Graph Cities: Their Buildings, Waves, and Fragments

James Abello[1] Daniel Nakhimovich Chengguizi Han Mridul Aanjaneya
Department of Computer Science, Rutgers University, USA
[1]DIMACS, Rutgers University, USA

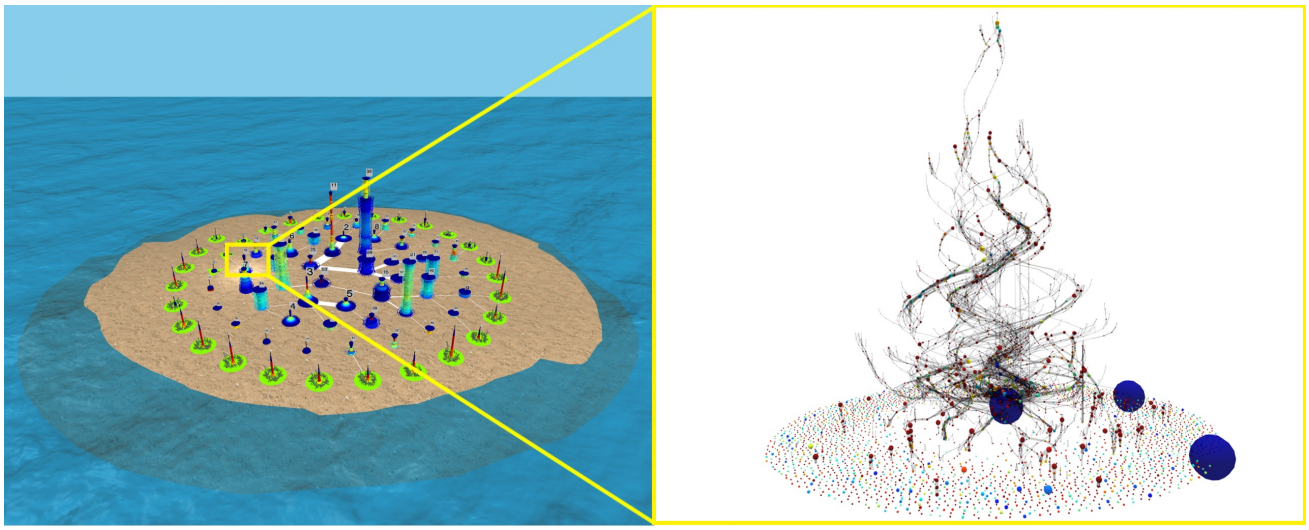

Figure 1: When a direct graph layout is unavailable or impractical due to memory or screen size constraints, our method can render humanly interpreted visualizations of massive graphs on the order of 10 seconds (after a few minutes of preprocessing) by leveraging the *Graph Waves* [3] decomposition. (Left) A Graph City for the Friendster social network with 1.8 billion edges. (Right) The internal structure of a building from the city with peel value 13 showing finer scale connectivity. By leveraging hierarchical decompositions, our framework allows for interactive visualization of massive graphs.

ABSTRACT

"Graph Cities" are 3D representations of maximal edge graph partitions. Each connected equivalence class corresponds to a "Building" that is formed by stacking graph "Edge Fragments". The number of such graph edge fragments determines the height of the building. The overall number of buildings is the number of equivalence classes in the edge partition. A poly-log bucketization of the size distribution of the equivalence classes is used to generate a 2D position for each bucket. For the buckets containing more than one equivalence class, we also generate a visual "Bush" representation. The Delaunay triangulation of these building locations determines the "street network" of the Graph City. The weight of a connection between two buildings on this street network is proportional to the intersection of the subgraph vertex sets represented by the two buildings. To handle equivalence classes (i.e., buildings) consisting of a large number of fragments, we use the notion of "Graph Waves" from [3]. Graph Waves are intervals of graph edge fragments with a "well-defined" beginning and end fragment. For computational purposes, the beginning and end fragments should satisfy a computationally "easy to verify" property. We illustrate Graph Cities obtained with the maximal edge partitions defined by the iterative edge core decomposition introduced in [4]. The graphs used include the Friendster social network (1.8 billion edges), a co-occurrence keywords network derived from the internet movie database (115 million edges), and a patents citation network (16.5 million edges). For graphs with up to 2 billion edges, all the elements of their corresponding Graph Cities are built in a few minutes (excluding I/O time) and storage proportional to the number of edges and vertices of a graph. Our ultimate goal is to obtain humanly-interpretable hierarchical descriptions of any graph that are accessible via a Unified Web Interface for Graph Analytics, without being constrained by the graph size.

1 INTRODUCTION

Techniques for analyzing massive data sets are becoming central to several communities, with the need for interactive and scalable visualizations being one of the most pressing issues [27]. Since a large variety of massive data sets can be abstracted as having an underlying graph topology, our interest is in computing graph decompositions that are useful for making sense of *massive* graphs, for which a direct layout is unavailable due to memory constraints or impractical due to screen size constraints. We focus on the identification of "global data patterns" that emerge due to the co-occurrence of pairs of well-defined data entities.

Motivated by [12] who proved that the graph degree sequence solely determines the expected Hopfield network pattern stability, we wanted to find an "efficient" visual representation of any graph that is driven by the dismantling of its degree distribution. We introduce *Graph Cities* as a visual representation of such dismantling. It has the potential of bringing together streaming computations and visualization to offer "large scale" structural graph information *without* losing the ability to interactively extract finer scale connectivity. All this is possible by streaming Graph Waves [3], which are in turn streams of graph fragments. The obtained visual representations can be rendered in a few minutes and offer humanly interpretable large scale features of graphs with billions of edges at different levels of granularity.

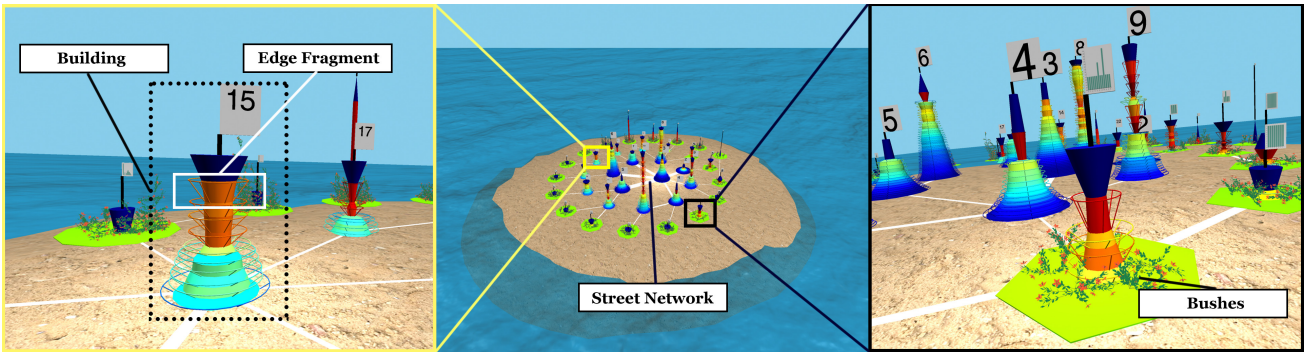

Figure 2: Overview of our proposed framework. (Left) The individual buildings in the graph city, which comprise of cylindrical edge fragments. (Middle) The entire graph city. (Right) Bushes representing buckets with more than one building.

The coarsest views are offered by Graph City buildings with floors and frustums representing Waves and their interconnecting Edge Fragments. Each building internal structure is represented by a Directed Acyclic Graph (DAG) whose macro-vertices can be expanded into their internal fragments, that can be navigated by usual node link diagrams at different levels of detail depending on their size. In summary, our approach follows a hierarchical edge decomposition with five levels: Edge Graph Decomposition, Buildings, Waves, Edge Fragments, with the bottom level consisting of "structured" node-link subgraphs of *reasonable* size that make them amenable to *human descriptions*. To our knowledge, this is the **first** time that such a large scale representation has been introduced for visualizing graph data sets, that is humanly interpretable at "natural" topological levels of granularity.

1.1 Summary of Our Overall Approach

An overview of our proposed framework is illustrated in Figure 2. Our work builds upon the *Graph Wave* decomposition recently introduced in [3], which is a refinement of the iterative degree edge partition [4]. Each connected equivalence class is visually represented as a "building". The size distribution of the equivalence classes in the given edge partition is bucketed via a poly-log function of the total number of edges in the entire data set. Buckets containing more than one building are visually represented by "bushes" that are generated via special L-systems (see green areas in Figure 2(right)). A spiral embedding of this poly-log expression provides a layout for all the equivalence classes in a graph city.

Each graph city has a "street network" that is obtained by computing the intersection graph of the collection of vertex sets of the buildings. The 2D positions of the centers of the bottom floors of the buildings are used to indicate the location of the building vertex sets. The weight of the geometric edge representing the distance between two buildings is obtained as a function of the size of the intersection of the corresponding buildings vertex sets (i.e., a *Jaccard-type* coefficient).

All the elements of a graph city including its buildings, bushes, and street network are built in a few minutes (excluding I/O time) and storage proportional to the number of edges and vertices of the graph. We exemplify our results on a variety of large graph data sets ranging in size from 1 million to over 2 billion edges. They include the Friendster social network (1.8 billion edges), a co-occurrence keywords network derived from the internet movie database (115 million edges), and a patents citation network (16.5 million edges). To summarize, our main contributions are:

- Graphs become represented as a collection of *buildings with floors*, with inter-floor volumes encoding subgraph sizes.
- A 2D spiral layout is used to fix building locations, and also for visualizing the street network, whose edge widths are determined by the intersection graph of the building vertex sets.
- Smaller buildings are bucketed into "green" areas of bushes that are generated by special natural-looking L-systems.
- Graph Cities can be navigated at different levels of granularity.
- The largest buildings within graphs with over a billion edges are rendered in a few seconds. The algorithms computing the data required to specify the rendering of a graph city from a given wave decomposition are linear both in time and storage.

As far as we know, Graph Cities constitute the *first* abstract visual representation of node link representations of graphs that are linearly computable, and that are amenable to interactive topological exploration at different levels of granularity for massive graph data sets. We plan to provide browser access to our tools for the exploration of graphs with up to a few billion edges. In this paper, we focus on the computational techniques to provide scalable visible abstractions for large graphs. In the future, we will couple the graph city abstraction with semantic information.

The paper layout is as follows: Section 2 summarizes relevant work. In Section 3, we introduce the main definitions and illustrate the main conceptual tools borrowed from a previous work: graph Edge Fragments and Graph Waves [3]. Section 4 introduces *Graph Cities*, their buildings, bushes, waves and fragments, and Section 5 describes what the drawn street network represents. Section 6 discusses 3 data sets used, and statistics of their size and various decompositions. Sections 7 and 8 discuss the rendering of graph cities and their interactive navigation. Section 9 briefly mentions end-user goals. Section 10 discusses avenues for future research and Section 11 closes with concluding remarks.

2 RELATED WORK

Efforts to deal with large graphs have mainly employed computational approaches to handle scale. Generally, computation and visualization appear to be treated as independent tasks. One approach is to develop scalable algorithmic tools that amplify users' understanding of the underlying data topologies at different levels of granularity. In general, macro graph views can in principle be obtained by some form of vertex or edge aggregation that is conceptually represented as hierarchy trees [5, 8, 13, 26, 32]. The choice of representations that facilitate smooth visual interaction is a subject of active research [19, 22, 23, 25].All these previous techniques have algorithms with running time *substantially greater than linear* on the number of graph elements, making them not suitable for massive graph visualization. Sampling, as a mechanism to shed light on graph structure, has been explored in

[24]. Graph Thumbnails, as a mechanism to identify and compare multiple graphs, are alone the subject of [31]. Uses of the Core decomposition to grasp some coarse graph topological views are discussed in [15]. Generation of graphs with a predefined core structure is the focus of [10]. Computational aspects of core related graph decompositions and graph sparsification are studied in [7, 9, 18, 20, 28, 30].Some algorithmic principles for graph reachability in large graphs are proposed in [17]. Machine learning approaches, such as those described in [16], have been proposed to learn low-level embeddings of graphs, however, such approaches are not yet scalable to billion edge graphs.

Our approach differs from prior work in the sense that we look for efficient data traversal algorithms *via* exploration primitives that lend themselves to visual representations that amplify users' understanding of the internal graph structure for graphs that are too large to be visualized directly. Buildings, Waves, and Fragments are examples of such primitives.

3 PROBLEM FORMULATION

We aim to provide visual abstractions for representing edge partitions of graphs to highlight connectivity and size distribution of subgraphs with special degree distributions - fixed points in our case (see Def. 3). We achieve this by first computing a maximal edge partition of the graph as in [4]. We then represent the equivalence classes of the given edge partition as "buildings". Each building consists of a collection of "Waves", each of which is formed by a stack of "Edge Fragments" [3], i.e., Waves are ordered maximal sequences of graph Edge Fragments (see Def. 5 and 6). This section will first describe what the graph structures called fixed points, waves, and fragments are and then the visual representations we use to display them will be described in Sec. 4.

3.1 Graph Edge Fragments and Waves

One approach to getting a sense of the topology of a graph begins with a choice of a starting set of vertices S_0 satisfying a property of interest P and marking them as "explored". All the edges with at least one endpoint in S_0 are then traversed, deleting them from the graph and adding them to the "Edge Fragment" associated with S_0. These edges then become the beginning part of the "Graph Wave" generated by S_0. If there are any edges with one endpoint not in S_0, we check whether those vertices not in S_0 still satisfy a property of interest Q (usually, a relaxation of P) in the remaining graph. If there are any such vertices we continue exploring in parallel only from such vertices, adding incrementally the new found edges into the current Wave started by S_0, and deleting them from the existing graph. This process ends when all edges with exactly one endpoint in the current Wave lead to vertices that do not satisfy P. This means that if there still remains "unexplored" edges, a new Wave can be initiated by selecting another starting set of vertices satisfying a property of interest R (usually, stricter than P). We formalize the process above with the following definitions.

3.2 Definitions

We consider **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 $\deg(u)$. For any $U \subset V$, let the notation $G(U)$ denote the subgraph of G induced by U. For the sake of completeness, we restate some definitions from [4].

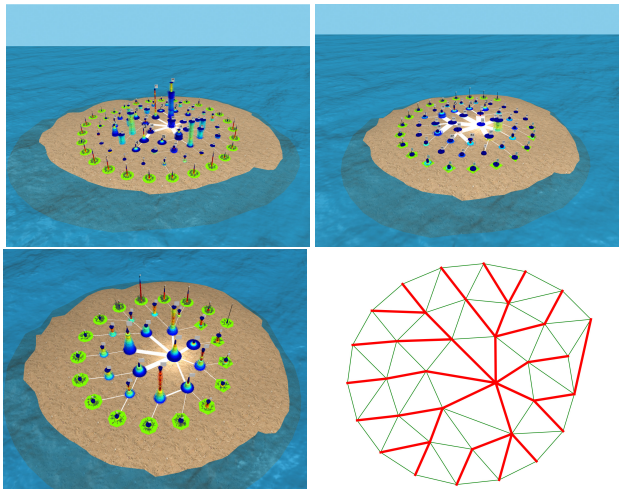

Figure 3: Graph cities for (top left) the Friendster network (1.8 billion edges), (top right) movie phrase co-occurrence network (115 million edges), and (bottom left) the patent citation network (16.5 million edges). The Delaunay triangulation (bottom right) of the spiral building layout (green). The red street network corresponds to the white street network in the city on the bottom left.

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, denoted $core(G)$, sometimes also called the k-core of G, is the subgraph induced by the maximal subset of vertices of G whose peel value is maximum.*

Definition 3. *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.*

Figure 4 shows vertices in a small graph colored by their peel value. It also shows edges colored according to the iterative edge decomposition from [4] which obtains an edge partition by removing the highest peel value edges (i.e., initially the red edges), updating the vertex peeling values, recoloring the affected edges, and identifying the next highest peel value edges and repeating until the whole graph is processed. A graph city (Sec. 4) can be derived from this edge partition by mapping each connected edge color class into a building.

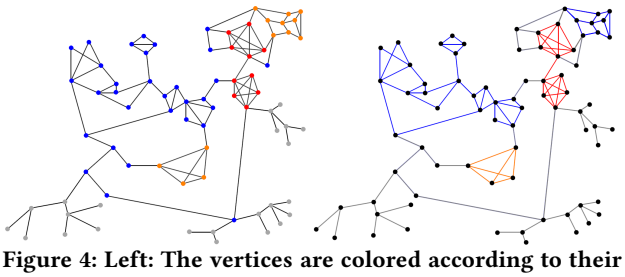

Figure 4: Left: The vertices are colored according to their peel value: 1-grey, 2-blue, 3-orange, 4-red. Right: The edges are colored according to the iterative edge decomposition using the same color scale.

3.3 Waves and their Edge Fragments

Next, we present some new definitions that generalize the notion of graph Edge Fragments and Waves introduced in [3].

Definition 4. *The **boundary** of a vertex set $S \subset V$ is defined as $\partial S = \{v \in V : \exists (u, v) \in E, u \in S, v \notin S\}$. The **proper***

boundary *of S, denoted $\partial_P S$, is the set of vertices in ∂S which satisfy a desired property P restricted to the graph induced by $V \setminus S$. These definitions are extended, in a straightforward fashion, to a collection of disjoint sets by taking their union.*

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

Definition 6. *A* ***graph wave*** *$W(S_0, P)$ is the union of fragments in the sequence $(frag(S_j))_{j=0}^{m}$, with S_0 being the source set of vertices satisfying P and each subsequent $S_{j+1} = \partial_P(\bigcup_{i=0}^{j} S_i)$.*

3.3.1 Why are Graph Waves useful abstractions? Waves were inspired by previous graph exploration approaches based on Sparse Nets [2] and Boruvka MST type contraction algorithms [1]. Graph Waves generated by minimum degree source sets were introduced in [3]. In this work, we generalize this notion and use Edge Fragments to describe and visually represent the Waves' internal structure (see subsection 3.3.2 below).

Graph Waves provide different "lenses" into the structure of a graph according to a particular property or substructure. For example, if S_0 is non-empty and consists of the set of vertices not in a given maximal edge-matching M, then the Wave generated by such an initial set S_0 provides a layered view of G as a sequence of independent sets, and the last Wave Edge Fragment is a perfect matching. This is because, for this example, S_0 is the complement of a maximally matched set of vertices. We refer to this example as the Maximal Matching Wave. Graph Waves derived from a graph's degree distribution are essential for discovering the non-regular macro structure of very large graphs, and at the same time help isolate Edge Fragments with peculiar levels of regularity. For example, if k is the minimum degree of a graph G and if S_0 consists of all the vertices of degree k, and subsequent boundary sets are restricted to have degree strictly less than this minimum k, the corresponding Wave is called a Fixed Point of Degree k in [4] with a vertex source set of minimum degree. We refer to these subgraphs as Minimum Degree Waves.

Graph Waves can be adapted to the particular properties of the boundary vertex sets and Edge Fragments being discovered during the algorithmic exploration of a large unknown graph topology. Efficiently and automatically determining the most appropriate properties to generate the Waves of a particular graph is an interesting direction for future work. In this work, we only use Waves generated based on vertex degree thresholds.

3.3.2 Meta-DAG Internal Structure of a Wave(S_0,P). Since the edges of a Wave(S_0, P) are obtained by taking the union of Edge Fragments in the sequences $(frag(S_j))_{j=0}^{m}$, where $S_{j+1} = \partial_P(\bigcup_{i=0}^{j} S_i)$, the Wave vertex set is an ordered partition of subsets $(S_0, S_1, ...S_m)$. We use the connected components of the subgraphs induced by each subset to define a Meta-DAG, where each macro-vertex represents such connected components. Weighted directed meta-edges $((C_{j,u}, C_{l,v}), W_{u,v})$ encode nonempty set of edges $\{(x, y) : x, y \in C_{j,u} \cup C_{l,v}\}$, where $C_{j,u}$ and $C_{l,v}$ are connected components of S_j and S_l. Finally, the weight $W_{u,v}$ encodes the density of edges running between $C_{j,u}$ and $C_{l,v}$.

The spanning subgraph of this metagraph that consists of only those meta-edges that connect components present in consecutive vertex sets, S_j, S_{j+1}, is called the *Spanning Meta-DAG* of the Wave (shown in Figure 5). It describes some of the most fundamental directed internal macro-connectivity of a Wave and is expected to be substantially smaller that the Wave itself. An extreme case in which this is not the case is when the Wave is a tree with the source set being the tree leaves. However, in this case we already have a simple description of the Wave: it is a tree with the number of fragments equal to the radius of the tree.

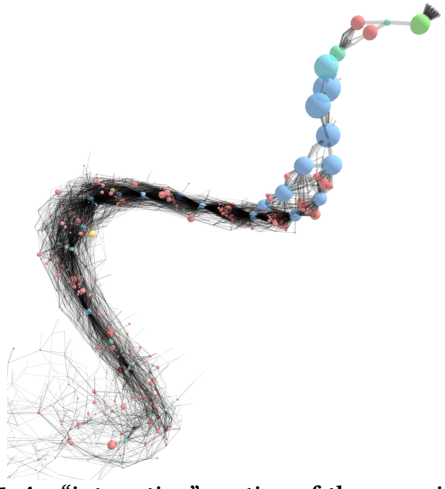

Figure 5: An "interesting" portion of the spanning Meta-DAG of a fixed point of peel value 101 from the movie phrase co-occurrence network. The whole fixed point contains 17,311 vertices and 182,085 edges while the spanning Meta-DAG only has 4181 vertices and 10,016 edges.

4 WHAT IS A GRAPH CITY?

We assume that the input is some ordered partition $(E_0, E_1, ..., E_z)$ of the edges of a graph $G = (V, E)$, where each E_i is edge-maximal with respect to a predefined property. An efficient representation for such partitions is stored as a set of triples (source, target, label$_i$). This assumption is justified since the edges of any graph G can be efficiently partitioned into edge-maximal subgraphs G_k, each of minimum degree at least k and average degree not more than $2k$ [4]. Furthermore, we assume that for each subgraph in the sequence $(G_k = (V_k, E_k))_{k=0}^{z}$ the Wave and Edge Fragment decompositions from [3] has already been computed. This decomposition is stored in three parts: triples of (source,target,unique fragment id), a mapping from unique fragment ids to wave numbers, and a mapping from wave numbers to edge label$_i$s. The vertex set V_k of each such subgraph G_k can be partitioned into ordered sets $V_{k,j}$ that correspond precisely to Minimum Degree Waves(S_0, P) (see Section 3.3), where S_0 consists of the vertices of minimum degree, and the property P corresponds to boundary vertices of degree less than the degree of vertices in S_0 [3].

A *Graph City* is a 3D representation of a given maximal edge graph partition, as shown in Figure 3. It consists of a floor plan of *buildings* (one per equivalence class), *bushes* that represent clusters of small buildings, and a weighted *street network*.

4.1 Graph City Buildings

For each edge-maximal subgraph G_k, we use the ordered sequence of vertex subsets $(V_{k,j})_{j=0}^{h}$, their "internal" edges, and those edges running between consecutive levels to create a visual representation for each such fixed point G_k, that resembles a building in a city with h floors (see Figure 2).

This building representation provides an alternative view of a fixed point $G_k = (V_k, E_k)$ that can be computed in time and space linearly dependent on the size of G_k, plus the complexity of finding and/or "describing" the initial subset $V_{k,0}$ of V_k. That is, the macro-structure of any fixed point can be described as a "building" whose internal structure is a Meta-DAG (see Section

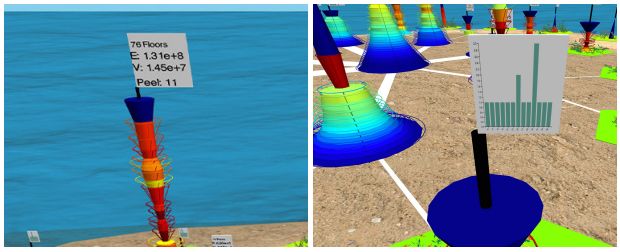

Figure 6: A flag on a building from the (left) Friendster City, and (right) from the patent citation network city.

3.3.2), with macro-vertices representing connected components of seed sets within each building floor (Figure 5).

A graph building with h floors representing a fixed point $G_k = (V_k, E_k)$ is completely determined by the "Disjoint Union of ordered Edge Fragments" specified by the partition of V_k into level sets $(V_{k,j})_{j=0}^{h}$. Our representation requires only $5h$ numbers. For each wave we specify 2 disk radii, the starting height, the color of a frustum, and a light intensity for night view.

The "j-th floor" of a building representing G_k corresponds to the subgraph induced by a subset of vertices $V_{k,j}$. Each floor is represented by two concentric disks, one above the other. The radius of the bottom disk encodes the number of vertices in the seed set of the starting fragment of the corresponding floor. The radius of the top disk encodes the total number of vertices besides the seed set vertices. The top disk is placed at the same height as the bottom disk of the next floor. A *frustum* between the bottom disks of adjacent floors is set to have volume encoding the number of edges running from one floor to the next. Since the radii of the two disks is already determined, the height of the frustum is calculated from the desired volume. For the last floor the height of the top disk is determined from the volume corresponding to the total number of edges on the floor.

A special fixed color map across the entire graph is used to encode the density of a variety of induced subgraphs. For example, the color of a frustum represents the density of the set of edges running between the corresponding two floors, and the same color map is used to highlight the density of the connected components within a floor, as shown in Figure 2.

A flag on top of a building displays summary information that includes the distribution of fixed point values of that bucket. Recall that "fixed point" refers to the edge-maximal subgraph G_k. For buildings that are alone in a bucket the flag shows the number of floors, the peel value, and the number of vertices and edges of the corresponding fixed point. Figure 6 shows both these cases. The height of the flag cloth encodes the total number of edges in G_k, and its width encodes the total number of vertices. The length of the flag pole is the overall edge density of G_k. If the user toggles the *night view* mode, a checkerboard of lights is applied to the frustum between all floors, as shown in Figure 9(right). The light intensity at each floor is proportional to the number of vertices shared between the set of edges represented by that floor and its complement with respect to the whole graph.

Handling buildings with lots of fragments does not present an issue because an intermediate structural level of granularity between Fixed Point Buildings and Fragments is provided by Waves (Section 3.3). The floors of a building represent contiguous segments of Fragments that satisfy some initial condition. They are characterized by their source layer of vertices and an ending layer of vertices whose unexplored neighbors violate a prespecified expansion condition. The beginning and end fragments of Minimum Degree Waves specifically satisfy a bounded-degree condition, i.e., the number of connections to non-Wave vertices is larger than the degree of vertices in the Wave source set.

4.2 Summary Graph City Sculpture

To provide an overview of the size distribution of fixed points in the set of buildings in a graph city we use a summary sculpture (see Figure 7). The aspect ratio of the sculpture encodes the average gap between consecutive fixed point values from the iterative edge core decomposition. The taller it is, the larger the average gap. This sculpture is obtained by considering each building as its set of connected components. All these edge maximal connected subgraphs with the same peel value and the same size are represented by cylinders encoding their frequency. Each cylinder of a particular peel value appears at a unique height in the sculpture. Larger cylinder radii correspond to higher frequency of a particular size. All the disks representing connected components with the same peel value (i.e., associated with the same building) are stacked vertically on top of each other sorted by size. The fixed point value is encoded by rainbow coloring the corresponding cylinders (increasing from blue to red). Our interface provides access to the location of all buildings in the graph city that correspond to a fixed point selected in the sculpture.

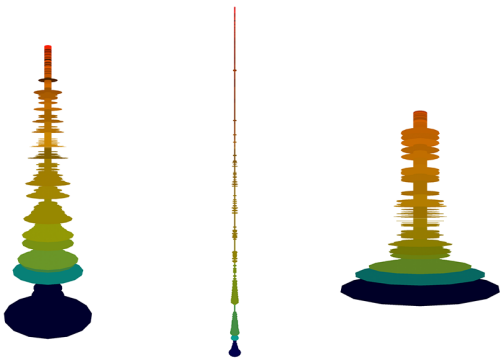

Figure 7: Graph City Sculptures for (left) the Friendster network, (middle) the movie co-occurence phrase network, and (right) the patent citation network. Note that the largest data set need not have the tallest city sculpture.

4.2.1 Vertex Diversity and Light Intensities. Graph cities can be seen as 3D representations of a coloring of the edges, where colors encode the edge peel values. This coloring partitions the edges adjacent to any particular vertex by their assigned color. The frequencies of these local colors for a vertex defines a profile vector for that vertex. Following [4] we compute this profile vector's Shannon Entropy and use it as a measure of the "diversity" of the color pattern of the local edge coloring around each vertex. Higher "diversity" of a vertex is an indicator of a higher weighted level of participation of that vertex in a given edge partition. It is worth noting that diversity is a more expressive measure than degree. Specifically, very high degree vertices can have very low diversity. We add "diversity" light intensities to the disks in the City Sculpture to encode the average diversity of the vertices in the corresponding connected fixed point.

4.3 Graph City Interpretation

"Graph Cities" provide visual representations of the overall macrostructure of graphs with few billion edges, i.e., GigaGraphs, which are derived by mapping each *connected equivalence class*, of a special edge partition, into a "city building". Below we address some common questions regarding their interpretation:

(1) **What do "floors" tell us about a "building" in a Graph City?** The number h of floors in a building (i.e. the number of waves) indicates a fixed point whose full exploration requires the sequential activation of h disjoint seed sets. In cases where a "building" is used to represent an edge equivalence class with several connected components then the number of floors in the "building" corresponds to the maximum number of waves in any of its components.
(2) **What does a "building" volume represent?** It encodes the number of edges of the represented edge equivalence class, i.e., a fixed point of degree peeling. A building with no enclosure represents a more localized topology, i.e., is a "tree like" fixed point with only consecutive edges.
(3) **How is the internal detailed structure of a "building" made accessible for user exploration?** It is represented by a **D**irected **A**cyclic Macro **G**raph obtained by contracting the connected components of each wave seed set. This DAG represents the connectivity between the connected components of all seed sets appearing in the waves. Our interface provides on-demand access to this DAG internal structure for user navigation and exploration on a per building basis.

A video illustrating our current interface can be found here[1].

5 GRAPH CITIES STREET NETWORK

5.1 Graph City Layout and Street Network

The size distribution of the equivalence classes is used to generate a 2D position for each building. This is done by bucketing the fixed points of a graph by size and then mapping each such bucket to a 2D location by following an Archimedean spiral. We create buckets containing connected fixed points the same way as [3]. These connected fixed points are grouped together into buckets according to the number of edges. Bucket i has fixed points of size s, such that $log^{i-1}(m) < s \leq log^{i}(m)$. We create a building for the largest fixed point in each bucket B, and if $|B| > 1$, we create "bushes" (see Section 5.1.1) for a representative selection of $log(8 * (|B| - 1))$ fixed points from B. Additionally, for such buckets we also draw a *grass patch* as a green polygon with $log(8 * (|B| - 1)) + 2$ sides (see Figure 2(right)). From each bucket, we display the tallest building and a flag with a histogram with peel values on the X-axis and the number of buildings with that peel value and the maximum size of all these buildings. For buckets with one fixed point the flag just shows the peel value.

The Graph City *street network* is determined by the Delaunay triangulation of the building locations in the spiral layout (see Fig. 3). The weight of a connection between two buildings is proportional to the intersection of the subgraph vertex sets represented by the two buildings. Graph-theoretically, the street network is determined by the intersection graph of the collection of vertex sets of the subgraphs $(G_k = (V_k, E_k))_{k=0}^{z}$, which in turn are determined by the given edge partition $(E_0, E_1, ..., E_z)$. When a particular building is selected by the user, a Euclidean spanning tree rooted at that building displays the corresponding street network, which is obtained by a Breadth First Search. If the building street network is disconnected, then we show a spanning forest instead. Note that the connectivity of the street network only depends on the connectivity of the subgraphs represented by the buildings and not the graphs represented by the entire bucket.

5.1.1 Bushes and L-Systems. To provide a visual indication for the properties of fixed points in a bucket (besides the largest one represented by the building) we sort the fixed points in a bucket by size and uniformly select a few fixed points to draw as bushes. These bushes visually show some of the same properties as a building but are a much "rougher" view of a fixed point.

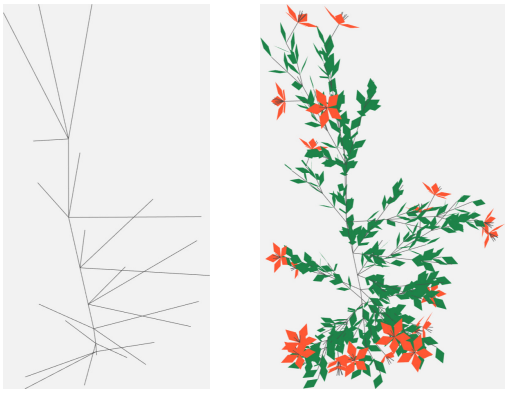

Figure 8: (Left) The bush skeleton for a fixed point of peel value 1 from the Friendster network [21]. (Right) The complete bush after applying 3 iterations of an L-system.

For generating a bush, a skeleton is first created based on the parameters of a given fixed point (see Figure 8(left)). Then, a few iterations of an L-System are executed with axiom points distributed along the skeleton drawing (see Figure 8(right)), to create natural-looking bushes. The skeleton consists of a central stem with branch segments coming out at points in between. These junction points are spaced the same as the disks in the building representation of the fixed point; thus, the length of stem segments correspond to the height of the respective frustums. The inclination of the stem segments ranges from 0 to 45 degrees off the vertical axis based on the density of edges in the frustum (0 degrees corresponding to 0 density and 45 degrees to a density of 1). Two sets of branches are produced at each stem junction corresponding to the inner and outer disks in the building. The number and length of branches encodes the number of vertices represented at that level. The inclination of the branches is proportional to the density of internal edges for that level. All branches are equally spaced at the junction and phase angles at successive junctions are randomized.

We extended an implementation of a turtle graphics based L-system interpreter[2] to draw the underlying skeleton structure and then applied 2-3 iterations of a natural looking L-system, starting at evenly spaced lengths along the stem and branches. The bushes and the green polygon together give the appearance of a "garden" for some of the buildings in the Graph City, that are primarily centered around the periphery of the spiral layout on the ground, as shown in Figures 3 and 2(right). Although these bushes don't reflect as much detail of the underlying fixed point they represent as opposed to the building metaphor, they are useful for quickly getting a sense of scale for the size and distribution of other fixed points in a bucket while being more efficient to render than a whole building.

6 DESCRIPTION OF OUR DATA SETS

We show examples of our system applied to 3 datasets. Our largest data set is the Friendster social network consisting of 65,608,366 nodes each representing a user and 1,806,067,135 edges representing "friendships" between them. This dataset was retrieved from the Stanford Large Dataset Collection (SNAP) [21]. Our next data set is a graph of phrases used in movie reviews. There are 218,052 nodes each representing a phrase and 115,050,370 edges, where

[1] https://rutgers.box.com/s/qeygblwr5udeti9vxcmr0nj6swuf179n

[2] https://github.com/andonutts/donatello

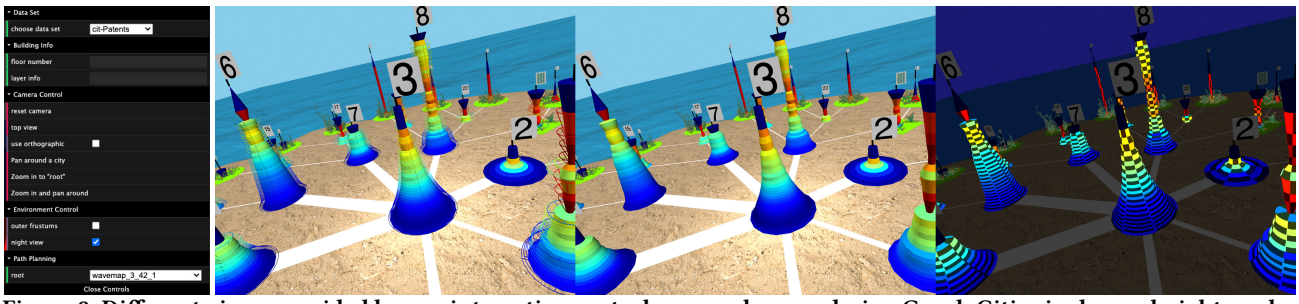

Figure 9: Different views provided by our interactive control menu when rendering Graph Cities in day and night modes.

each edge connects two phrases both used to describe the same movie in a review. This data set was derived from the Internet Movie Database [3]. Our last data set is a patents citation network, also from [21]. There are 3,774,768 nodes each representing a patent and 16,518,947 edges each linked to a cited patent.

Table 1 shows the number of connected components (CC), connected fixed points (FP), peel value of the core (CV), maximum number of waves among its fixed points (MW), and maximum number of fragments among its fixed points (MF) for each data set. The spiral length is related to the total number of graph edges. The number of fragments in a building, i.e., the building's height, encodes the longest path length in the building's Meta-DAG.

Dataset	CC	FP	CV	MW	MF	RT
Friendster	1	29,692	304	212	3,279	11.82
Movies	38	2,044	3114	37	282	3
Patents	3,627	6,469	64	47	996	3.3

Table 1: Statistics for all data sets. The last column shows the rendering times (RT) for the graph city in seconds.

7 RENDERING GRAPH CITIES

We use *Three.js* [6], a 3D Javascript library to create and display Graph Cities interactively in the web browser using WebGL. Each building in the Graph City consists of several floors (i.e., Edge Fragments). For each floor, we instantiate a *cylinder* shape geometry, where the top and bottom face radii, height, and color are chosen appropriately from the data (see Section 4.1), the number of balcony segments defaults to 6, with 3 windows per floor in the night view. All floors are generated in the material space, centered at the origin, and subsequently translated in the Y (up) direction to form a building in the world space. Each building is also translated in the X and Z directions to form the spiral layout (see Section 5.1). *Bushes* are generated with an L-system.

On top of each building, a *flag* displaying summary information for that building (i.e., edge-maximal subgraph) is added. A *box* shape geometry is used for the flag, and a *cylinder* shape geometry is used for the mast. The length of the mast, size of the flag, and color of the flag are all determined from the data set.

To highlight the *street network* representing data flow from one building to the remaining Graph City, we precompute the Delaunay triangulation for all the buildings in the Graph City at the beginning. Subsequently, when a user selects a particular building from the interactive control menu (see Figure 9(left)), we perform a Breadth First Search (BFS) from the root building to the rest, and display only those edges that are encountered in the search. The width of the edges are determined by the data. Rendering times for the Graph City for each data set are summarized in the last column in Table 1.

[3]https://www.imdb.com/interfaces/

8 NAVIGATING A GRAPH CITY

The interactivity provided by *Three.js* [6] allows the user to explore different parts of a Graph City conveniently from a browser. An interactive control menu is provided on the top right (see Figure 9(left)) for toggling between various options, such as the current data set being viewed, displaying building information, changing camera/environment controls and day/night modes, or selecting different street networks. The user can use the mouse to zoom in and out, as well as pan around the Graph City. Upon entering a particular building, the view changes to highlight the finer scale internal connectivity for each edge-maximal subgraph (or fixed point) that constitutes a building, as shown in Figure 10.

Our interactive visualization ultimately owes its speed to the hierarchical Graph Wave decomposition [3]. At the macroscopic level, our abstraction has a much smaller spatial complexity than the actual data set, which makes rendering cheap. The only computation that happens is whenever the root for the street network changes, but this is only linear in the number of buildings. When the user views inside a particular building, although we show finer scale connectivity, it is still conveniently partitioned into Edge Fragments. Thus, rendering is never the bottleneck, as the user is only visualizing a *subset* of the data.

9 CATALOGUE OF SUBGRAPH PATTERNS

A useful outcome of user or computer explorations of any graph city will be a summary and an extensive catalogue of the subgraph patterns found. For this to be feasible, users must be provided with annotation and summarization tools that can keep track of their exploration trails. These patterns should be classified at least by size, density, frequency of occurrence, rarity, interest and usefulness. To assess the efficacy of these tools we are currently identifying a list of basic tasks that a user can perform in order to conduct a substantial number of user experiments. For example: can graph cities be used effectively on shortest path approximations queries?

One of the ultimate goals of this work will be to have a descriptive semantic summary of the expected patterns that can be fed into a deep learning engine to learn to discriminate and find new patterns according to certain specified criteria. In our current experimentation with a variety of data sets, the most naturally detected patterns include: tree forests, cliques, bi-cliques, and hierarchical compositions of these basic patterns.

10 FUTURE WORK

Streaming Graph Cities: An interesting question is the feasibility of efficiently updating the Wave and Fragment decomposition when the streaming input graph is known a priori to be a fixed point or a Wave of fixed degree peeling. This is the main bottleneck for streaming a Graph City. If the wave decomposition

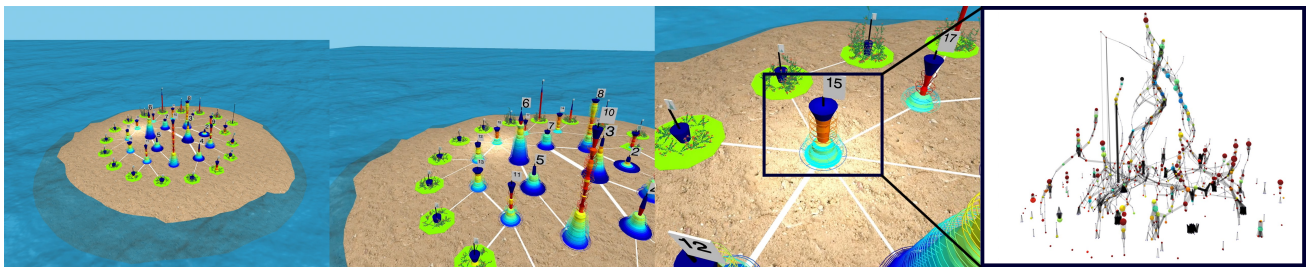

Figure 10: Snapshots of a user navigating the Graph City for the patents citation network with interactive camera control.

is provided then we can easily obtain web renderings of the corresponding individual building(s) at interactive frame rates.

Composing Global Solutions from Local Ones: The Wave and Edge Fragments decomposition will be more advantageous in those cases where a given question or structure of interest on $G = (V, E)$ can be obtained as a "composition" of the local Wave and/or Fragment solutions. For example, is there any algebraic relation between the all-pairs shortest path distances in G versus their restrictions to the Waves and/or Fragments of G?

How to pick interesting Waves? Automating the process of choosing the property of interest for the Wave decomposition would be a powerful addition to the pipeline. Can the type of wave be chosen efficiently based on local or global graph structure?

Semi-random Graph Processes: It will be interesting to understand the type of Graph Waves that can be built by these semi-random processes with high probability in $O(n)$ rounds where n is the number of vertices [11].

Hypergraph Cities: A central problem in simplicial finite set systems is to identify a global structure whose intersection with a given finite set family is highly concentrated around its expectation. Random matchings are an example [14].

11 CONCLUSIONS

The iterative edge decomposition partitions the edges of a graph into Fixed Points of degree peeling; they are in turn decomposed into Graph Waves and Edge Fragments. They provide mechanisms that may help assess the topological and statistical reasons that explain the emergence of a large class of bipartite graph-like patterns in very large graph data sets.

We introduced 3D representations of Fixed Points based on their wave decomposition that resemble buildings in a city, hence Graph Cities. A spiral arrangement of the buildings is obtained from the size distribution of the Fixed Points. The Delaunay triangulation of the building locations determines the graph city street network. The size distribution of all the Fixed Points is summarized by a graph city sculpture (Sec 4.2).

Dense bipartite graph-like patterns have been proposed as an abstract formalization of "concepts" in [29]. Their identification in very large data sets has defied computation. Nevertheless, Graph Cities offer a promising approach to the efficient detection of a large sub-class of these patterns in attributed graphs.

Due to lack of space, some technical results were put in an appendix here[4].

REFERENCES

[1] J. Abello, A. L. Buchsbaum, and J. R. Westbrook. 1998. A functional approach to external graph algorithms. In *European Symp. on Alg.* Springer, 332–343.

[2] James Abello, Daniel Mawhirter, and Kevin Sun. 2019. Taming a Graph Hairball: Local Exploration in a Global Context. In *Business and Consumer Analytics: New Ideas.* Springer, 467–490.

[3] James Abello and Daniel Nakhimovich. 2020. Graph Waves. In *The 3rd International Workshop on Big Data Visual Exploration and Analytics with EDBT/ICDT.*

[4] J. Abello and F. Queyroi. 2013. Fixed points of graph peeling. In *Proc. of the 2013 IEEE/ACM Int. Conf. on Adv. in Soc. Net. Anal. and Mining.* 256–263.

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

[6] Ed Angel and Eric Haines. 2017. An Interactive Introduction to WEBGL and Three.JS. In *ACM SIGGRAPH 2017 Courses.* Article 17, 95 pages.

[7] A. Arleo, O.-H. Kwon, and K.-L. Ma. 2017. GraphRay: Distributed pathfinder network scaling. In *2017 IEEE 7th Symp. on Large Data Anal. and Vis.* 74–83.

[8] B. Bach, N. H. Riche, C. Hurter, K. Marriott, and T. Dwyer. 2017. Towards unambiguous edge bundling: Investigating confluent drawings for network visualization. *IEEE Trans. on Visualization & Computer Graphics* (2017), 1–1.

[9] V. Batagelj and M. Zaveršnik. 2011. Fast algorithms for determining core groups in social networks. *Adv. in Data Anal. and Class.* 5, 2 (2011), 129–145.

[10] M. Baur, M. Gaertler, R. Görke, M. Krug, and D. Wagner. 2007. Generating graphs with predefined k-core structure. In *Eur. Conf. of Compl. Sys.* Citeseer.

[11] O. Ben-Eliezer, L. Gishboliner, D. Hefetz, and M. Krivelevich. 2020. Very fast construction of bounded-degree spanning graphs via the semi-random graph process. In *Proc. of the Symposium on Discrete Algorithms.* SIAM, 718–737.

[12] Daniel Berend, Shlomi Dolev, and Ariel Hanemann. 2014. Graph Degree Sequence Solely Determines the Expected Hopfield Network Pattern Stability. *Neural computation* 27 (11 2014), 1–9.

[13] 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.

[14] Peter Frankl and Andrey Kupavskii. 2018. The Erdös Matching Conjecture and concentration inequalities. *arXiv preprint arXiv:1806.08855* (2018).

[15] P. Govindan, C. Wang, C. Xu, H. Duan, and S. Soundarajan. 2017. The k-peak decomposition: Mapping the global structure of graphs. In *Proceedings of the 26th International Conference on World Wide Web.* 1441–1450.

[16] W. L. Hamilton, R. Ying, and J. Leskovec. 2017. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin* (2017).

[17] R. Jin, N. Ruan, S. Dey, and J. Y. Xu. 2012. SCARAB: scaling reachability computation on large graphs. In *Proc. of Int. Conf. on Man. of Data.* 169–180.

[18] H. Kabir and K. Madduri. 2017. Shared-memory graph truss decomposition. In *2017 IEEE 24th Int. Conf. on High Perf. Comput. (HiPC).* IEEE, 13–22.

[19] M. Krommyda, V. Kantere, and Y. Vassiliou. 2019. IVLG: Interactive Visualization of Large Graphs. *Int. Conf. on Data Engineering* (2019), 1984–1987.

[20] R. Laishram, A. Erdem Sar, T. Eliassi-Rad, A. Pinar, and S. Soundarajan. 2020. Residual Core Maximization: An Efficient Algorithm for Maximizing the Size of the k-Core. In *Proc. of Int. Conf. on Data Mining.* SIAM, 325–333.

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

[22] 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.*

[23] 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.

[24] Q. H. Nguyen, S.-H. Hong, P. Eades, and A. Meidiana. 2017. Proxy graph: visual quality metrics of big graph sampling. *IEEE TVCG* 23, 6 (2017), 1600–1611.

[25] R. Pienta, F. Hohman, A. Endert, A. Tamersoy, K. Roundy, C. Gates, S. Navathe, and D. H. Chau. 2018. VIGOR: interactive visual exploration of graph query results. *IEEE Trans. on Vis. and Comp. Graph.* 24, 1 (2018), 215–225.

[26] L. Royer, M. Reimann, B. Andreopoulos, and M. Schroeder. 2008. Unraveling protein networks with power graph analysis. *PLoS comp. bio.* 4, 7 (2008).

[27] Siddhartha Sahu, Amine Mhedhbi, Semih Salihoglu, Jimmy Lin, and M Tamer Özsu. 2017. The ubiquity of large graphs and surprising challenges of graph processing. *Proceedings of the VLDB Endowment* 11, 4 (2017).

[28] N. Wang, D. Yu, H. Jin, C. Qian, X. Xie, and Q.-S. Hua. 2017. Parallel algorithm for core maintenance in dynamic graphs. In *Prof. of ICDCS.* 2366–2371.

[29] Rudolf Wille. 1992. Concept lattices and conceptual knowledge systems. *Computers & mathematics with applications* 23, 6-9 (1992), 493–515.

[30] H. Wu, J. Cheng, Y. Lu, Y. Ke, Y. Huang, D. Yan, and H. Wu. 2015. Core decomposition in large temporal graphs. In *Int. Conf. on Big Data.* 649–658.

[31] V. Yoghourdjian, T. Dwyer, K. Klein, K. Marriott, and M. Wybrow. 2018. Graph thumbnails: Identifying and comparing multiple graphs at a glance. *IEEE Transactions on Visualization and Computer Graphics* 24, 12 (2018), 3081–3095.

[32] H. Zhou, P. Xu, X. Yuan, and H. Qu. 2013. Edge bundling in information visualization. *Tsinghua Science and Technology* 18, 2 (2013), 145–156.

[4] https://rutgers.box.com/s/qeygblwr5udeti9vxcmr0nj6swuf179n