Atlas: Local Graph Exploration in a Global Context

James Abello*
Rutgers University
New Brunswick, NJ, USA
abelloj@cs.rutgers.edu

Fred Hohman*†
Georgia Tech
Atlanta, GA, USA
fredhohman@gatech.edu

Varun Bezzam
Georgia Tech
Atlanta, GA, USA
varun.bezzam@gatech.edu

Duen Horng Chau
Georgia Tech
Atlanta, GA, USA
polo@gatech.edu

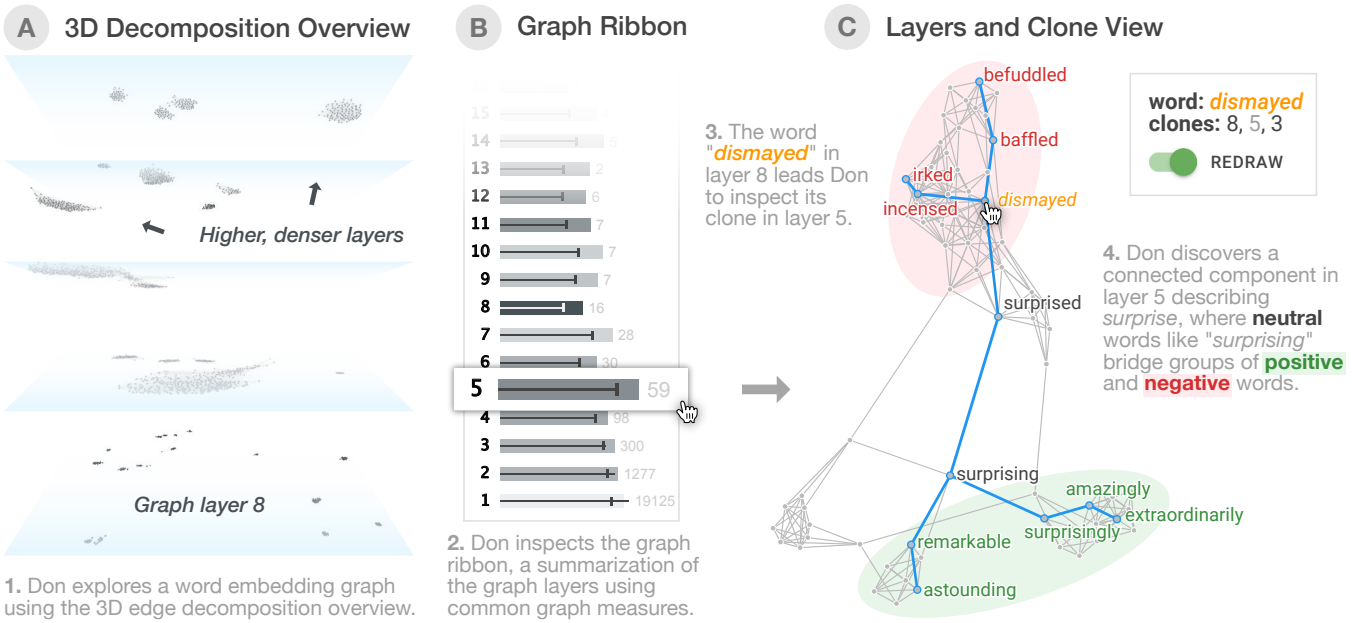

Figure 1: Atlas adapts scalable edge decomposition to provide novel modes of large graph exploration, through three coordinated views. A. Our user Don first explores the edge decomposition of a word embedding graph in the *Overview* by decomposing a graph into 3D graph layers. **B.** Don then inspects the *Ribbon* for a summary of the layers. **C.** From the word "**dismayed**," in layer 8, Don performs *cross-layer exploration*, to reach layer 5. Using the *Layers* view's interactive node-link diagrams, Don discovers a component in the word embedding describing one's *surprise*, where neutral words (e.g., "surpised" and "surprising") bridge multiple quasi-cliques that describe more positive (e.g., "remarkable" and "astounding") and negative (e.g., "irked" and "incensed") *surprise* words. Blue perspective planes, and red and green ellipses are illustrative annotations.

ABSTRACT

Graphs are everywhere, growing increasingly complex, and still lack scalable, interactive tools to support sensemaking. To address this problem, we present Atlas, an interactive graph exploration system that adapts scalable edge decomposition to enable a new paradigm for large graph exploration, generating explorable multi-layered representations. Atlas simultaneously reveals peculiar subgraph structures, (e.g., quasi-cliques) and possible vertex roles in connecting such subgraph patterns. Atlas decomposes million-edge graphs in seconds, scaling to graphs with up to 117 million edges. We present the results from a think-aloud user study with three graph experts and highlight discoveries made possible by Atlas when applied to graphs from multiple domains, including suspicious yelp reviews, insider trading, and word embeddings. Atlas runs in-browser and is open-sourced.

*Authors contributed equally.
†Corresponding author.

IUI '19, March 17–20, 2019, Marina del Rey, CA, USA

ACM ISBN 978-1-4503-6272-6/19/03…$15.00
https://doi.org/10.1145/3301275.3302275

CCS CONCEPTS

• **Human-centered computing** → **Visualization systems and tools**; *Graph drawings*; • **Mathematics of computing** → **Graph algorithms**.

KEYWORDS

Interactive graph exploration, graph sensemaking, graph visualization, edge decomposition

ACM Reference Format:
James Abello, Fred Hohman, Varun Bezzam, and Duen Horng Chau. 2019. Atlas: Local Graph Exploration in a Global Context. In *24th International Conference on Intelligent User Interfaces (IUI '19), March 17–20, 2019, Marina del Rey, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3301275.3302275

1 INTRODUCTION

Graphs are everywhere, growing increasingly complex, and still lack scalable, interactive tools to support sensemaking. In a recent online survey, graph analysts rated scalability and visualization as the most pressing issues to address [57]. Graph drawing approaches such as "super-noding" [4, 10, 11], and edge bundling [5, 24, 36] have been designed to help visually reduce the number of glyphs visible to a user. Some work abstracts graphs to higher-level representations, such as using contours and heat maps as a proxy for vertex density [22, 44], graph motifs for repeating structural patterns [27], and overall graph summarizations [42, 68]. New modes of exploration based on relevance and measures of "interestingness" have also been developed to explore large graphs without showing every vertex and edge [23, 33, 55]. While these approaches may help users develop insights into a graph's functional properties, scalability, interaction, and extracting overall descriptive information about an unknown graph as it is being explored remain pressing issues in large graph exploration systems.

Edge decomposition algorithms, based on fixed-points of degree peeling, have strong potential in helping users explore unfamiliar graph data [1, 3], because (1) they can discover peculiar subgraph patterns structurally similar or dissimilar to regular subgraphs; (2) they can quantify possible "roles" a vertex can play in the overall network topology; and (3) they scale to large graphs.

In this work, we present Atlas (Figure 1), an interactive graph exploration system that adapts scalable edge decomposition [3] to enable a new paradigm for large graph exploration, generating explorable multi-layered representations. Through Atlas, we contribute:

- **New paradigm for graph exploration.** Our novel approach introduces two new actionable concepts, **graph layers** and **vertex clones**, that help analysts discover interesting and unusual graph substructures. It decomposes a large graph into an ordered set of **graph layers** (see Figure 1A), such that each edge participates in a unique layer. Vertices, however, can exist in multiple layers; we call these **vertex clones**. Graph layers help users identify potentially interesting and unusual substructures, by extricating such patterns from the whole graph. In the graph layer set, layers with denser structures rise to the top (e.g., quasi-cliques, multi-partite-cores), while those with sparser structures (e.g., trees, stars) sink to the bottom. Vertex clones allow one to perform **cross-layer exploration**, investigating a graph across layers, to examine local structures with a global context.
- **Fast, scalable edge decomposition via memory mapping and multicore parallelization.** Atlas decomposes million-edge graphs in seconds, scaling to graphs with up to 117 million edges from many domains (e.g., social networks, hyperlink networks, and co-occurrence networks). To the best of our knowledge, this is the first published timing results of this edge decomposition algorithm. Our algorithm is open-sourced.[1]
- **Atlas: a web-based interactive graph exploration system.** Atlas is open-sourced.[1] It offers three coordinated views for exploring large graphs. It accelerates graph rendering and layout using GPUs (e.g., via 3D graphics library three.js), supporting real-time visualization of graph layer overview and interactive cross-layer exploration of local subgraph structures.
- **User Study.** Through a think-aloud user study with three graph experts, we highlight discoveries made possible by Atlas, such as spotting suspicious connections among yelp reviewers and insider traders.

2 ILLUSTRATIVE SCENARIO

To illustrate how Atlas can help users explore large graphs and discover interesting structure, consider a user Don who wants to make sense of a word embedding graph generated from Wikipedia from 2014. Creating word embeddings is a popular and powerful technique to turn words into high dimensional vectors [18, 48, 51], which are then fed as input to machine learning algorithms to solve a variety of problems, e.g., visual question answering [9], neural machine translation [13]. Therefore it is important to make sense of what information a word embedding has captured and how well the embedding matches our understanding of language.

Don's Wikipedia word embedding graph is generated using GloVe: an unsupervised learning algorithm for obtaining vector representations for words [51]. The graph contains 65,870 vertices and 213,526 edges. Each vertex is a unique word, and an edge connects two words if the angular distance between their two word vectors is less than some threshold.[2]

Visualizing edge decompositions. Exploring the word embedding for the first time, Don wants to first see an overview of the graph. Atlas decomposes the word embedding graph into *graph layers*, and visualizes them as a 3D structure in the Overview (Figure 1A), one of three main views in Atlas. Layers with denser structures rise to the top of the 3D structure (e.g., quasi-cliques), while those with sparser structures (e.g., trees, stars) sink to lower layers.

Finding interesting graph layers. Atlas's Ribbon (see Figure 1B) provides Don with a compact visual summary of the edge decomposition using well-studied graph measures (e.g., #vertices, clustering coeff.). Each graph layer is encoded by a glyph. Seeing that a layer's clustering coefficient is encoded by color brightness (a darker bar represents a denser layer), layer 8 caught Don's attention, because it is very dense (i.e., very dark bar), yet it is further down in the Ribbon than other dense layers. Don clicks layer 8 in the Ribbon to display it in the Layers view. The layer contains many small connected components (Figure 2, left), whose node positions have been

[1] Visualization: https://github.com/fredhohman/atlas.
Algorithm: https://github.com/fredhohman/atlas-algorithm.
[2] Angular distance is closely related to cosine similarity, and is an effective method for measuring the linguistic or semantic similarity of corresponding words [51]. The threshold to connect two words is set to 0.9. Words with numbers/digits are removed from the dataset.

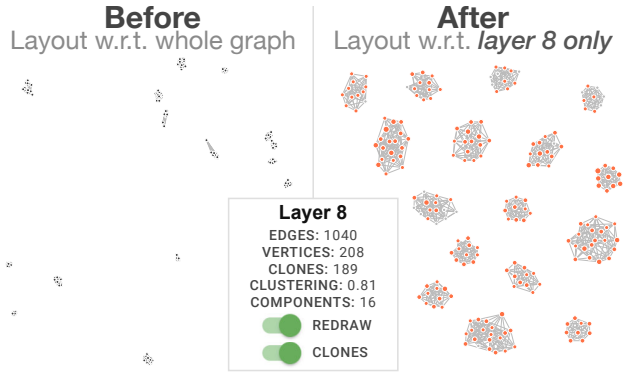

Figure 2: Layer 8 from the word embedding graph. On the left is the original layout computed with respect to the entire graph, but now that we have separated out layer 8 from the remaining graph, we can recompute its layout independently. This produces the layout on the right, where the cloned vertices are colored red and sized according to how many clones they have in the remainder of the graph.

computed with respect to the entire graph. Since we are now visualizing only a single layer, Atlas performs a force-directed layout for these components, with respect to this layer only, revealing that layer 8 is in fact a collection of highly dense quasi-cliques (Figure 2, right). This view is fully interactive: Don can zoom and pan over the graph layer, brush a vertex to see its label, and highlight its immediate neighbors.

Cross-layer exploration. While exploring layer 8, Don discovers many interesting quasi-cliques of related words, such as one describing familial relationships (with words like "daughter," "husband," and "grandparent") and another that describes the levels of negative emotion one can experience (including words like "annoyed," "dismayed," and "mortified"). As a word can have multiple meanings, Don wonders if the word "dismayed" also participates in other layers. Atlas supports interactive cross-layer exploration. Don clicks on the "Clone" toggle, which colors and sizes vertices that are cloned in other layers red (Figure 2, right), revealing that many vertices in the quasi-clique also exist in other layers. Don inspects the vertex "**dismayed**," and notices that it has vertex clones in layers 5 and 3 (Figure 1C, top right). Clicking layer 5 brings Don to that layer for further exploration.

Local exploration with a global context. Unlike in layer 8, "dismayed" in layer 5 is connected to a larger component (Figure 1C), and as Don explores the neighborhood of "dismayed," he notices the words transition to more neutral words like "surprised" and "surprising," then to positive words like "remarkable," "astounding," and "extraordinarily." Using Atlas, Don has now discovered that words describing *surprise* are represented in this word embedding similar to how humans would think of them: one can be *surprised* in both positive or negative ways, thus "bridging" quasi-cliques of positive and negative *surprise* words (Figure 1C). To help Don keep track of and summarize his exploration, Atlas automatically computes the shortest paths that connect vertices (words) that he has inspected and highlight the paths in blue. We call this interactive visual summary the shortest-path-net, which allows a user to explore semantic information and information transition within a connected component of a graph layer.

Multiple exploration choices. Don now has multiple choices for continuing exploring this word embedding graph using Atlas: (1) visit the other connected components in layer 5, (2) backtrack to layer 8 and use the last clone of "dismayed" as a mechanism to perform further cross-layer exploration, or (3) return to the beginning and inspect the Overview and Ribbon for a completely different layer to explore. Regardless of what Don chooses, he can gain a better understanding of the word embedding graph both globally, by visualizing graph layer structure, and locally, by using vertex clones and shortest-path-net representations.

3 RELATED WORKS

Visualizing graphs is an active area of research that has motivated the development of many tools and techniques, of which many are surveyed in [35] and more recently [62]. Interacting with graphs for sensemaking is also a popular and successful avenue for research that is surveyed in [52]. In a recent online survey that was conducted to gather information about how graphs are used in practice, researchers uncovered that scalability and visualization are undeniably the most pressing issues faced by graph data analysts [57]. Lastly, a survey from some graph visualization pioneers outlines future research directions for graph drawing, visualizing, mining, and analytics, noting that while graph visualization has its own research trajectory, oftentimes it has significant overlap with the broader field of visual analytics [63].

Analytics to support sensemaking. Typical approaches to visualize graphs include force directed layouts, vertex clusterings, and topological contractions to reduce visual complexity. Advanced analytic techniques like identifying vertex roles and diversity [34], graph motifs [27], and graph summarizations [43, 64, 68] can show deep insight into a network's functional properties, yet scalability and interaction are two pressing issues of central importance in large graph exploration systems. Some work aims to summarize graphs by hiding redundant vertices and edges commonly used in node-link diagrams by using contour maps and heat maps to show the density of vertices in a particular region of the graph [7, 22, 44]. Extracting overall descriptive information about an unknown graph data set as it is being explored is a desirable feature that can amplify users ability to discover "out of the box" data features [61]. There has also been work done on relevance based exploration approaches where a system can recommend particular vertices to explore within large graphs [23, 33, 55]. Lastly, while in this work we present the benefits of using edge decomposition as a means for interactive graph exploration, previous work for vertex decompositions, such as k-core decomposition, has been explored [8, 17, 68].

Visualizing large graphs. Visualizing large graphs has seen research attention for years [2, 12], with more recent work bringing fluid interactions into interfaces for the web [30]. There are now an abundance of open-source graph visualization libraries, some of which [15, 28] have been highly successful in practice; however, most toolkits and libraries do not scale as graphs grow larger. Some

work has tackled this scalability problem head on by utilizing powerful graphics processing units (GPUs) to rendering millions of vertices and edges on screen at once [40, 47, 50]. While this work pushes the boundary of visual scalabiltiy, showing an entire graph, of even modest size, often limits the amount of interaction possible; recent work has furthered underscored the importance of developing interactive visual analytics systems that can handle "extreme" scale data [67] and graphs.

Visual scalability. As an alternative to visualizing an entire graph, visual reduction techniques like multi-layered and hierarchical visualizations of graphs have also been addressed [4, 11]. Here, similar or nearby vertices are grouped together into a "super-node," also called "meta-nodes" [10]. Some work has developed interactive systems to zoom into super-nodes on demand [4, 45, 65] and navigate graphs bottom-up [25] to reduce the amount of vertices shown at one time, while other work has explored the feasibility of using hyperbolic geometry for graph exploration [49]. Similar techniques have been developed to address the visual scalability of edges in graphs. Often referred to as edge bundling, this technique aims to group similar edges together into larger paths to highlight major connections across a graph. Edge bundling has seen great interest both historically [24, 36, 37] and recently [5, 29, 31, 58]. Earlier research has studied visualizing graphs in 2.5D, 3D, and stereoscopic space and incorporated some of the above techniques [6, 17, 20]. However, Atlas introduces a new paradigm for large graph exploration and new actionable concepts (e.g., graph layers, vertex clones) that existing work has not investigated.

Interactive graph querying. Another alternative to visualizing an entire graph is interactive graph querying, where the general approach is to query large graphs for meaningful, user-defined subgraphs in order to glean insight about a larger graph's structure [54, 56]. However, some graph queries may return hundreds of results; therefore, some recent work has addressed summarizing graph query results [53] using dimensionality reduction to produce graph embeddings that can be visualized as interactive 2D scatterplots [53, 64]. Querying allows one to beginning to explore modern day massive graphs, with the potential for billions and even trillions of edges [66]; however, returned queries often lose their global context with respect to the original graph.

4 ATLAS: INTERACTIVE LARGE GRAPH EXPLORATION

Here we describe design challenges and our solutions that guided the design decisions for Atlas. The following three subsections each describe one of the main coordinated views of Atlas and highlight their core features for graph exploration; these include the 3D Overview, the Graph Ribbon, and the Layers view.

4.1 Challenges and Design Rationale

Challenge 1: Variety of overlapping subgraph structure. There are a variety of existing techniques that aim to discover structure and patterns in graphs (as discussed in Related Works). However, while these techniques may find individual structure and patterns, they do not link the findings together, nor do they explain how multiple patterns are associated with one another. Revealing such kinds of links between structure and pattern is a hallmark capability that is crucial to sensemaking [32, 38]. *Our solution:* We utilize the dual nature of graph layers: (1) layers can be explored independently from one another, but more importantly, (2) layers can be linked together using vertex clones as a mechanism of traversal from layer to layer. We call this cross-layer exploration (see Figure 3). Visualizing the decomposition in 3D may help users more clearly see the overlapping graph structure, which could help them choose which layer of the graph to explore first.

Challenge 2: Local exploration of large graphs. Since large graph exploration is difficult from both a visual and computational scalability perspective, querying a graph or considering subgraphs to explore locally can be helpful. However, often the global context is lost using these approaches, as users do not know where in the graph they are exploring, or how different subgraphs are related to one another. *Our solution:* We design a novel visual summarization of the edge decomposition called the Graph Ribbon and embed it in the middle of the user interface (Figure 3). The Ribbon encodes each layer as a glyph and functions as a global map of the decomposition and graph. A small triangle pointing left or right (denoting if the layer is visualized in the 3D Overview or the Layers view) is displayed next to visualized layers' glyphs. We also design novel local exploration techniques within a graph layer (e.g., shortest-path-net via sequential egonet expansion) that help users explore graphs locally with a global context.

Challenge 3: Handling large graphs. While many graphs are small and can be visualized in 2D with standard layouts, many modern graphs are large and complex. Not only is this problematic for data visualization itself, but also troublesome for engineering interactive tools. The sheer size of the data render many existing visualization tools unusable as they are often designed to visualize the entire graph. *Our solution:* We display a visual summarization of the edge decomposition (called the Ribbon) for a high-level view of the graph and its decomposition. Then, we can selectively load and visualize only the layers we desire, skirting scalability challenges that come with visualizing an entire graph at once. We further support Atlas by open-sourcing all of its code, from the decomposition algorithm to the interactive visualization system, ensuring that it is cross-platform and accessible without specialized hardware.

4.2 3D Graph Decomposition Overview

The left view of Atlas, called the Overview (Figure 3), visualizes graph decompositions in 3D and allows users to zoom, pan, and rotate the 3D structure in-browser and in real-time. Since the graph edge set is uniquely partitioned into graph layers, a natural approach to visualize the decomposition is to first perform a traditional 2D layout of the graph in the plane (this assigns vertices x and y coordinates); however, we now assign a z coordinate to each vertex that is a function of the vertex peel value, i.e., the graph layer number. Since graph layers are numerically ordered, when visualizing a decomposition in 3D the highest, most dense layers (e.g., quasi-cliques) rise to the top while the lower layers sink to the bottom (e.g., trees, stars). Computing the initial 2D layout of large

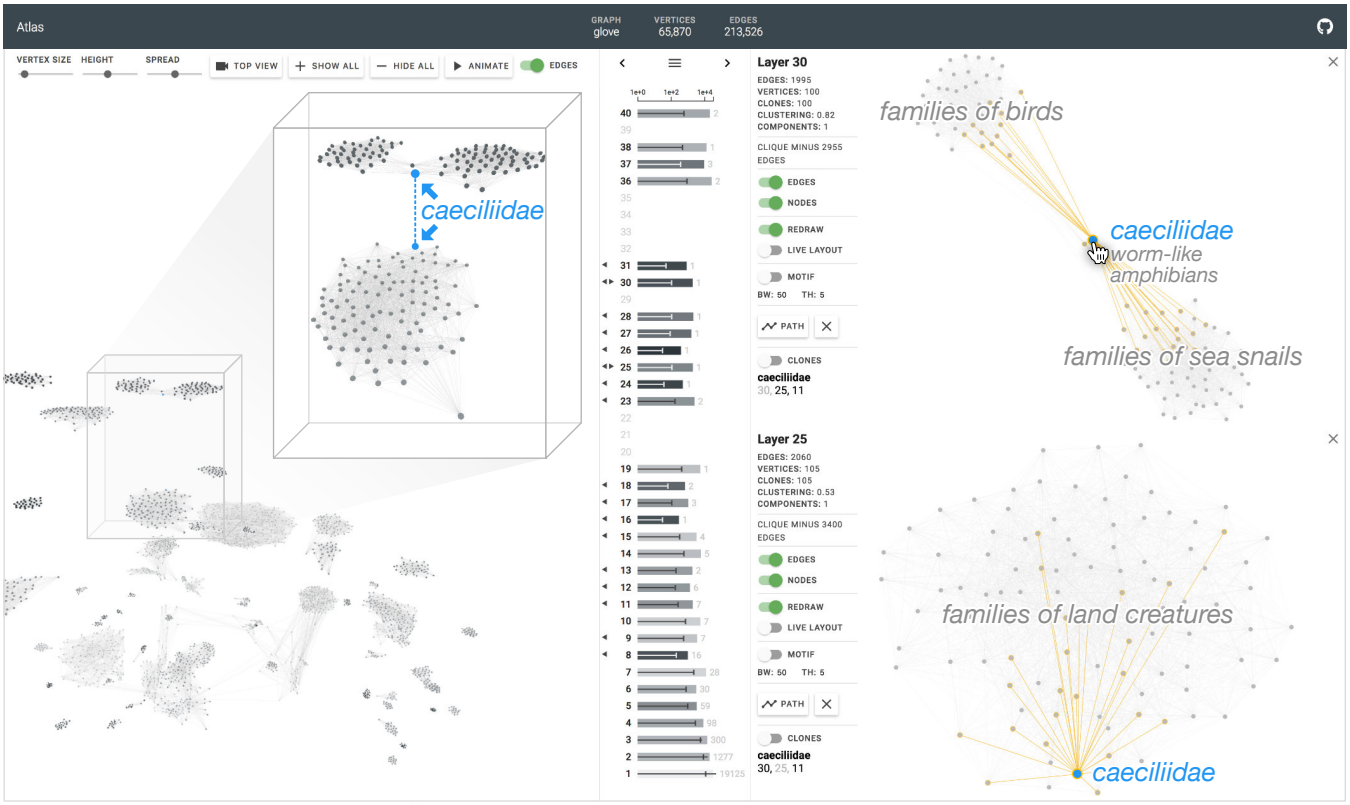

Figure 3: Atlas user interface. Atlas is composed of three main views: the 3D Overview (left), the Ribbon (middle), and the Layers view (right). The Ribbon splits the interface and can be dragged left or right to adjust the visible screen real estate that either the Overview or Layers view shows. Here, the vertex "caeciliidae" is selected, colored blue in both the Overview and Layers view. We see "caeciliidae" (a worm-like amphibian) in layer 30 bridges two quasi-cliques (families of birds and families of sea snails) together, while its clone in layer 25 participates in another single quasi-clique (families of land creatures).

graphs is non-trivial; therefore, we use a GPU-accelerated implementation [21] of the Barnes-Hutt approximation [14] to compute large graph layouts in minutes.

Users can display all graph layers at once or selectively add layers to the Overview. The Overview also contains options to help users explore and manipulate the 3D structure, including: sliders for adjusting the size of the vertices, the height of the layers (e.g., dragging this slider animates splitting the graph into its graph layers), and the spread of the layers (i.e., scaling the x and y positions of the nodes). Since navigating large 3D structures suffers from a distorted perspective, clicking the "Top View" button returns the camera to its original position. This 3D Overview naturally visualizes how graphs decompose into layers and highlights how vertices can be cloned throughout multiple layers; if a vertex has clones, they will be stacked vertically along the z-axis (see the two blue vertex clones for "caeciliidae" in Figure 3, left).

4.3 Graph Ribbon: Edge Decomposition Summarization

For each layer produced by the edge decomposition, we compute a set of measures that together provide a quantitative summary of the edge decomposition. We encode these measures for every layer as a horizontal bar glyph to create the visualization in the middle view of the Atlas user interface, called the Ribbon (Figure 3). While there are many graph measures originating from graph theory, graph mining, and network science, we selected five measures to summarize the graph layers that could guide users in exploring and prioritizing their investigation: #edge, #vertex, #clone, #connected components, and clustering coefficient. For example, with our graph decomposition (discussed in next section), layers with denser substructures rise to the top (e.g., quasi-cliques, multi-partite-cores) — this observation motivates our decision to include these five measures because denser layers have a higher clustering coefficient value, a higher vertex-to-edge ratio, and significantly fewer connected components than lower layers (since

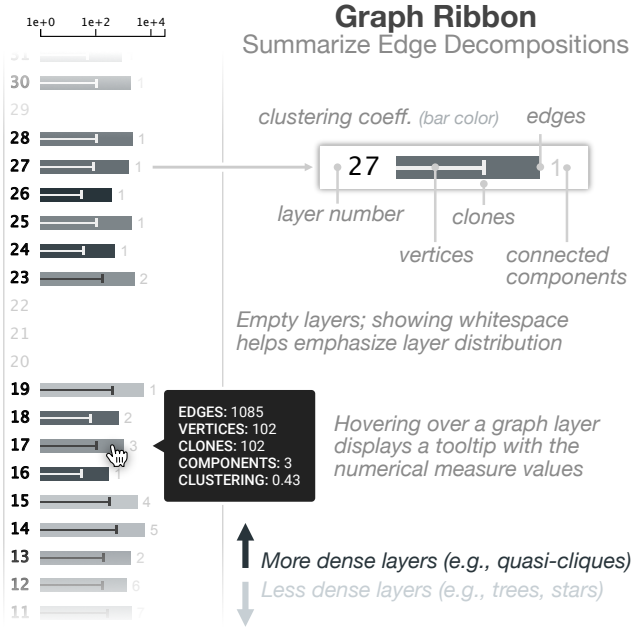

Figure 4: The Ribbon for the Wikipedia GloVe word embedding graph. The graph Ribbon summarizes the edge decomposition using graph measures such as the vertex count, the edge count, the cloned vertex count, clustering coefficient, and number of connected components for each graph layer.

such dense substructures are likely rare and isolated). The clone-to-vertex ratio could indicate how a substructure is connected to other nodes in the graph.

While inspecting graph measures on each layer independently can be enlightening, visualizing each metric across layers as a distribution highlights the power of the edge decomposition. Hovering over a layer displays a tooltip with the five computed measures displayed as numerical values for a given layer. The top of Ribbon includes a menu button that contains options to toggle each of the visualized measures, as well as a linear / log scale toggle for the axis. Clicking on a layer's glyph displays that layer in the Layers view, while a Command+Click displays that layer in 3D in the Overview. Lastly, the entire Ribbon can be dragged using either of the arrows at the top to give more screen real estate to either the Overview or Layers view. Listed below are the five measures and how they are visualized in Atlas.

- **Edges:** The number of edges within each layer. Our edge decomposition partitions the edges into unique layers, therefore the sum of all edges across all graph layers equals the total number of edges in the original graph. This measure is encoded as the large bars in Figure 4.
- **Vertices:** The number of vertices within each layer. Recall this decomposition produces vertex clones, i.e., vertices that have multiple existences across layers; therefore, the sum of all vertices across all graph layers will be at least the total vertices in the original. Note that within a single layer the number of vertices is bounded above by the number of edges in that layer (except for layer 1, in which a single edge can be a connected component; in this case a component contributes 1 edge and 2 vertices). This measure is encoded as the thinner bars in Figure 4.
- **Vertex clones:** The number of vertex clones within each layer. For a vertex to qualify as a clone, it must have one other existence in another layer; therefore, the sum of all clones across all graph layers will be at most the number of vertices in the original graph. Note that within a single layer the number of clones is bounded above by the number of vertices in that layer; this bound is achieved when every vertex of a layer is a clone. This measure is encoded as the thin vertical tick in Figure 4.
- **Connected components:** The number of connected components within each layer. This identifies the number of disjoint graphs within a single graph layer. Note that the number of connected components within a layer is bounded above by the number of vertices in a layer; however, due to the extra constraint generated by fixed-points of degree peeling (i.e., for a vertex v to belong in layer l it must have at least degree l), the number of components in practice is much smaller than the number of vertices. This measure is represented numerically on the right hand side of each layer glyph in Figure 4.
- **Clustering coefficient:** The global clustering coefficient of each layer. This gives a measure of the density of the produced layer. This metric ranges between 0 and 1, where smaller values represent sparsely connected graphs (e.g. trees) and higher values represent denser graphs (e.g. cliques). This measure is encoded as the color of the larger bars in Figure 4, where denser graphs are darker and sparser graphs are lighter.

4.4 Exploration with Graph Layers and Vertex Clones

The last main view of Atlas is the Layers view (Figure 3, right). When a layer in the Ribbon is clicked, Atlas visualizes that layer as an interactive node-link diagram. This visualization is completely interactive: users can zoom and pan on the graph, as well as drag, pin, and select specific vertices. Hovering over a vertex highlights it, its edges, and its neighbors orange (Figure 3, right). The computed layer measures are listed in the top left corner of the Layers view. If the specified layer only contains a single connected component, a message is shown displaying how many edges the component requires to become a complete clique. Conversely, if the specified layer contains multiple connected components, a different message is shown displaying the largest connected component's vertex and edge count. A slider is also available that hides components in decreasing order of their size, i.e., dragging the slider from left to right hides the smallest connected components, eventually showing only the largest component in the layer.

Independent graph layer layouts. Atlas supports multiple interactions for exploring within a single layer. Users can hide and show the vertices or edges of a layer. The "Redraw" toggle animates the layer unraveling using a precomputed independent force-directed layout to better show the decomposition's found structure (Figure 2). However, users can also run a force-directed layout in-browser by clicking the "Live Layout" toggle; the layout computation continues until the toggle is turned off. This can be useful for computing a

larger connected component's layout within a layer; by using the component slider to hide smaller components the desired larger component can be redrawn independently for better structural clarity.

Graph layer contour motifs. The "Motif" toggle computes a contour map of a graph layer by performing kernel-density estimation (KDE) on the layer's vertices. KDE parameter controls are present underneath the toggle. Contour motifs provide a higher-level representation of a graph layer, creating a proxy for vertex density [7, 22, 44]. The contour motif is also instantly recomputed whenever a user drags a vertex or uses one of the above interactions to re-redraw a layer.

Shortest-path-nets via sequential egonet expansion. The "Path" button allows users to explore a single graph layer by building a shortest-path-net representation. When two vertices are selected within a graph layer, clicking the "Path" button will compute the shortest path between the vertices, or an approximation depending on the component size, highlight the computed path blue, and display the vertex labels along this path (seen in Figure 1, right). A user can now select a third vertex somewhere else in the layer and click "Path" once again to find an approximate shortest path from the third vertex to any other vertex along the existing path; iterating this process computes what we call a shortest-path-net via sequential egonet expansion. This mode of exploration is especially useful for observing the transition of semantic information from one side of a large connected component to another.

Vertex clones. Lastly, the "Clone" toggle shows which vertices of a graph layer are clones or not. When toggled on, Atlas colors cloned vertices red and sizes each vertex according to how many clones that vertex has in the entire graph (see Figure 2). When locally exploring a single graph layer, visualizing the vertex clones provides global context for how a particular vertex may participate in many graph layers at once. Conversely, vertices that do not have any clones remain colored gray, and stand out as "secret agents" within a particular layer. These vertices are equally informative, as all of their edges exist within a single layer, indicating that they play a singular, and potentially anomalous, role in the graph. Hovering over a vertex displays its label and lists the other layers its clones exist in. If a user clicks on one of the clones in the list, Atlas shows the selected layer underneath the original visualized layer and centers each of their displays on the selected vertex and its clone (see Figure 3, right). These vertices are now selected and synced, i.e., dragging one of the vertices will also drag the other, updating their position in both layers, reinforcing the notion that a single vertex can participate and influence multiple layers throughout an entire graph. For example, in Figure 3 on the right, the blue vertex "**caeciliidae**" (a worm-like amphibian) in layer 30 bridges two quasi-cliques (families of birds and families of sea snails) together, while its clone in layer 25 participates in another single quasi-clique (families of land creatures). A selected node in the Layers view is also highlighted in the Overview.

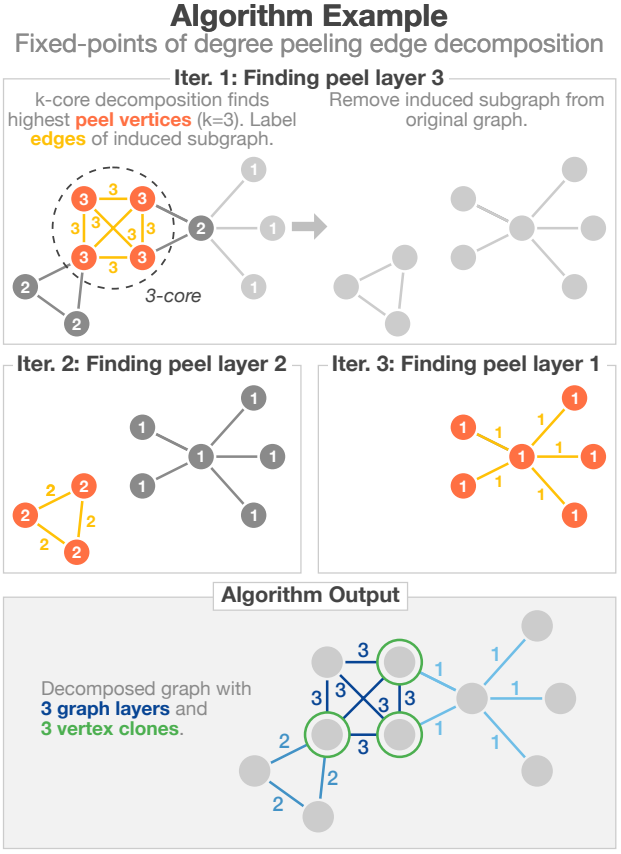

Figure 5: An exemplar graph demonstrating the edge decomposition's procedure. The decomposition is top-down: it finds the highest peel layer first and iterates downward. In the example above, the result is an edge decomposition containing **three layers (shades of blue)** and **three vertex clones (circled in green)**.

5 FAST AND SCALABLE EDGE DECOMPOSITION

5.1 Edge Decomposition Algorithm Summary

Below, we summarize the algorithm, with an accompanying example seen in Figure 5. This algorithm iteratively removes vertices of minimum degree from a graph $G = (V, E)$, which partitions the vertices of any graph into a collection of subsets, each of which is characterized by its iterative minimum degree in G, called the *peel value* of the subset. We call the subgraph induced by the subset of vertices with highest peel value the *EdgeCore* of G, i.e., the highest edge *layer* of G (Figure 5, "Iter. 1: Finding peel layer 3", left). If $EdgeCore(G)$ is different from $E(G)$, remove the *EdgeCore* from $E(G)$ and iterate (Figure 5, "Iter. 1: Finding peel layer 3", right).

This iterative edge decomposition produces an ordered set of *graph layers* where lower layers consist of sparse graph structures depending on the overall minimum degree of the graph (e.g., layer 1 consists of "trees" and "stars") and higher layers consist of higher-order patterns, such as quasi-cliques, which are often candidates to

Table 1: Results for our edge decomposition algorithm across of number of different graphs varying in size and domain. Experimental timings are the average of 5 runs for each graph. We can decompose graphs with millions of edges in seconds, and graphs with hundreds of millions of edges in minutes.

Graph	Graph Type	Vertices	Edges	Time (sec.)	Layers	Highest Peel
Bible Names	co-occurrence	1,774	9,131	0.01	12	15
Google+	social network	23,628	39,242	0.02	10	13
arXiv astro-ph	co-authorship	18,771	198,050	0.10	47	56
Amazon	co-purchase	334,863	925,872	0.12	6	6
US Patents	citation network	3,774,768	16,518,947	11.73	41	64
Pokec	social network	1,632,803	30,622,564	12.33	44	70
LiveJournal	social network	4,847,571	68,993,773	120.70	179	510
Wikipedia Links (German)	hyperlink network	3,225,565	81,626,917	225.40	320	1656
Orkut	social network	3,072,441	117,184,899	91.84	91	253

be explored to find traditional local graph "communities" based on high density. This is seen in the bottom of Figure 5 in the Result section where each edge is labeled with its peel value, and the **three produced graph layers** are colored accordingly in blue. Each such layer can be explored and analyzed, independent of other layers, by using methods that exploit the fact that each vertex in a layer has the same peel value, e.g., layer i contains vertices of peel value i. Whereas each edge is assigned only one peel value, a vertex can appear in multiple layers if it is an endpoint of multiple edges that belong in different layers. We call vertices that appear in multiple layers *cloned vertices*; in the example in Figure 5 in the bottom panel, the **three cloned vertices** are circled in green. Therefore, not only does this edge decomposition algorithm assign peel values to edges, it also produces a vector profile for each vertex describing what layers that vertex exists in, which helps reveal the various roles a vertex can play in the overall graph.

5.2 Speed and Scalability Improvements

Our algorithm uses the ParK multicore [26] k-core decomposition [16] to find the highest peel layer iteratively. To enable our algorithm to work with large graphs that may not fit in main memory, we use the memory mapping (MMap) technique [46], which leverages the fundamental virtual memory capability found on all modern operating systems to load the graph data into the virtual memory space instead. Our algorithm computes traditional k-core decomposition L times, where L is the number of layers in a graph. Therefore, our algorithm runs in $O(LE)$, since we can compute a single k-core decomposition in linear time $O(E)$ [16, 26]; however, our algorithm's speed benefits from the guaranteed inequality $L \leq \sqrt{|V|}$.

Our algorithm's procedure is enumerated in steps below:

(1) **Input**: a graph $G = (V, E)$ represented using an edge list (i.e. source and target pairs) encoded as a binary file.
(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. This is done to access the neighbors of any vertex in a graph in $O(1)$ constant time for undirected graphs. The position of each edge in the original edge list is tracked in the modified edge list.
(3) **MMap:** the modified edge list is memory mapped., i.e., treated as if it were fully loaded into memory [46].
(4) **Vertex degree:** compute the degree of each vertex $v \in V$ in a single pass through the memory mapped edge list.
(5) **k-core decomposition:** the ParK [26] algorithm computes k-cores of the graph. This assigns a core value to each vertex in the graph.
(6) **Compute highest graph layer:** the maximum k-core is taken as the current peel value p. All vertices that have core value equal to the current peel value p are selected and their induced subgraph is computed. This induced subgraph is the pth graph layer.
(7) **Remove graph layer:** all edges in the pth graph layer are labeled with the current peel value p and are logically deleted (using a tombstone array) from the graph G and the degree of their vertices is updated.
(8) **Iterate:** steps 5 though 7 iterate until all the edges $|E|$ have been logically deleted from the graph.
(9) **Output:** for each edge in the original edge list, the corresponding label is recovered and the edge and corresponding peel value label is written to disk.
(10) **Metadata:** some metadata, such as the time taken to preprocess the data and the time taken to run the edge decomposition is written to a separate output log.

5.3 Large Graph Decomposition Experimental Results

We report results on decomposing graphs using our algorithm. We chose a wide range of of graphs, varying in both size (e.g. thousands to hundreds of millions of edges) and domain (e.g. social, hyperlink, and co-occurrence networks). We performed our experiments on a single commodity computer equipped with an Intel i7 6-core processor clocked at 3.3GHz and 32GB of RAM. For each graph, the timing result is averaged over 5 runs. All results are tabulated in Table 1, which includes the graph, its vertex and edge count, the algorithm compute time without preprocessing steps (e.g., data formatting), the number of layers each graph produces, and the highest peel value from the decomposition (since a graph with L layers does not necessarily mean the L layers correspond to $[1, 2, 3, \ldots, L]$). We can decompose graphs with millions of edges in seconds, and graphs

with hundreds of millions of edges in minutes. To the best of our knowledge, this is the first published timing results of this edge decomposition algorithm.

6 SYSTEM DESIGN

Our edge decomposition is implemented in C++; however, we improve performance by leveraging memory mapping [46] to handle large graphs. The edge decomposition runs traditional k-core decomposition L times, where L is the number of layers in the graph; therefore, we use a recent multicore k-core decomposition algorithm to achieve significant speedup [41].

For graph drawing, we avoid slow and computationally expensive force-directed layouts and instead utilize a GPU-powered Barnes-Hutt [14] approximation to achieve significant speedup for computing original graph layouts [21]. Note that computing the edge decomposition of our graph and the global graph layout are independent computations. When both are completed, we process their output together using Python to compute graph layer measures, vertex clones, and format the data to be ingested by Atlas. The visualization system is web-based and uses the latest JavaScript libraries for graphics rendering (D3 [19] for 2D and three.js (https://threejs.org) for 3D). Both the algorithm and Atlas are open-sourced.

7 USER STUDY

7.1 Study Description

To better understand how Atlas may help graph data analysts, we recruited three graph experts, who all work with graphs on a weekly basis, to use Atlas to explore three graph datasets. The three graph experts include:

P_1: machine learning research scientist at Symantec Research
P_2: cybersecurity researcher at NASA Jet Propulsion Lab
P_3: software systems engineer at NASA Jet Propulsion Lab

All participants hold a PhD in computer science or mathematics. Our participants' average age was 33, and all three were male. Each session lasted 90 minutes, and the participants were paid $10 for their time. Participants completed an intro questionnaire to provide demographic data and information about how they use graphs in their work. Then, we introduced them to the edge decomposition algorithm, including a walk-through of the example presented in Figure 5. Next, each participant is guided through Atlas's user interface and demoed all available interactions. At the end of the study, participants completed an exit questionnaire to provide feedback on Atlas. We recorded audio and video during each session.

Table 2 summarizes the three graphs: a Yelp user-user review network [39] where two users are connected if they both reviewed the same 5 venues over 9 days; an SEC traders graph [60] where two traders are connected if they traded more than 5 times during the same day; and the GloVe word embedding graph described earlier in Illustrative Scenario. We chose these three graph datasets because they contain interesting structures (e.g., traders whose transactions often coincide suspiciously in the SEC graph) and encourage investigative behavior in our participants. Thus, the participants' goal was to analyze the three graphs using Atlas to spot any patterns that they would consider as interesting, mimicking what they may do in their own work. They were free to use any features of Atlas to allow for more natural use of the system.

7.2 Key Observations

We summarize our key observations from interacting with the three participants into three themes, each highlighting how Atlas helped them with their exploration.

3D for overview, 2D for details. Upon displaying a new graph in Atlas, all three participants used the 3D representation to develop an overview and build intuition about the graph structure. For example, they commonly used the "Show All" button to show all layers; and the "Height" slider to place layers into a 3D structure, to help them see multiple perspectives of the decomposition. After a few minutes of exploring the 3D representation, all participants gradually transitioned to inspecting the Ribbon and interacting with graph layers in the Layers view. Participants used the Ribbon as both a summarization of the graph and as a mechanism to identify potentially interesting graph layers for detailed investigation. While interacting with the Layers view, all participants used the hovering interaction to show a vertex's immediate neighbors often. Participants would also re-draw some selected layers, which is particularly helpful for larger graphs like the GloVe word embedding graph, where the re-drawn layout helps better reveal structures that the decomposition found. P_3 had more interactions total than the other two participants, making most use of the path feature to build shortest-path-net representations of each layer. P_2 was more specific about which vertices to explore, only using certain features decisively instead of the more causal exploration demeanor of P_3.

In summary, P_1 and P_3 spent a majority of their sessions exploring the graph layers in 2D, e.g., P_1 would drag the Ribbon to the left hand side of the UI to give more screen real estate to the Layers view; however, P_2 alternated between the 3D and 2D views during his entire session. P_1 further attributed his actions for using the 3D representation as an overview and the Layers view for more fine-grained exploration by referencing the well-known Visual Information-Seeking Mantra [59]: "overview first, zoom and filter, then details on demand."

Identifying and linking meaningful graph substructures. While the participants analyzed the structure of each layer independently, they also frequently toggled the vertex clones to see which vertices in a particular layer existed elsewhere in the graph. Each participant made thorough use of cross-layer exploration, by entering a new layer from a particular vertex's clone via an original layer (as an example, see Figure 1C). P_3 said: *"I thought it was extremely useful to [move] from one layer to another. Most of my time using the system was exploring one layer and then finding the clones from that layer to the other layers. The system presented this very well."*

P_1 said syncing the position of a vertex and its clones across multiple layers was useful for identifying how a single vertex could play different roles in the graph. This was particularly evident during P_3's exploration of the GloVe word embedding, where he found multiple semantically related connected components of words (components included types of wine, British towns, and medical disciplines). As he was forming higher level concepts, he said: *"I could spend all day doing this."*

Table 2: Participant graph data. The three graphs the participants explored in the user study.

Graph	Graph Meaning	Vertices	Edges	Layers	Highest Peel
Yelp Reviews Network	**Vertices:** Yelp reviewers. **Edges:** connect two reviewers if both review the same 5 venues over 9 days	996	1,189	6	10
SEC Insider Trading Graph	**Vertices:** traders at companies. **Edges:** connect two traders if their trades (purchases) coincide with each other on 5 or more days	1,678	2,631	12	14
GloVe Word Embedding Graph	**Vertices:** words. **Edges:** connect two words if the angular distance between their word vectors is less than 0.9	65,870	213,526	31	40

Application to anomaly detection. When analyzing the Yelp network, all three participants came to the same conclusion that likely the top and most dense layers of the graph showed potential for unusual reviewer activity, since finding a quasi-clique of 12 reviewers who all have reviewed at least the same 5 venues over 9 days seems highly improbable. This could indicate a fake Yelp review scheme where multiple accounts are assigned, or even paid, to review particular venues. Besides looking at the topmost layer in each graph, each participant also used the "Clones" toggle to identify any vertices that had *no* clones, suggesting that these vertices are suspicious and are worth investigating. For example, P_3 discovered that the top most layer in the SEC trading network (layer 14) is a complete 15-clique (Figure 6, left), but visualizing this layer's clones reveals that 8 of the vertices have **clones** that form a star graph in layer 1. Most interestingly, this star's **hub** in layer 1 is *not* a clone (Figure 6, right). Together, this means that of the 15 traders who all trade with every other trader (layer 14), 8 of them also trade with one other less-connected trader. P_2 was particularly enticed by using Atlas for these anomaly detection-like tasks, and wrote in his exit questionnaire: *"I like the fact that the analysis (using a combination of vertex clones and layers) naturally reveals potentially anomalous substructures and vertices. This is highly useful from a cybersecurity perspective."*

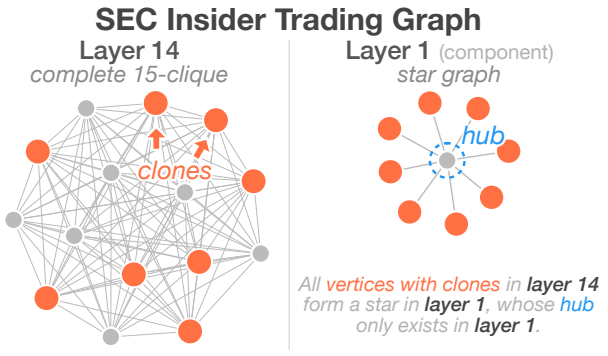

Figure 6: Identifying suspicious trading. A participant found that the top most layer in the SEC insider trading graph (layer 14, left) is a complete 15-clique, but when visualizing this layer's **clones (red vertices)** it is revealed that 8 of the vertices have clones that form a star graph in layer 1 (right). Most interestingly, this star's **hub (blue vertex)** in layer 1 is *not* a clone.

7.3 User Feedback for System Improvements

Overall, all participants recorded in their exit questionnaires that they enjoyed using the system, it was well-designed, and it presented the data from the decomposition effectively. After the study, we asked the participants to critique our approach and suggest improvements.

More analytic features for deployment. P_1 suggested that if Atlas were to be deployed that it could benefit from the inclusion of more graph analytic features (e.g., incorporating vertex metadata into the system), which could better assist graph data analysts when trying to take action upon an interesting layer or vertex clone. P_3 similarly suggested an interaction for selecting arbitrary connected components in any layer to display component specific measures and information.

Suggesting interesting layers to investigate. However, the most interesting improvement suggested by P_2 points to potential future work; he suggested that Atlas itself should recommend the most peculiar layers based on some edge decomposition-specific notion of "interestingness" to explore first. Right now, Atlas requires a user to glean this notion by inspecting each layer's glyph in the Ribbon and reasoning about it, e.g., how dark the bar is (indicating a high clustering coefficient) or how close the clone tick is to the inner bar (indicating how many of the vertices are clones). Lastly, two participants desired to see how the discovered graph layers better compare to standard graph motifs, suggesting that comparing layers and standard structures could enable analysts to explore massive graphs more quickly and effectively.

8 DISCUSSION AND FUTURE WORK

There are multiple promising directions of future research for pairing edge decompositions and interactive data visualization together to further improve graph exploration.

Dynamic graph decomposition visualization. All graphs decomposed and visualized in this work are non-dynamic graphs. Applying the edge decomposition to time-varying and dynamic graphs to visualize how graph layers change given the addition or removal of particular edges or vertices could reveal deeper insight into important structure within dynamic graphs. An interactive visualization system like Atlas for dynamic edge decompositions could also motivate novel interactions and visualizations for how to make sense of a changing decomposition over time.

Further scalability improvements. Computationally, while our edge decomposition algorithm is fast, and scales to graphs with hundreds of millions of edges (e.g., the Wikipedia Links (English) hyperlink network with 12M vertices and 378M edges takes 2,237 seconds to decompose), it could be further improved by leveraging GPUs for computing peel values. While our decomposition partitions a graph into smaller subgraph layers, we can further improve visual scalability of our system by adopting visual reduction techniques described in Related Works as additional options for exploring a single graph layer, such as hierarchical super-noding [4, 10, 11], edge bundling [5, 24, 36], and graph summarizations and motif abstractions [27, 42, 68]. There is also potential to incorporate relevance based exploration techniques for exploration within a single large graph layer, such as [23, 33, 55].

New graph representations. Given two graph layers l_i and l_j with different peel values whose vertices intercept one another (i.e., layer l_i has at least one clone in l_j), can we efficiently compute the vertex clone intersection of the two layers? Once computed, is it best to visualize this intersection of the vertex clones in 2D or possibly using a 3D volumetric representation to generate new, anthropomorphic representations of graphs? This could potentially aid with the task of comparing the underlying structure of large graphs that evolve over time.

9 CONCLUSION

We introduce Atlas: an interactive graph exploration system that wields a fast and scalable edge decomposition algorithm. Atlas introduces a new approach for exploring large graphs that simultaneously reveals (1) peculiar subgraph structure discovered through the decomposition's layers, and (2) possible vertex roles in linking such subgraph patterns across layers. We presented the results from a user study with three graph experts and highlighted some of the findings made possible by Atlas when applied to graphs from multiple domains. Atlas runs in-browser, and is open sourced.

ACKNOWLEDGMENTS

We thank our study participants for their time and the anonymous reviewers for their constructive feedback. This work was supported by NSF grants IIS-1563816, IIS-1563971, TWC-1526254, CNS-1704701, and a NASA Space Technology Research Fellowship.

REFERENCES

[1] James Abello, Fred Hohman, and Duen Horng Chau. 2015. 3D exploration of graph layers via vertex cloning. In *IEEE Conference on Visual Analytics Science and Technology, Poster*.
[2] James Abello and Jeffrey Korn. 2002. MGV: a system for visualizing massive multidigraphs. *IEEE Transactions on Visualization and Computer Graphics* 8, 1 (2002), 21–38.
[3] James Abello and François Queyroi. 2014. Network decomposition into fixed points of degree peeling. *Social Network Analysis and Mining* 4, 1 (2014), 1–14.
[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] Basak Alper, Benjamin Bach, Nathalie Henry Riche, Tobias Isenberg, and Jean-Daniel Fekete. 2013. Weighted graph comparison techniques for brain connectivity analysis. In *ACM Conference on Human Factors in Computing Systems*. ACM, 483–492.
[6] Basak Alper, Tobias Hollerer, JoAnn Kuchera-Morin, and Angus Forbes. 2011. Stereoscopic highlighting: 2d graph visualization on stereo displays. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2325–2333.
[7] Basak E Alper, Nathalie Henry Riche, and Tobias Hollerer. 2014. Structuring the space: a study on enriching node-link diagrams with visual references. In *ACM Conference on Human Factors in Computing Systems*. ACM, 1825–1834.
[8] 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.
[9] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. 2015. Vqa: visual question answering. In *IEEE International Conference on Computer Vision*. 2425–2433.
[10] Daniel Archambault, Tamara Munzner, and David Auber. 2007. Grouse: feature-based, steerable graph hierarchy exploration.. In *The EG and VGTC Conference on Visualization*, Vol. 2007. 67–74.
[11] 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).
[12] David Auber. 2004. Tulip: a huge graph visualization framework. In *Graph drawing software*. Springer, 105–126.
[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations*.
[14] Josh Barnes and Piet Hut. 1986. A hierarchical O (N log N) force-calculation algorithm. *Nature* 324, 6096 (1986), 446.
[15] Mathieu Bastian, Sebastien Heymann, Mathieu Jacomy, et al. 2009. Gephi: an open source software for exploring and manipulating networks. *AAAI Conference on Weblogs and Social Media* 8 (2009), 361–362.
[16] Vladimir Batagelj and Matjaz Zaversnik. 2003. An O(m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049* (2003).
[17] Michael Baur, Ulrik Brandes, Marco Gaertler, and Dorothea Wagner. 2004. Drawing the AS graph in 2.5 dimensions. In *International Symposium on Graph Drawing*. Springer, 43–48.
[18] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3, Feb (2003), 1137–1155.
[19] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D^3 data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309.
[20] Ulrik Brandes and Steven R Corman. 2003. Visual unrolling of network evolution and the analysis of dynamic discourse. *Information Visualization* 2, 1 (2003), 40–50.
[21] Govert G Brinkmann, Kristian FD Rietveld, and Frank W Takes. 2017. Exploiting GPUs for fast force-directed visualization of large-scale networks. In *International Conference on Parallel Processing*. IEEE, 382–391.
[22] Nan Cao, Jimeng Sun, Yu-Ru Lin, David Gotz, Shixia Liu, and Huamin Qu. 2010. Facetatlas: multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics* 16, 6 (2010), 1172–1181.
[23] Duen Horng Chau, Aniket Kittur, Jason I Hong, and Christos Faloutsos. 2011. Apolo: making sense of large network data by combining rich user interaction and machine learning. In *ACM Conference on Human Factors in Computing Systems*. ACM, 167–176.
[24] Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. 2008. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics* 14, 6 (2008), 1277–1284.
[25] Tommy Dang, Paul Murray, and Angus Forbes. 2017. BioLinker: bottom-up exploration of protein interaction networks. In *IEEE Pacific Visualization Symposium*. IEEE, 265–269.
[26] Naga Shailaja Dasari, Ranjan Desh, and Mohammad Zubair. 2014. ParK: an efficient algorithm for k-core decomposition on multicore processors. In *IEEE International Conference on Big Data*. IEEE, 9–16.
[27] Cody Dunne and Ben Shneiderman. 2013. Motif simplification: improving network visualization readability with fan, connector, and clique glyphs. In *ACM Conference on Human Factors in Computing Systems*. ACM, 3247–3256.
[28] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. 2001. GraphvizâĂŤopen source graph drawing tools. In *International Symposium on Graph Drawing*. Springer, 483–484.
[29] Ozan Ersoy, Christophe Hurter, Fernando Paulovich, Gabriel Cantareiro, and Alex Telea. 2011. Skeleton-based edge bundling for graph visualization. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2364–2373.
[30] 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.
[31] Emden R Gansner, Yifan Hu, Stephen North, and Carlos Scheidegger. 2011. Multilevel agglomerative edge bundling for visualizing large graphs. In *IEEE Pacific Visualization Symposium*. IEEE, 187–194.
[32] Dedre Gentner and Arthur B Markman. 1997. Structure mapping in analogy and similarity. *American Psychologist* 52, 1 (1997), 45.

[33] Jeffrey Heer and Stuart K Card. 2004. DOITrees revisited: scalable, space-constrained visualization of hierarchical data. In *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM, 421–424.

[34] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. Rolx: structural role extraction & mining in large graphs. In *ACM Conference on Knowledge Discovery and Data Mining*. ACM, 1231–1239.

[35] Ivan Herman, Guy Melançon, and M Scott Marshall. 2000. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics* 6, 1 (2000), 24–43.

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

[37] Danny Holten and Jarke J Van Wijk. 2009. Force-directed edge bundling for graph visualization. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 983–990.

[38] Keith J Holyoak and Paul Thagard. 1997. The analogical mind. *American Psychologist* 52, 1 (1997), 35.

[39] Paras Jain, Shang-Tse Chen, Mozhgan Azimpourkivi, Duen Horng Chau, and Bogdan Carbunar. 2015. Spotting suspicious reviews via (quasi-) clique extraction. *IEEE Symposium on Security and Privacy, Poster* (2015).

[40] Toma Jeowicz, Milo Kudelka, Jan Plato, and Vaclav Snael. 2013. Visualization of large graphs using gpu computing. In *International Conference on Intelligent Networking and Collaborative Systems*. IEEE, 662–667.

[41] 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.

[42] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2014. VOG: summarizing and understanding large graphs. In *SIAM International Conference on Data Mining*. SIAM, 91–99.

[43] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. 2015. Summarizing and understanding large graphs. *Statistical Analysis and Data Mining: The ASA Data Science Journal* 8, 3 (2015), 183–202.

[44] Yu-Ru Lin, Jimeng Sun, Nan Cao, and Shixia Liu. 2010. Contextour: contextual contour visual analysis on dynamic multi-relational clustering. In *SIAM International Conference on Data Mining*. SIAM, 418–429.

[45] 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*.

[46] Zhiyuan Lin, Minsuk Kahng, Kaeser Md Sabrin, Duen Horng Polo Chau, Ho Lee, and U Kang. 2014. Mmap: fast billion-scale graph computation on a pc via memory mapping. In *IEEE International Conference on Big Data*. IEEE, 159–164.

[47] 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.

[48] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[49] Tamara Munzner. 1997. H3: Laying out large directed graphs in 3D hyperbolic space. In *IEEE Symposium on Information Visualization*. IEEE, 2–10.

[50] Alexandros Panagiotidis, Guido Reina, Michael Burch, Tilo Pfannkuch, and Thomas Ertl. 2015. Consistently GPU-accelerated graph visualization. In *International Symposium on Visual Information Communication and Interaction*. ACM, 35–41.

[51] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*. 1532–1543.

[52] Robert Pienta, James Abello, Minsuk Kahng, and Duen Horng Chau. 2015. Scalable graph exploration and visualization: sensemaking challenges and opportunities. In *International Conference on Big Data and Smart Computing*. IEEE, 271–278.

[53] 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.

[54] Robert Pienta, Fred Hohman, Acar Tamersoy, Alex Endert, Shamkant Navathe, Hanghang Tong, and Duen Horng Chau. 2017. Visual graph query construction and refinement. In *ACM International Conference on Management of Data*. ACM, 1587–1590.

[55] Robert Pienta, Minsuk Kahng, Zhiyuan Lin, Jilles Vreeken, Partha Talukdar, James Abello, Ganesh Parameswaran, and Duen Horng Chau. 2017. Facets: adaptive local exploration of large graphs. In *SIAM International Conference on Data Mining*. SIAM, 597–605.

[56] Robert Pienta, Acar Tamersoy, Alex Endert, Shamkant Navathe, Hanghang Tong, and Duen Horng Chau. 2016. Visage: interactive visual graph querying. In *ACM International Working Conference on Advanced Visual Interfaces*. ACM, 272–279.

[57] 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).

[58] David Selassie, Brandon Heller, and Jeffrey Heer. 2011. Divided edge bundling for directional network data. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2354–2363.

[59] Ben Shneiderman. 2003. The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization*. Elsevier, 364–371.

[60] Acar Tamersoy, Elias Khalil, Bo Xie, Stephen L Lenkey, Bryan R Routledge, Duen Horng Chau, and Shamkant B Navathe. 2014. Large-scale insider trading analysis: patterns and discoveries. *Social Network Analysis and Mining* 4, 1 (2014), 201.

[61] Stef Van den Elzen and Jarke J Van Wijk. 2014. Multivariate network exploration and presentation: From detail to overview via selections and aggregations. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2310–2319.

[62] Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J van Wijk, J-D Fekete, and Dieter W Fellner. 2011. Visual analysis of large graphs: state-of-the-art and future research challenges. In *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 1719–1749.

[63] Pak Chung Wong, Chaomei Chen, Carsten Gorg, Ben Shneiderman, John Stasko, and Jim Thomas. 2011. Graph analytics-lessons learned and challenges ahead. *IEEE Computer Graphics and Applications* 31, 5 (2011).

[64] Pak Chung Wong, Harlan Foote, George Chin, Patrick Mackey, and Ken Perrine. 2006. Graph signatures for visual analytics. *IEEE Transactions on Visualization and Computer Graphics* 12, 6 (2006), 1399–1413.

[65] Pak Chung Wong, Harlan Foote, Patrick Mackey, George Chin, Heidi Sofia, and Jim Thomas. 2008. A dynamic multiscale magnifying tool for exploring large sparse graphs. *Information Visualization* 7, 2 (2008), 105–117.

[66] Pak Chung Wong, David Haglin, David Gillen, Daniel Chavarria, Vito Castellana, Cliff Joslyn, Alan Chappell, and Song Zhang. 2015. A visual analytics paradigm enabling trillion-edge graph exploration. In *IEEE Symposium on Large Data Analysis and Visualization*. IEEE, 57–64.

[67] Pak Chung Wong, Han-Wei Shen, Christopher R Johnson, Chaomei Chen, and Robert B Ross. 2012. The top 10 challenges in extreme-scale visual analytics. *IEEE Computer Graphics and Applications* 32, 4 (2012), 63–67.

[68] Vahan Yoghourdjian, Tim Dwyer, Karsten Klein, Kimbal Marriott, and Michael Wybrow. 2018. Graph thumbnails: identifying and comparing multiple graphs at a glance. In *IEEE Transactions on Visualization and Computer Graphics*.