|	|E|	F_k	W	F	Time (s)
2	773841	953738	7682	7	55	2.779
3	1663386	4097177	37	33	411	18.587
4	198984	579406	787	15	267	1.652
5	617644	2457675	42	17	610	8.187
6	438875	2210942	100	47	996	8.935
7	246576	1406616	66	19	619	4.02
8	158620	1051389	85	44	765	3.243
9	80554	590337	61	25	511	1.46
10	56961	469522	73	45	579	1.211
11	20602	168159	80	14	188	0.272
12	13507	126909	73	12	236	0.192
13	15043	154245	38	13	284	0.249

Table 3: Results of performing the wave decomposition on select layers of the Friendster social network. These are only the layers that contain a fixed point of size at least 2^{22} edges. (L: layer, F_k: number of fixed points, W: number of waves, F: number of fragments). The highlighted row corresponds to the layer containing the largest fragment of 21,128,481 edges.

L	|V|	|E|	F_k	W	F	Time (s)
2	20031131	26084775	39188	8	99	168.9
3	20912756	46640174	564	27	564	887.1
4	1509262	4760942	4816	11	317	27.0
5	14030072	54607126	447	10	536	314.2
6	6352083	32606979	793	25	1177	233.0
8	2882071	19767228	193	14	903	116.1
11	13635761	131376950	5	75	2532	1422.0
15	5011964	68883449	9	46	1874	496.0
17	738488	11279923	16	18	970	50.8
21	432014	7931887	5	106	978	40.9
25	10460847	229286219	1	17	1127	1511.5
29	244598	6430697	2	15	722	23.4
38	3144226	107272464	1	213	3279	1828.4
40	216081	8035443	2	50	889	31.4
53	6878498	322297063	1	10	1260	2543.1
65	155562	9282393	1	7	415	31.6
71	126835	8339470	1	56	768	30.9
89	4570064	367562291	1	49	737	5647.4
120	908835	97669888	1	2	367	418.0
140	764401	94970270	1	7	443	410.1
169	228332	32277749	1	29	357	115.6
175	197091	32971011	1	40	931	121.1
234	157908	32985831	1	128	1188	123.8
304	24528	6301889	1	7	203	19.2