Graph-Based Techniques for Visual Analytics of Scientific Data Sets

Chaoli Wang

University of Notre Dame

1 Introduction

Imagining a typical workflow for a climate scientist to explore a 3D volumetric data set produced from a simulation. The data set is time-varying and multivariate, containing scalar and vector quantities. The scientist first plays back the entire time sequence of the precipitation scalar field and spots some "features" of interest. She wants to extract those features and track them over time. However, the features are difficult to select or pick directly in 3D. Visually tracking multiple features over time is very difficult as she has to rely on her memory to tie together their connections. Furthermore, she also integrates a set of flow lines from the wind velocity vector field and needs to examine their relationships. Visual exploration of these flow lines poses quite a challenge due to the lack of capability to observe them and their spatial relationships in an occlusion-free and controllable fashion. Moving forward, she even hopes to investigate these scalar and vector quantities simultaneously. For instance, identifying regions of high pressure and low precipitation and studying flow lines passing through these regions only, or exploring flow lines that connect attracting spiral saddles and repelling spiral saddles. This becomes increasingly impossible without a new solution for data abstraction and relationship exploration. As the size and complexity of scientific data continue to grow, all these challenges will only become more severe.

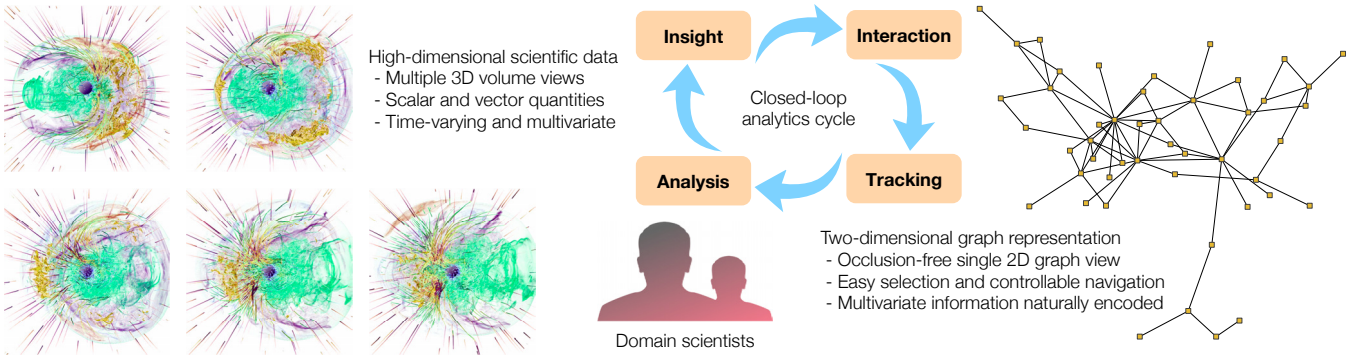

Figure 1: Mapping relationships extracted from the high-dimensional scientific data to the 2D graph representation for visual analytics and knowledge discovery.

The above challenges call for a transformative view to explore scientific data sets. Over the years, intensive research efforts have been spent on developing better algorithms and techniques for processing, managing, and rendering scientific data. However, much less attention has been paid to the design of effective visual analytics tools that fully integrate techniques from information visualization into the scientific visualization workflow. As sketched in Figure 1, instead of being confined to the original spatiotemporal domain, we should extract data relationships over space and time, display these relationships in a low-dimensional transformed space, and enable users to perform queries and make connections to the original data. Extracting data relationships makes it possible for users to make clear and quantified observations at runtime. Turning the visual representation of data relationships into a user interface for navigation essentially transforms users from passive observers to active participants in the visual analytics process. As a result, we are able to address scientific visualization problems whose size, complexity, and the need for closely coupled human and machine analysis may make them otherwise intractable.

In information visualization, graphs are well-known structures that use node-link diagrams to represent various kinds of data relationships. Transforming this generic and powerful visual means into a visual interface for navigating and exploring scientific data sets requires a fully integrated pipeline of data transformation, representation, and visual mapping. In this article, we present our recent works of TransGraph [5] for time-varying data visualization and FlowGraph [7] for flow field exploration. Both graph designs are hierarchical, enabling level-of-detail exploration of large scientific data in an adaptive manner. Moreover, we describe our mining approach to abstract TransGraph via motif simplification and support visual recommendation [6]. We also explore a different way to organize different "objects" associated with a flow field into a semantic flow graph [11], contrasting the structural FlowGraph design and highlighting the differences it makes. For a comprehensive overview of the past, present, and future of graphs in scientific visualization, we refer readers to the survey paper by Wang and Tao [12].

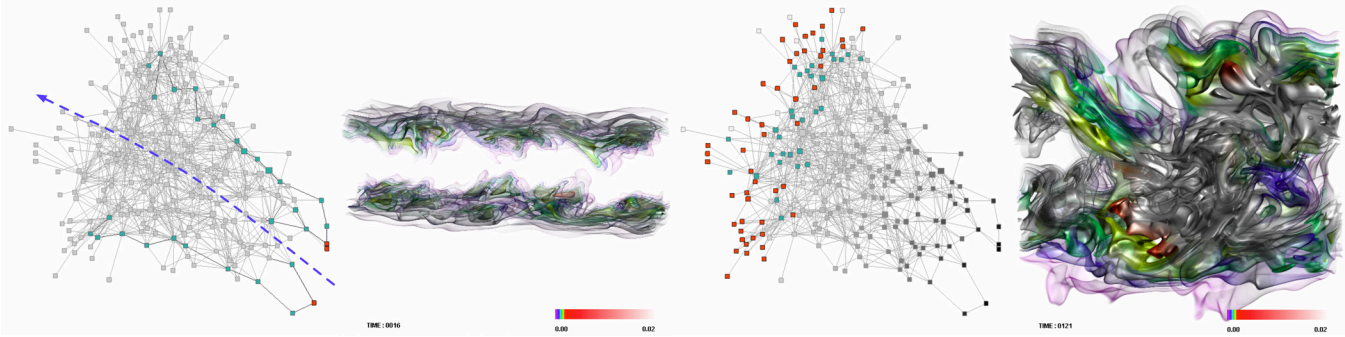

Figure 2: Transition query of the combustion data set using TransGraph. (a) shows the query of the transitions having their frequency larger than 0.12. (b) shows the volume highlighting corresponding to the red nodes brushed at time step 16. (c) shows the query of the states that are only involved in self-transition at time step 121. (d) shows the volume highlighting corresponding to the red nodes shown in (c).

2 TransGraph

Time-varying data are all about changes. What makes time-varying data visualization unique yet challenging is the very dynamic behaviors of the data. Among them, transitions among data items over time capture the evolution of time-varying data and therefore are the most important relationships we seek to extract and visualize. We advocate an adaptive approach by building states from data blocks and clustering states into a hierarchy. By extracting a global view of data transitions, we aim to present states and transitions in an occlusion-free, controllable, and adaptive manner using a graph-based representation.

2.1 Hierarchical State Transition and Graph Construction

Given a time-varying volumetric data set, we first perform uniform subdivision of each time step separately into a list of data blocks with an equal size. We use the histograms of blocks to compute their distances, define states based on blocks, and cluster states hierarchically to extract their transition relationships. From partitioned blocks, we identify representative blocks and compute a distance matrix recording the distance between any two of them. Later on, for any two blocks in the volume, we simply look up the distance between their representative blocks as the approximation.

We define a *state* as a configuration of neighboring spatiotemporal blocks that are characterized by their value distributions. In hierarchical state clustering, we select a representative state from a group of states. The group is formed by merging similar states that are spatially overlapping or neighboring and are in the same or neighboring time steps. For efficiency, we use a method similar to region growing in our clustering. The spatial and temporal constraints ensure that clustered states are space-time continuous which makes the subsequent tracking process meaningful. A *transition* occurs between two states, from one state at time step t to another state at time step $t+1$, if and only if their corresponding central blocks are the same in the spatial domain. We also define the directional transition probability to indicate how frequently a state is transferred to another state.

We create TransGraph, a hierarchical state transition graph, to record transition relationships among states at various levels of detail. In this way, we convert a 4D space-time data set to a single graph which can be visualized in 2D. In TransGraph, a node represents a state (leaf) or a cluster of states (non-leaf) and the edge between two nodes represents their transition probabilities.

2.2 Visual Exploration

TransGraph stores rich information about the nodes (states) and edges (transitions) over time. To make the best use of TransGraph, we design a set of intuitive queries, such as state query, transition query, and time step query, to enable knowledge discovery from the underlying time-varying data. We dynamically link together the volume view and the graph view for dual-domain interaction.

Figure 2 shows an example of transition query. The evolution of time in TransGraph is indicated by the blue dashed arrow. The graph reveals that in early time steps, few states and transitions are available. The numbers of states and transitions increase as the time step increases. This indicates the increasingly turbulent nature of the combustion data set over time. Such a finding can be solely inferred from the graph view without the presence of the volume view. The volume view can help us put the graph in the context, make the connection to the spatiotemporal data, and confirm our findings derived from TransGraph. For example, the nodes satisfying the query condition are

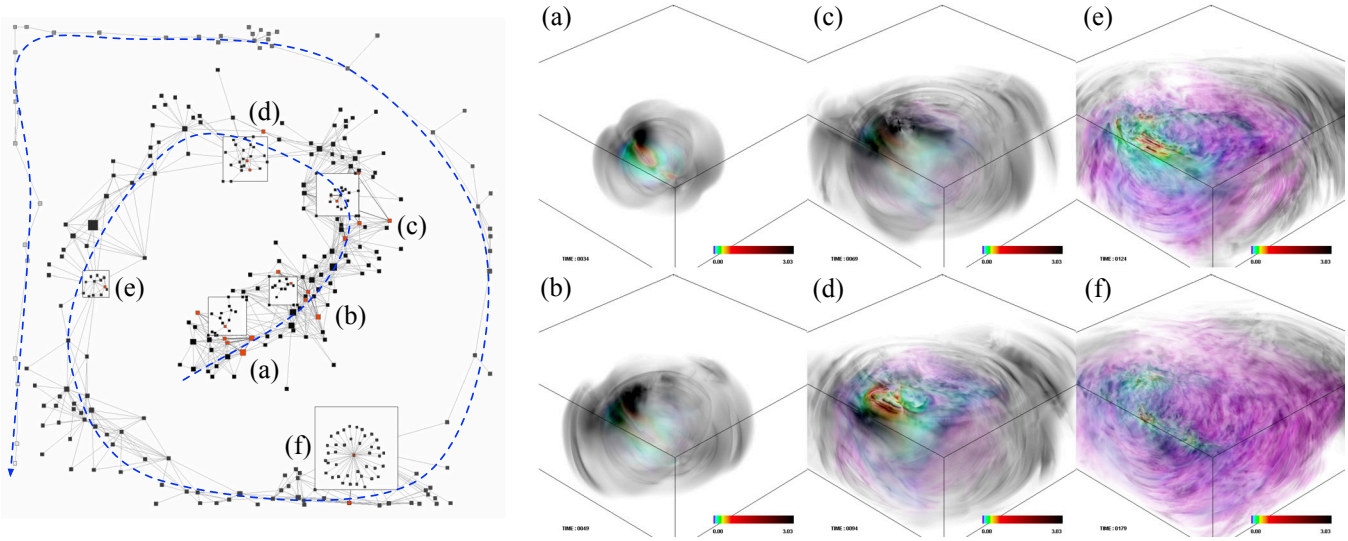

Figure 3: Dynamic tracking of the earthquake data set using TransGraph. A volumetric region corresponding to the epicenter is selected at time step 34, which is highlighted in its original color saturation in (a). (a) to (f) are the dynamic tracking results at selected time steps 34, 49, 69, 94, 124, and 179, respectively.

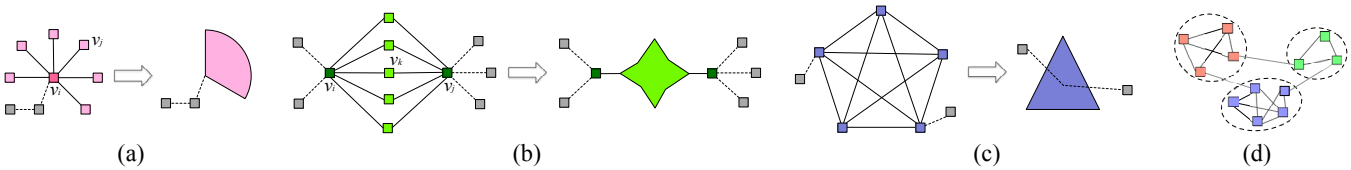

Figure 4: (a) to (c) are fan, connector, and clique and their simplified forms. (d) shows a simple graph with three communities, enclosed by the dashed circles.

highlighted. In (a), nodes having the high frequency at the current time step are shown in red while others are shown in green. In (c), nodes only involved in self-transition are in red and the rest of nodes at the same time step are in green. The corresponding volumetric regions are highlighted in their original color saturation.

Figure 3 shows an example of dynamic tracking, which conveys the impression of data transition in the volume while tracking the corresponding state changes over time. A spiral pattern is formed for TransGraph due to a large number of time steps (599) involved. The graph reveals that the earthquake data set consists of many more states and transitions in the early time steps than later time steps. This matches the fact that the earthquake breakout is within the first 200 time steps. After that, it diminishes gradually for the rest of 400 time steps. Even though the underlying transitions are computed in a block-wise manner, we allow the user to adjust the propagation speed to give the impression of voxel-wise data transition. We consider the distances of a voxel to the centers of initial blocks selected to further adjust the color saturation of the voxel. Corresponding to the precise tracking result, in the graph view we highlight graph nodes at different levels of hierarchy in red.

3 Mining TransGraph

When the data set is large and the relationships are complex, navigating and exploring the resulting graph is a daunting task. This motivates us to investigate a mining approach that automatically extracts graph features from TransGraph for cost-effective exploration and understanding. Beyond straightforward graph properties such as node size and edge density, users are now provided with further guidance through a series of graph analysis techniques including graph simplification, community detection, and visual recommendation. These new techniques provide the convenience and capabilities for graph exploration which is difficult or impossible to achieve through standard interaction techniques.

3.1 Graph Simplification and Community Detection

Graph simplification condenses a large TransGraph to a smaller one by abstracting known structures, such as fan, connector, and clique [2], presenting a less cluttered view for quick comprehension of the overall graph structure. In a fan (Figure 4 (a)), a node v_i transits to another node v_j at later time steps and transit back to itself eventually. v_j can be considered as an *interruption* of v_i. As a result, a fan could represent a volumetric region that has *turbulence* and eventually returns to the original state. In a connector (Figure 4 (b)), the nodes (e.g., v_k) in between can be

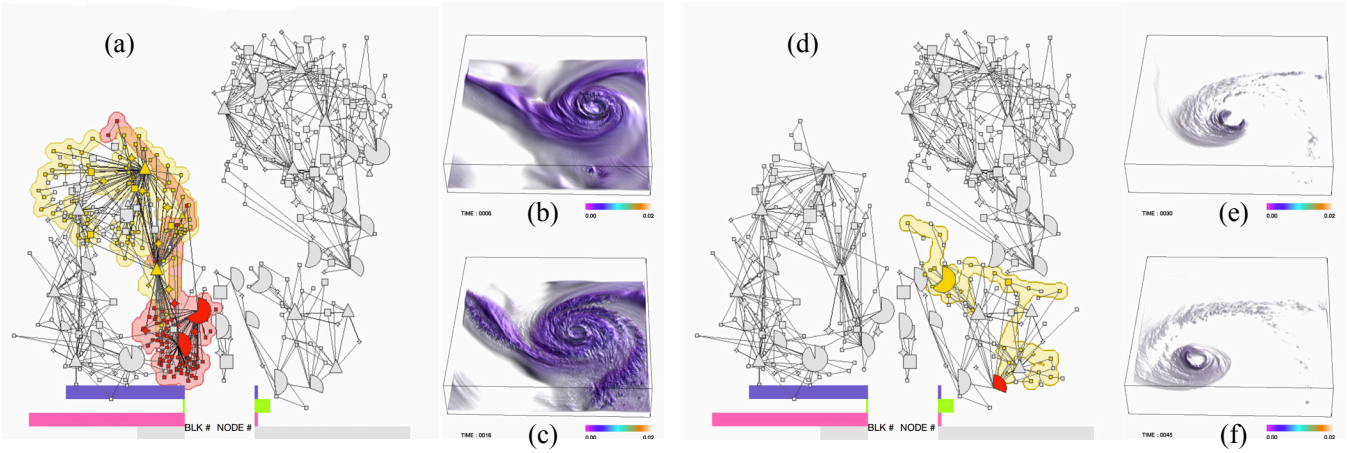

Figure 5: Visual recommendation of the hurricane data set using TransGraph. (a) shows a chosen community highlighted in red and the recommended community highlighted in yellow. (b) and (c) show the rendering of the corresponding blocks of the chosen and recommended communities at time steps 6 and 16, respectively. (d) shows a chosen fan highlighted in red and the recommended nodes highlighted in yellow. (e) and (f) show the rendering of the corresponding blocks of the chosen node at time step 30 and recommended nodes at time step 45, respectively.

considered as a set of *intermediate states* between the two ending nodes v_i and v_j. Therefore, a connector represents a *transition* from a state v_i to another state v_j. A clique (Figure 4 (c)) is a simplification of several nodes that have all-to-all connections between them. Because the corresponding states transit among themselves, a clique represents a *locally stable region* in the volume.

Community detection organizes nodes with close relationships into groups, allowing visual comparison between groups of nodes instead of individual nodes. Since a community (Figure 4 (d)) could be treated as a less stringent clique, it could represent a *relatively stable region*. The stronger the link density of the community, the more stable the region. Each community can be replaced by a symbol as well for simplification.

3.2 Visual Recommendation

Visual recommendation automatically highlights individual nodes or node groups based on user-selected items, thus enabling users to spend more time on the actual analysis instead of painstaking interaction. We support *community recommendation* and *node recommendation*. For community recommendation, we assume that a list of communities is detected from the graph and users select a community of interest. We convert each community to an unweighted undirected graph and treat the similarity between two graphs as the similarity between the two corresponding communities. To calculate the distance between two graphs, we use a graph matching algorithm [9] which finds a serial ordering of the nodes via random walks and converts graphs to strings. By comparing the difference between the two strings, we compute the difference between the two graphs. For node recommendation, we recommend similar nodes using the SimRank algorithm [4]. SimRank considers two nodes to be similar if they are related to similar nodes. We revise it to accommodate weighted directed graphs such as TransGraph.

Figure 5 shows an example of visual recommendation. The correspondence of volume highlighting results demonstrates the effectiveness of visual recommendation. Communities and node groups are highlighted using Bubble Sets [1]. Bubble Sets use continuous, often concave, isocontours to delineate group memberships, producing tight capture of group members with less ambiguity compared with using convex hulls. Although highlighting communities or node groups itself does not reduce visual complexity, the reduction in cognition overhead and interaction cost allows users to achieve effective visual understanding and analytics.

4 FlowGraph

Striking a balance among coverage, occlusion, and complexity is a resounding theme in the visual understanding of large-scale flow field data. We design FlowGraph, a graph-based visual representation and interface that substantially increases our ability to understand and explore a flow field at various levels of detail, providing the clarity and flexibility previously unavailable. In the following, we describe FlowGraph in the context of steady flow fields. Its extension to unsteady flow fields is presented in [8].

4.1 Graph Definition, Construction, and Drawing

We define FlowGraph as a compound hierarchical graph that consists of streamline clusters (L-nodes) and spatial regions (R-nodes) as nodes. Edges are formed between L-nodes (L-L edges) indicating the number of common regions

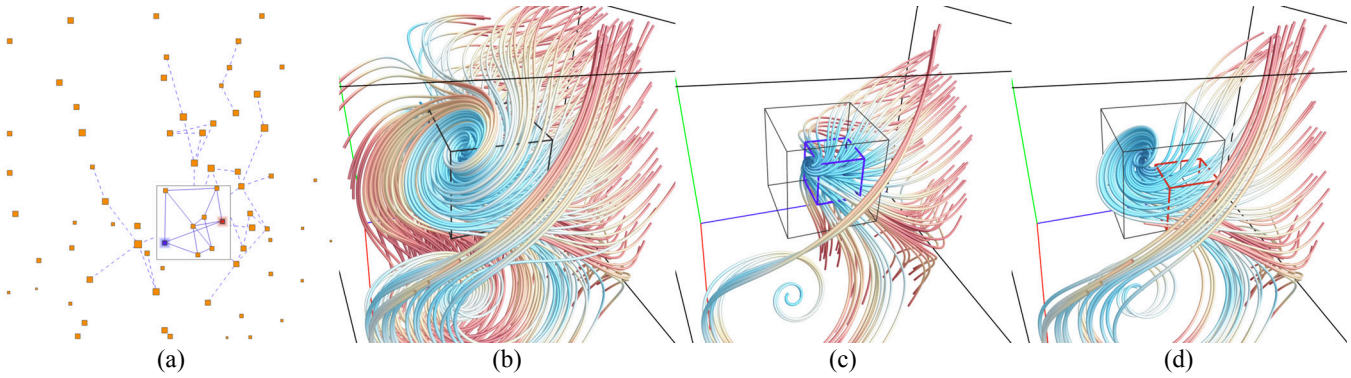

Figure 6: Exploring the five critical points data set using FlowGraph. (a) shows the identification of an important R-node by filtering R-nodes based on the R-R edge weight. (b) to (d) show the streamlines passing through the parent R-node (shown in black) and two child R-nodes (shown in blue and red). Velocity magnitude is mapped to streamline color (low to medium to high mapped to blue to gray to red).

traversed in order, between R-nodes (R-R edges) indicating the number of common streamlines shared, and between L-nodes and R-nodes (L-R edges) indicating their interconnections.

To construct the region hierarchy, we create a spatial partition similar to an adaptive mesh refinement (AMR) grid by considering the flow entropy in each region. To construct the streamline hierarchy, we design a similarity measure based on the regions two streamlines traverse in common as well as the point-wise distance between the two streamlines. We take a bottom-up approach to group streamlines together into an initial set of clusters, then identify a representative from each cluster to form the next level of the hierarchy and so on.

The graph drawing and layout adjustment of FlowGraph follow those of TransGraph. The two types of nodes are displayed in different styles: yellow circles for L-nodes and orange squares for R-nodes. For the underlying graph representation, L-L edges and L-R edges are undirected while R-R edges are directed. For simplicity, we draw a single undirected R-R edge instead of double directed R-R edges. While all L-L edges and L-R edges are used for computing the layout, for R-R edges, we only use edges that across neighboring spatial regions. This prevents the force model from pulling two R-nodes together although they are far away in space.

4.2 Visual Interrogation

FlowGraph contains a wealth of information that can be effectively utilized for visual interrogation. We provide standard interactions such as hierarchical exploration, node filtering, and edge querying. Figure 6 shows an example of hierarchical exploration and interactive filtering of FlowGraph. In (a), only R-nodes and R-R edges are displayed in the graph view. We select an R-node that has a strong connection with its neighbor and the streamlines passing through this R-node are displayed in (b). Since the number of streamlines displayed is fairly large, we further explore the children of this R-node. Two child R-nodes and the streamlines passing through each of them are shown in (c) and (d). Such level-of-detail explorations are very necessary when exploring large and complex 3D flow fields where dense streamlines are commonly exhibited throughout the entire volume.

Furthermore, we enable users to clearly compare streamline clusters in terms of their paths going through different regions, or compare spatial regions in terms of streamline clusters passing through them. For example, for path comparison, users can select two or multiple L-nodes and compare their corresponding streamline clusters' paths by linking their R-nodes in order. An algorithm similar to the maximum spanning tree is used to capture the main structure of the streamline clusters when the path becomes cluttered. In this way, users can compare and track streamline clusters in an occlusion-free manner, which is not possible with the spatial streamline view alone. An example of comparing the paths of two streamline clusters is shown in Figure 7 (a) and (b). These two streamline clusters both start from the volume boundary and get increasingly intertwined toward the center. The highlighted path results also match the spatial arrangement of these two clusters. The two R-nodes shared in common by these two streamline clusters are highlighted in both views.

Figure 7 (c) exhibits an interesting FlowGraph layout of the car flow data set: many L-nodes and R-nodes are pushed to the sides of the drawing area. This is due to the fact that many of the streamlines we trace over the volume only form the straight pattern, i.e., they simply pass by rather than passing through the car. In contrast, L-nodes and R-nodes around the center of the graph correspond to streamline clusters and spatial regions at the center of the volume. They have more connections to their neighboring nodes and are important nodes for our visual exploration. We explore three L-nodes and filter out streamline clusters at two different levels of detail that well capture the flow pattern passing through the car.

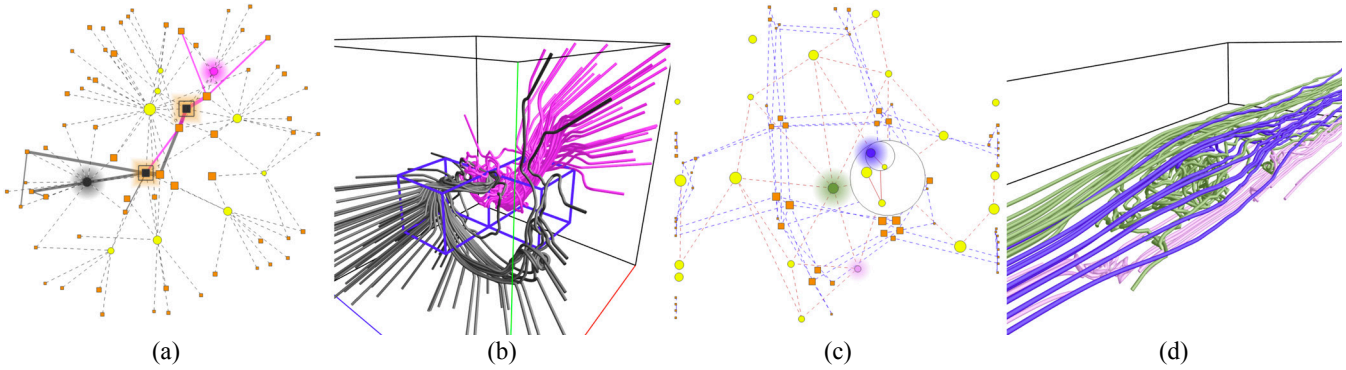

Figure 7: Exploring the supernova and car flow data sets using FlowGraph. (a) and (b) show the path comparison of two highlighted streamline clusters in both views for the supernova data set. Their shared spatial regions are also highlighted. (c) and (d) show the selection of three L-nodes (one at the next level of the hierarchy) of the car flow data set to capture the main flow structure passing through the car.

5 Semantic Flow Graph

The design of semantic flow graph (SFG) is radically different from that of FlowGraph. It mimics the heterogeneous graph from social network analysis and supports dynamic graph reconfiguration, thus giving users the unprecedented flexibility to customize the graph according to their own needs in the process of data exploration and knowledge discovery. SFG is a novel graph representation and interaction framework that enables users to explore the relationships among key objects (i.e., flow lines, critical points, vortex cores, and spatial regions) of a 3D flow field.

5.1 What are Missing in FlowGraph?

Although the FlowGraph design has its value, it suffers from several limitations. First, the underlying graph structure is predetermined and *fixed*, offering no opportunity for *dynamic* graph reconfiguration at runtime. Second, as nodes and edges in FlowGraph do not carry *attributes*, we are not able to encompass various data attributes or relationships and leverage the concept of *ontology* to describe how various node and edge attributes relate to each other. Third, *features* such as critical points and vortex cores in the vector field are not explicitly encoded in the graph, which often entails painstaking trial and error on the user side and hinders *goal-oriented exploration*. As such, the FlowGraph design could not address many "what-if" questions such as "what if we want to investigate both scalar and vector fields in the same graph?", "what if we want to flexibly slice and dice the graph based on different criteria?", and "what if we want to perform feature-driven exploration in both the graph and data spaces?" All these limitations prompt us to reexamine the existing solution and investigate a new, naturally suited graph design for visualizing and exploring multifaceted flow data sets.

5.2 SFG Construction, Drawing, and Exploration

To capture the relationships among different types of objects in a flow field, we define three types of nodes for SFG: L-nodes for streamlines, P-nodes for critical points, and R-nodes for spatial regions. Each type of nodes carries a set of attributes, e.g., average curvature and torsion for L-nodes, types of critical points for P-nodes, and entropy of vector directions and magnitudes for R-nodes, etc. Each node is an aggregation of objects that share similar attribute values (e.g., an L-node may represent streamlines with high curvature and high torsion values) or connect to the same set of nodes (e.g., an L-node may represent streamlines connected to a P-node containing all repelling saddles).

To reveal the relationships of these objects, we consider three kinds of edges: L-P edges between L-nodes and P-nodes, L-R edges between L-nodes and R-nodes, and R-P edges between R-nodes and P-nodes. A streamline and a critical point are connected, if the minimum distance between a point on the streamline and the critical point is smaller than a certain threshold. A streamline and a region are connected, if the streamline passes through the region. A critical point and a region are connected, if the region contains the critical point.

The graph view shows the current SFG, as shown in Figure 8 (a) and (b). The three types of nodes are displayed in different styles: blue circles for L-nodes, orange pentagons for P-nodes, and green squares for R-nodes. The three types of edges are displayed in different colors: blue for R-P edges, orange for L-R edges, and green for L-P edges. We design two graph layouts: a standard layout (Figure 8 (a)) using stress majorization [3] for revealing the connections among all nodes, and an egocentric layout (Figure 8 (b)) for examining the neighborhood of selected nodes.

The exploration of SFG starts with an ontology graph containing an L-node representing all streamlines, a P-node representing all critical points, and an R-node representing all spatial regions. Finer structures in the flow field are

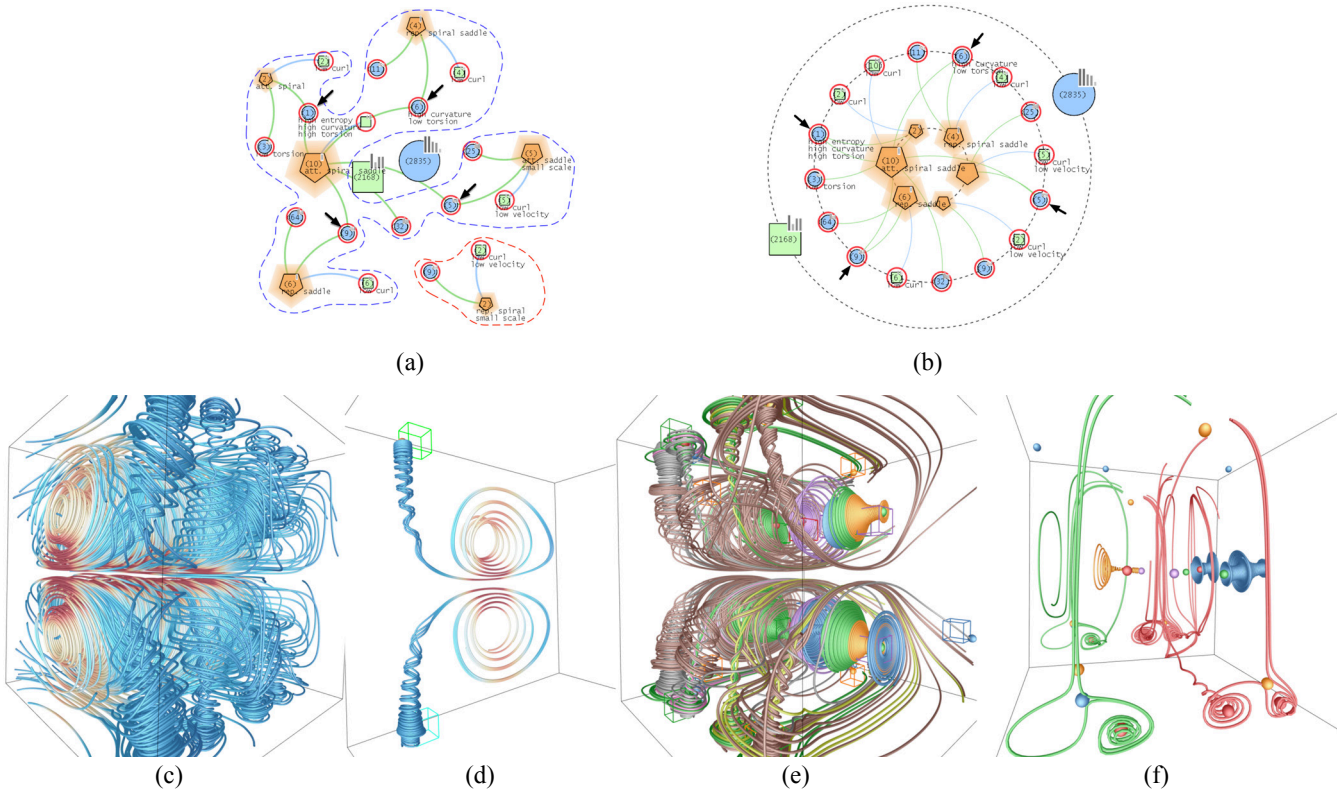

Figure 8: Exploring the two swirls data set using semantic flow graph. (a) shows the neighborhood inspection for all P-nodes where each P-node represents a type of critical points. (b) shows the same inspection result of (a) with the egocentric layout. (c) shows 300 streamlines randomly selected from the entire pool of 3000 streamlines. (d) and (e) show the objects in the red and blue dashed boundaries of (a), respectively. (f) shows the connectors of the P-nodes.

discovered through splitting the nodes in SFG. We provide two types of split: *attribute-based split* and *structure-based split*. Attribute-based split divides a selected nodes into multiple nodes with a user-specified attribute. Each generated node contains the objects with similar values of the specified attribute. In Figure 8 (a), the P-nodes are generated by performing attribute-based split with "type" attribute, so that each P-node contains critical points of a distinctive type. Structure-based split divides all other nodes based their connection to a selected set of nodes. In Figure 8 (a), the L-nodes and R-nodes are generated using structure-based split with all P-nodes selected, so that each L-node or R-node contains objects connected to the same type of critical points.

To investigate node connections, we provide two types of inspection: *connector inspection* and *neighborhood inspection*. Connector inspection identifies the nodes serving as connectors between any pair of selected nodes. In Figure 8 (a), the connectors of all P-nodes are inspected and highlighted by circular percentage bars. Each percentage bar indicates the percentage of connector objects in the highlighted node. Neighborhood inspection reorganizes the nodes according to their connections to a selected set of nodes. In Figure 8 (b), all P-nodes are selected and placed at the center of an egocentric layout. The nodes connected to the P-nodes are placed at the middle layer, and all other nodes are placed at the outer layer.

In Figure 8, we can see in (a) that most streamlines and regions are not related to the critical points, and they form the two largest nodes at the center. In most cases, an L-node or R-node connects to a single P-node. Only four of the L-nodes serve as connectors between the P-nodes, as indicated by the black arrows. We find that the P-nodes form two groups through the L-nodes: two isolated flow structures are highlighted by the red dashed boundaries and the corresponding streamlines are shown in (d), while the other group is highlighted by the blue dashed boundary and the corresponding streamlines are shown in (e). Compared to a randomly selected subset of 300 streamlines in (c), the streamlines in (d) and (e) reveal the more inner structure of the flow field. We can only see the two larger swirling patterns and some smaller ones in (c), but their connections and the small spirals inside the two large swirls can be found in (d) and (e). In (f), we show the streamlines connecting different types of critical points. Their corresponding L-nodes are indicated with arrows in (a) and (b).

6 Concluding Remarks

Our experiments and experiences demonstrate that the graphs we generate are fairly lightweight in terms of storage cost and quite scalable in terms of interaction with hierarchical representation. We conduct the expert evaluation with the domain scientists on TransGraph mining, FlowGraph, and SFG. The evaluation results show that although these graphs abstract data relationships into a transformed space, they are actually easy for training and intuitive to get familiar with. This is because the graph layouts encode data information and their node-link structures reflect the very nature and characteristics of the underlying data. Intuitive understanding can be conveniently gained by users through dual interaction in the original data view and the graph view. SFG, however, is a bit more challenging for users as the learning curve for structure-based split and connector inspection is a little steep, especially for those who are only familiar with traditional visualization techniques. Our work represents a systematic visual analytics approach to exploring scientific data sets. For further information about our work including papers, videos, and code, please visit `http://sites.nd.edu/chaoli-wang/`. We envision this approach to be adopted by the community and integrated into their scientific visualization workflow, thus changing the tools and feature sets we have at hand to perform scientific visualization for big data analytics.

Acknowledgements

The author would like to thank Yi Gu, Jun Ma, and Jun Tao who contributed to the development of graph-based techniques for scientific visualization, and the domain scientists Robert Jacob, Jingfeng Jiang, and Seung Hyun Kim who worked with us in evaluating these techniques. This research was supported in part by the U.S. National Science Foundation through grants IIS-1017935, IIS-1456763, and IIS-1455886; and the Notre Dame Global Collaboration Initiative Program.

Biography

Chaoli Wang is an associate professor of computer science and engineering at University of Notre Dame. His research interests include scientific visualization and visual analytics. Wang received his PhD in computer and information science from The Ohio State University. He is a senior member of the IEEE. Contact him at chaoli.wang@nd.edu.

References

[1] C. Collins, G. Penn, and M. S. T. Carpendale. Bubble Sets: Revealing set relations with isocontours over existing visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1009–1016, 2009.

[2] C. Dunne and B. Shneiderman. Motif simplification: Improving network visualization readability with fan, connector, and clique glyphs. In *Proceedings of ACM SIGCHI Conference*, pages 3247–3256, 2013.

[3] E. R. Gansner, Y. Koren, and S. North. Graph drawing by stress majorization. In *Proceedings of International Symposium on Graph Drawing*, pages 239–250, 2004.

[4] J. Glen and J. Widom. SimRank: A measure of structural context similarity. In *Proceedings of ACM SIGKDD Conference*, pages 528–543, 2002.

[5] Y. Gu and C. Wang. TransGraph: Hierarchical exploration of transition relationships in time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2015–2024, 2011.

[6] Y. Gu, C. Wang, T. Peterka, R. Jacob, and S. H. Kim. Mining graphs for understanding time-varying volumetric data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):965–974, 2016.

[7] J. Ma, C. Wang, and C.-K. Shene. FlowGraph: A compound hierarchical graph for flow field exploration. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 233–240, 2013.

[8] J. Ma, C. Wang, C.-K. Shene, and J. Jiang. A graph-based interface for visual analytics of 3D streamlines and pathlines. *IEEE Transactions on Visualization and Computer Graphics*, 20(8):1127–1140, 2014.

[9] A. Robles-Kelly and E. R. Hancock. String edit distance, random walks and graph matching. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):315–327, 2004.

[10] Y. Sun and J. Han. Mining heterogeneous information networks: Principles and methodologies. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 3(2):1–159, 2012.

[11] J. Tao, C. Wang, N. V. Chawla, L. Shi, and S. H. Kim. Semantic flow graph: A framework for discovering object relationships in flow fields. *IEEE Transactions on Visualization and Computer Graphics*. Accepted.

[12] C. Wang and J. Tao. Graphs in scientific visualization: A survey. *Computer Graphics Forum*, 36(1):263–287, 2017.