This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2773071, IEEE Transactions on Visualization and Computer Graphics



# Semantic Flow Graph: A Framework for Discovering Object Relationships in Flow Fields

Jun Tao, Chaoli Wang, *Senior Member, IEEE,* Nitesh V. Chawla, *Member, IEEE,* Lei Shi, *Senior Member, IEEE,* and Seung Hyun Kim

**Abstract**—Visual exploration of flow fields is important for studying dynamic systems. We introduce semantic flow graph (SFG), a novel graph representation and interaction framework that enables users to explore the relationships among key objects (i.e., field lines, features, and spatiotemporal regions) of both steady and unsteady flow fields. The objects and their relationships are organized as a heterogeneous graph. We assign each object a set of attributes, based on which a semantic abstraction of the heterogeneous graph is generated. This semantic abstraction is SFG. We design a suite of operations to explore the underlying flow fields based on this graph representation and abstraction mechanism. Users can flexibly reconfigure SFG to examine the relationships among groups of objects at different abstraction levels. Three linked views are developed to display SFG, its node split criteria and history, and the objects in the spatial volume. For simplicity, we introduce SFG construction and exploration for steady flow fields with critical points being the only features. Then we demonstrate that SFG can be naturally extended to deal with unsteady flow fields and multiple types of features. We experiment with multiple data sets and conduct an expert evaluation to demonstrate the effectiveness of our approach.

**Index Terms**—Flow visualization, heterogeneous graph, semantic abstraction, critical points, vortex cores, FTLE, field lines.

## 1 Introduction

Effectively displaying field lines (streamlines for steady flow fields and pathlines for unsteady flow fields) faces significant challenges, especially considering the ever-growing size and complexity of flow data generated from scientific simulations. One fundamental challenge is occlusion and clutter, which stems from projecting 3D field lines to 2D screen. To reduce occlusion and clutter so that flow patterns can be depicted clearly, researchers commonly adopt an approach that balances field line densities among different spatiotemporal regions through field lines seeding or selection. However, this approach could only *alleviate* but not *eliminate* the problem, since capturing flow patterns (which demands field lines to pass different flow features) and reducing occlusion and clutter (which calls for less field lines for clarity) are somewhat conflicting with each other. Often, we have to reduce the field lines passing through unimportant regions in order to enhance the visual perception of important regions.

Important field lines or regions are often referred to as features. Although there is no universal definition of features, flow patterns related to critical points are of crucial interest. A *critical point* is a singularity in a flow field where the velocity vanishes. Flows around the vicinity of critical points are often complicated. Without explicitly capturing them using streamlines, we can hardly infer them from their neighboring regions. Therefore, for steady flow fields, a major goal of streamline visualization is to reveal the flow patterns around critical points and investigate the connections among these critical points. For unsteady flow fields, it is also important to discover the temporal development of critical points detected at each time step. Previous approaches were developed to capture these patterns [36] and reveal their connections [28]. However, they usually serve specific purposes, provide very limited interaction support, and fail to meet various exploration needs.

In this paper, we present *semantic flow graph* (SFG), a novel solution that leverages techniques from information visualization and social network analysis to enable semantic-aware exploration of relationships among field lines, features, and spatiotemporal regions. Most early works in scientific visualization focused on extracting features and developing their relationships to construct various graph representations [29], [30]. These representations provide an abstracted overview of the underlying scientific data sets. We take a drastically different direction: instead of focusing on the construction of a *fixed* graph structure or hierarchy, we allow flexible grouping of objects through *dynamic* graph construction and shift our focus to graph exploration. With a rich set of interactions, users are given the unprecedented flexibility to customize the graph according to their own needs in the data exploration and knowledge discovery process.

Our approach is inspired by the exploration of *heterogeneous graphs* [24] in social network analysis. A heterogeneous graph consists of different types of nodes and edges, and each type is associated with a set of attributes. For example, in a bibliographic network, three types of nodes are usually considered along with their attributes: authors with affiliations, papers with topics, and venues with locations. Edges represent the relationships among the nodes (e.g., authoring or co-authoring). Similarly, our approach considers each field line, feature, and region as an *object*, and organizes their relationships as a heterogeneous graph, as shown in Figure 1. Each object in SFG carries attribute information, such as the average curvature values for streamlines, types for critical points, and velocity entropy values for regions. We then utilize these attribute information to aggregate objects of the


- *J. Tao, C. Wang, and N. V. Chawla are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556. E-mail: {jtao1, chaoli.wang, nchawla}@nd.edu.*
- *L. Shi is with the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, Beijing, China 100190. Email: shil@ios.ac.cn.*
- *S. H. Kim is with the Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, OH 43210. E-mail: kim.5061@osu.edu.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2773071, IEEE Transactions on Visualization and Computer Graphics



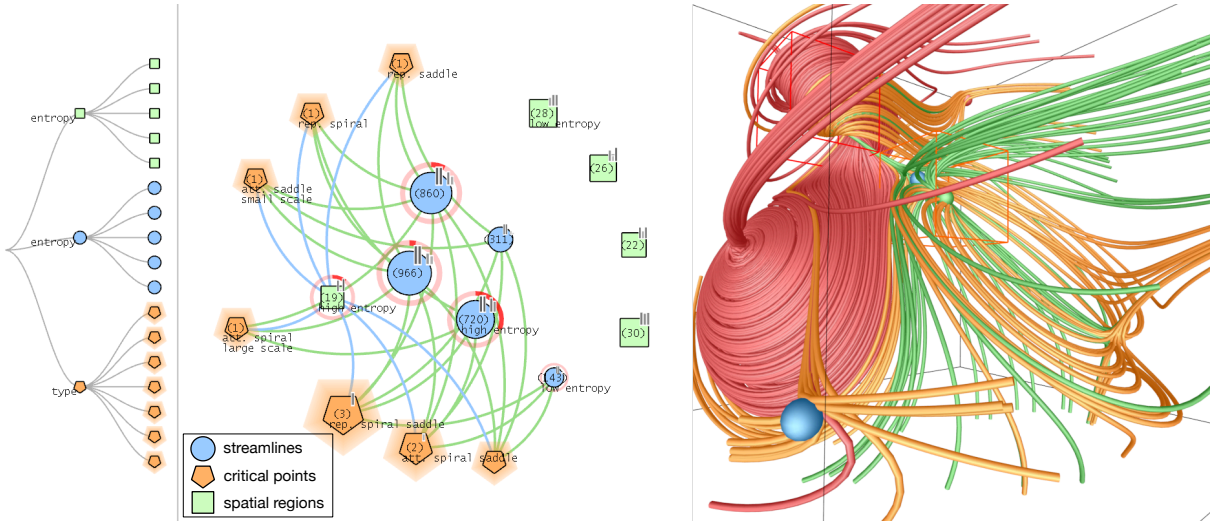



Fig. 1: Semantic flow graph (SFG) interface showing the exploration of the five critical points data set. (a) The tree-like history view. The three children of the pseudo-root are at the highest level of abstraction, each of which corresponds to a type of nodes. The label on a node indicates the criterion used to split the node. (b) The graph view of the current SFG. All types of critical points are selected and highlighted by halos, and their connectors are indicated by red circular percentage bars. The edges indicate connections between selected nodes and their connectors. (c) The corresponding volume view. The critical points and the streamlines connecting them are shown. Different colors of critical points and streamlines indicate different corresponding nodes.

same category into nodes and generate a visual abstraction of the heterogeneous graph. This abstraction not only reduces visual clutter, but also assigns the specific semantics to each node. If needed, the hierarchical information of field lines or regions can be incorporated into this framework by including the cluster index as an attribute in the description of semantics. For simplicity, throughout the paper, we describe the SFG construction and exploration in the context of steady flow fields with critical points being the only features. The construction of the SFG to handle multiple types of features for unsteady flow fields will be explained in details when the specific data set is discussed in Section 5.

We specifically design a suite of operations for SFG to explore flow fields. The operations include two kinds of splits (attribute-based and structure-based) that involve structural changes of the SFG in order to observe features at different levels, and two kinds of inspections (neighborhood and connector) that examine the relationships among nodes under the current configuration. All these operations are performed under the current context of SFG. The split operations allow the nodes to be further divided and their objects to be regrouped. This kind of dynamic construction of SFG provides great flexibility to investigate relationships among objects at different levels and in different ways. As shown in Figure 1, we create three views: a *graph view* (Figure 1 (b)) to show the current SFG, a *history view* (Figure 1 (a)) to present the node split history, and a *volume view* (Figure 1 (c)) to visualize the objects in the original 3D space. Brushing and linking allows users to build the connection among these views.

The contributions of our work are the following. First, to the best of our knowledge, we are the first to introduce the concept of semantic graph for flow visualization from which we propose an interaction framework for dynamic graph construction. With a set of newly-designed operations, we enable semantic-aware knowledge discovery to explore the relationships among field lines, features, and spatiotemporal regions. Second, we design a history view that effectively captures the node split history, so that users can better understand the structure of the current SFG and interact with the graph more conveniently. Finally, we present case studies using multiple data sets and conduct an expert evaluation to demonstrate that a variety of tasks can be performed more efficiently and effectively using our approach.

## 2 Related Work

**Flow Field Exploration Techniques.** Visual exploration of 3D flow fields is an important topic in flow visualization. Most works focused on identifying and locating certain flow pattens. For instance, Heiberg et al. [6] selected a set of structures and located similar patterns in an input vector field. They defined a similarity measure to compare the predefined flow patterns and the local neighborhood of each point in the vector field. Schlemmer et al. [19] utilized invariant moments to describe and compare 2D flow fields, which is invariant under translation, scaling and rotation. They constructed a multiscale moment pyramid to match patterns at different scales. Bujack et al. [1] later introduced a definition of moment invariants for 3D vector fields. These approaches detect flow patterns by directly comparing subdomains of vector fields. Other approaches explore the flow patterns using streamlines. Rössl and Theisel [15] transformed a vector field into a set of 2-manifolds by mapping streamlines to points in $\mathbf{R}^n$ space. Their mapping preserves the Hausdorff distance in the original space and provides a compact visualization of the characteristic flow geometry and topology. Tao et al. [26] defined a similarity measure for streamline segments based on the registration of their respective point sets. The streamline segments are clustered and represented by a set of shape characters so that flow patterns can be queried in a textual manner. Researchers also investigated into sketch-based interface for intuitive flow field exploration using streamlines. Schroeder et al. [20] designed a sketch-based interface for illustrating 2D vector fields. The interface allows the illustrator to draw directly on top of the data for fine tuning of streamline density and length. Wei et al. [33] proposed a similarity

measure based on curvature and torsion along streamlines. Their interface allows users to sketch a 2D curve and match projected streamlines under the current view. They also used agglomerative hierarchical clustering to create streamline templates and support on-the-fly partial streamline matching. Salzbrunn and Scheuermann [16] described streamline predicates that assign boolean values to streamlines with respect to features. Using the predicates, the streamlines can be filtered to reveal structures related to user-selected features.

**Critical Point Detection and Visualization.** Critical points characterize interesting flow patterns in the vector field. Locating and classifying critical points has attracted a significant amount of attention. Scheuermann et al. [18] proposed methods to locate high-order critical points in 2D vector fields using Clifford algebra. Mann et al. [13] presented an octree based solution using geometric algebra for finding critical points in 3D vector fields. Klein and Ertl [8] determined the scales of critical points by tracking their locations in the scale space. The Gaussian scale space is formed by a series of smoothed versions of the original vector field. Other works focused on visualizing flow patterns related to critical points. Löffelmann et al. [9] proposed two representations to visualize flow patterns near critical points. The first one encodes the order of magnitude of eigenvalues of the Jacobian matrix along the characteristic trajectories, and the second one stochastically seeds streamlets on a small sphere around critical points. Ye et al. [36] designed different seeding templates for different types of critical points. The shape of template can change depending on the degree of a critical point transitioning from one type into another. Theisel et al. [28] presented saddle connectors that visualize the topological skeleton of vector fields. Iconic representations are used for critical points, and the specific streamlines connecting different critical points are displayed.

**Heterogeneous Graph Exploration.** Visual exploration of heterogenous graphs is well studied in information visualization. Wattenberg [32] introduced PivotGraph, an attribute-centric node-link visualization of heterogeneous graphs. PivotGraph uses a roll-up operation to aggregate the nodes according to one or two user-specified attributes. The aggregated nodes are placed at 2D grid points according to the attribute values. Shen et al. [21] presented OntoVis that considers both semantic abstraction and structural abstraction based on the ontology graph of the heterogeneous network. The network is filtered by selecting a type of nodes in the ontology graph, and other nodes become attributes of selected nodes. OntoVis further applies structural abstraction to prune the graph for clear observation. Shneiderman and Aris [23] designed semantic substrates to place nodes on multiple non-overlapping regions based on their attributes. Users can use sliders to adjust the visibility of nodes for clutter reduction. Shi et al. [22] proposed OnionGraph that groups the nodes in a top-down manner using both topology and attribute information. Structural equivalence is used to simplify the graph, and multiple iconic symbols are designed to represent the semantic information of nodes.

**Graph-based Flow Field Exploration.** Closely related to our work are two graph-based exploration techniques for flow visualization: flow web presented by Xu and Shen [35] and FlowGraph presented by Ma et al. [11], [12]. In flow web, each node represents a region in the domain and the weight of an edge between two nodes indicates the number of particles traveling between the two regions. Similar representations have been used for workload estimation in parallel and out-of-core streamline tracing [2]. FlowGraph further adds one more type of nodes (i.e., L-node for streamline cluster), and two types of edges (i.e., L-L edge between two L-nodes, and L-R edge between an L-node and an R-node). Therefore, FlowGraph provides a more complete picture than flow web, as it allows users to explicitly investigate the relationships among streamline clusters as well. However, these two approaches only emphasize the overall structure of flow fields. The interactions are constrained to splitting nodes in the precomputed hierarchy and exploring through brushing and linking between the graph view and volume view. These fixed graph structures could not be changed to meet different needs, e.g., queries of streamlines or regions with certain properties, not to mention the investigation of their relationships. Moreover, these approaches do not explicitly capture and encode critical points. For detailed comparison between our SFG and these closely related approaches, please refer to Section 5.2.

## 3 Semantic Flow Graph

In many applications of information visualization and visual analytics, semantic abstraction has been shown to be an effective means for visualizing heterogeneous graphs. In this section, we briefly introduce the concept of heterogeneous graphs and their semantic abstraction. Then we present our graph construction and discuss the operations designed for flow field exploration.

We study three types of fundamental *objects* in flow fields: field lines, features, and spatiotemporal regions. Each node is associated with semantic information named *attributes*, and the connections among nodes are captured in a *heterogeneous graph*. Semantic flow graph (SFG) is a semantic abstraction of such a heterogeneous graph. To distinguish the similar concepts, throughout this paper, we use "object" to refer to a field line, a feature, or a region, and "node" to refer to an aggregation of objects using a certain user-specified attribute. Similarly, "link" is used to indicate that one object is related to another, and "edge" is used to indicate the connection between two nodes. In other words, "object" and "link" are concerned with the underlying heterogeneous graph, which is mainly used for computational purpose; while "node" and "edge" are concerned with the visualization of SFG, i.e., the semantic abstraction being displayed and interacted with.

### 3.1 Terminology and Notation

Formally, we define a *heterogeneous graph* as $G_H = (O, L)$, where $O = \{o_1, \ldots, o_{|O|}\}$ is the set of all objects and $L = \{l_1, \ldots, l_{|L|}\}$ is the set of all links between objects. Each object $o_i \in O$ has a type and a set of attribute values associated with it, denoted by $T(o_i)$ and $A(o_i) = \{a_1(o_i), \ldots, a_{|A(o_i)|}(o_i)\}$, respectively. Note that the number of attributes may vary for different types of objects. We denote the *semantic abstraction* of $G_H$ as $G_S = (V, E)$, where $V = \{v_1, \ldots, v_{|V|}\}$ denotes a node set and $E = \{e_1, \ldots, e_{|E|}\}$ denotes an edge set. Every node $v_i \subset O$ is a semantic aggregation of objects of the same type based on the values of some selected attributes. An edge $e_{ij}$ exists between two nodes $v_i$ and $v_j$ if any of their objects are linked, i.e., $\exists\ o_r \in v_i$ and $o_s \in v_j,\ l_{rs} \in L$. We define the *ontology graph* associated with $G_H$ as $G_O = (T_V, T_E)$, where $T_V = \{tv_1, \ldots, tv_{|T_V|}\}$ is the set of node types and $T_E = \{(tv_i, tv_j) \mid tv_i, tv_j \in T_V\}$ is the set of edge types. The ontology graph is the highest-level semantic abstraction of a multivariate graph, by considering each node type $tv_i$ as an aggregation of all its objects, i.e., $O(tv_i) = \{o \mid o \in O, T_V(o) = tv_i\}$, where $O(tv_i)$ represents all objects of type $tv_i$.

Another way to interpret the relationships among $G_H$, $G_S$ and $G_O$ is that $G_H$ and $G_O$ are *fixed* graphs which represent the

| notation | explanation |
|---|---|
| $G_H$ | heterogeneous graph |
| $G_S$ | semantic graph |
| $G_O$ | ontology graph |
| $O(v_i)$ | all objects in a node $v_i$ |
| $O(\tilde{V})$ | all objects in a set of nodes $\tilde{V}$ |
| $V(o_i)$ | the corresponding node of an object $o_i$ |
| $V(\tilde{O})$ | the corresponding nodes of a set of objects $\tilde{O}$ |
| $N(o_i)$ | all neighboring objects of an object $o_i$ |
| $N(\tilde{O})$ | all neighboring objects of a set of objects $\tilde{O}$ |
| $N(v_i)$ | all neighboring nodes of a node $v_i$ |
| $N(\tilde{V})$ | all neighboring nodes of a set of nodes $\tilde{V}$ |

TABLE 1: Key notations used in this paper.

graph data at the finest and coarsest levels of detail respectively, while $G_S$ is a *dynamic* graph which represents a certain level of detail in between $G_H$ and $G_O$. Therefore, semantic exploration becomes the key to generate meaningful forms of $G_S$ under various considerations.

We also define the following notations. The objects in a node $v_i$ is denoted as $O(v_i)$, and all objects in a set of nodes $\tilde{V}$ is denoted as $O(\tilde{V}) = \cup_{v_i \in \tilde{V}} O(v_i)$. Each object $o_i$ belongs to only one node $v_j$ during the aggregation, which is denoted as $V(o_i) = v_j$. For a set of objects $\tilde{O}$, we denote the corresponding nodes as $V(\tilde{O}) = \cup_{o_i \in \tilde{O}} V(o_i)$. We denote the neighbors of an object $o_i$ as $N(o_i) = \{o_j \mid o_j \in O \text{ and } l_{ij} \in L\}$, and the neighbors of a set of objects $\tilde{O}$ as $N(\tilde{O}) = \cup_{o_i \in \tilde{O}} N(o_i) \setminus \tilde{O}$. Note that $N(o_i)$ or $N(\tilde{O})$ may include objects from different types. Similarly, the neighbors of a node $v_i$ and the neighbors of a set of nodes $\tilde{V}$ are denoted as $N(v_i)$ and $N(\tilde{V})$, respectively. For easy reference, we summarize the key notations in Table 1.

### 3.2 SFG Construction

In this section, without loss of generality, we describe the construction of SFG in the context of steady flow fields with critical points being the only features. Specifically, we use three types of objects in the SFG: streamlines, critical points, and spatial regions. In certain application domains, additional types of features can be flexibly included in the SFG by appropriately linking them to the existing objects. In addition, SFG can be naturally extended to handle unsteady flow fields by replacing streamlines with pathlines and adding an additional "time step" attribute to regions and features. To understand the temporal evolution of flows, we further incorporate two well-accepted tools: finite-time Lyapunov exponents (FTLE) and feature flow fields [27]. These extensions can be established under the same construction described here with slightly different configurations (node types and attributes). In Section 5.1, we discuss a specific configuration when the related data set and case study are described. Corresponding to the three types of objects for steady flow fields, we define three kinds of nodes for SFG:

- *L-nodes*: An L-node is an aggregation of streamlines grouped by user-specified attributes. For a streamline, we consider the following attributes: average curvature, average torsion, entropy, and cluster index. We consider two distance measures for streamline clustering: the mean of the closest point (MCP) distances and the Fréchet distance. The entropy of a streamline is computed using the velocities at all sample points along the streamline, considering both velocity directions and magnitudes.
- *P-nodes*: A P-node is an aggregation of critical points. We consider two attributes associated with a critical point: scale and type. The scale of a critical point indicates its size of influence. In 3D, there are nine types of critical points determined by the Jacobian matrix at one point [36]: sink, source, center, attracting spirals, repelling spirals, attracting saddles, repelling saddles, attracting spiral saddles, and repelling spiral saddles.
- *R-nodes*: An R-node is an aggregation of spatial regions. A spatial region has the following attributes: average velocity magnitude, average divergence, average curl magnitude, and entropy. The entropy of a region is computed using the velocities at all grid points inside the region. Moreover, if scalar fields are associated with the vector field, we further include the average of those scalar values in the region as additional attributes.

Different types of objects are linked according to certain simple criteria. An edge exists between two nodes, if at least one link exists between their objects. The weight of an edge is the sum of weights of all links between objects in the two incident nodes. Note that since we only define the relationships among different types of objects, there is no edge between nodes of the same type. We define three kinds of edges for SFG:

- *L-P edges*: An L-P edge is formed between an L-node and a P-node, indicating that one or more streamlines in this L-node are linked to certain critical points in the P-node. The link between a streamline and a critical point is determined by the minimal Euclidean distance from any sample point on the streamline to the critical point. If this minimal distance is smaller than a predefined threshold, the streamline and the critical point are considered to be linked. Note that we do not restrict our attention to the end points of streamlines, as streamlines passing the neighborhood of saddles do not terminate at the critical points. The link weight is given by subtracting the distance from the predefined threshold, so that the weights are non-negative and a larger weight corresponds to a smaller distance.
- *L-R edges*: An L-R edge is formed between an L-node and an R-node, indicating that one or more streamlines in the L-node pass through certain regions in the R-node. A streamline is linked to a spatial region if it passes through that region, i.e., any of its sample points falls into the region. This link is a boolean relationship and we assign a constant weight for it. The actual number of sample points on a streamline falling into the region is not taken into consideration.
- *R-P edges*: An R-P edge is formed between an R-node and a P-node, indicating that one or more regions in the R-node contain certain critical points in the P-node. Similar to the links between streamlines and regions, this kind of link is boolean and thus carries a constant weight.

### 3.3 SFG Operations

We design a suite of operations to explore the relationships among objects. When a node is selected, users can perform *attribute-based split* based on a specified attribute. When a set of nodes are selected, users can perform *structure-based split* that splits all other nodes according to the connections among their objects and the selected nodes. They can also perform *neighborhood inspection* that finds all nodes connecting to any of the selected nodes, or *connector inspection* that finds all nodes connecting to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2773071, IEEE Transactions on Visualization and Computer Graphics



any two of the selected nodes. All operations are performed to the nodes that currently exist in the SFG.

**Attribute-based Split.** Given a selected node $v_i$, attribute-based split divides it based on a user-specified attribute $a_j$. For an attribute whose value set is a range of real numbers, we discretize it into five categories labeled as "high", "medium-high", "medium", "medium-low", and "low" so that it can be enumerated. Let $\{a_{j1},\ldots,a_{jp}\}$ be all possible values of attribute $a_j$ over the entire data set. This operation produces a set of nodes $\{v_{i1},\ldots,v_{ip}\}$, where $O(v_{ik}) = \{o \in O(v_i) \mid a_j(o) = a_{jk}\}$. After the split, $\{v_{i1},\ldots,v_{ip}\}$ will be added into $G_S$, and $v_i$ will be removed. Note that if a new node $v_{ik}$ does not contain any object, that node will not be added. The edges will be updated accordingly: all edges connecting $v_i$ will be removed, and the edges between the previously existing nodes and the set of new nodes will be computed and added into SFG. Each newly-added node contains objects with similar attribute values, so that users can further investigate into the impact of those attributes. For example, users may split the P-node in the ontology graph according to the types of critical points. Each new node added contains critical points of the same type. The resulting graph will help users further study the relationships among different types of critical points in relation to other objects.

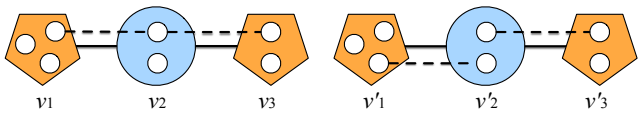
**Structure-based Split.** Given a set of selected nodes $\tilde{V} \subset V$, structure-based split divides each of the other nodes in $G_S$ by partitioning its objects according to the concept of structural equivalence [10]. Two nodes $v_i$ and $v_j$ are *structurally equivalent*, if they have exactly the same neighbor set, i.e., $N(v_i) = N(v_j)$. In our scenario, we consider structural equivalence for two objects $o_i$ and $o_j$ with respect to the selected node set $\tilde{V}$, i.e., $V(N(o_i)) \cap \tilde{V} = V(N(o_j)) \cap \tilde{V}$. Let $\mathbb{P}(\tilde{V}) = \{\mathbb{P}(\tilde{V})_1,\ldots,\mathbb{P}(\tilde{V})_p\}$ be the power set of $\tilde{V}$. For each node $v_i \notin \tilde{V}$, this operation produces a set of nodes $\{v_{i1},\ldots,v_{iq}\}$, where $O(v_{ik}) = \{o \in O(v_i) \mid V(N(o)) \cap \tilde{V} = \mathbb{P}(\tilde{V})_k\}$. Similar to attribute-based split, the non-empty nodes and their corresponding edges will be updated. With this operation, we allow users to observe which objects are connected to individual nodes, and which objects are shared in the neighborhood of different nodes. For example, users may select two P-nodes and perform structure-based split. From the resulting graph, they will be able to compare the streamlines connecting to each P-node only, and the streamlines connecting to both.

**Neighborhood Inspection.** Given a set of selected nodes $\tilde{V} \subset V$, neighborhood inspection finds their neighbors $N(\tilde{V})$. For each neighboring node $v_i \in N(\tilde{V})$, we draw a circular percentage bar to indicate the percentage of objects in $v_i$ that are connected to any object in $O(\tilde{V})$. We define the neighborhood percentage as

$$NP(v_i) = \frac{|O(v_i) \cap N(O(\tilde{V}))|}{|O(v_i)|}. \quad (1)$$

We highlight the edges between selected nodes and their neighbors. To better understand the relationships among the selected nodes and the other nodes, we further arrange all nodes in an egocentric layout, with the selected nodes placed at the center. Other nodes are placed on concentric circles at different layers according to their distances to the selected nodes. We compute the distance based on the underlying heterogeneous graph $G_H$, instead of the current semantic graph $G_S$ being displayed. Let $d_{G_H}(o_p, o_q)$ be the graph's geodesic distance between two objects $o_p$ and $o_q$. We define the distance between nodes $v_i$ and $v_j$ as

$$d_{G_H}(v_i, v_j) = \min_{o_p \in O(v_i), o_q \in O(v_j)} d_{G_H}(o_p, o_q). \quad (2)$$




(a) (b)

Fig. 2: (a) An L-node connector $v_2$. One streamline in $v_2$ connects two critical points in the two P-nodes $v_1$ and $v_3$. (b) An L-node $v'_2$ connects to P-nodes $v'_1$ and $v'_3$, but $v'_2$ is not a connector. No connection exists between critical points in $v'_1$ and $v'_3$ through a streamline in $v'_2$.

Similarly, the distance between a node $v_i$ and a set of selected nodes $\tilde{V}$ is defined as

$$d_{G_H}(v_i, \tilde{V}) = \min_{o_p \in O(v_i), o_q \in O(\tilde{V})} d_{G_H}(o_p, o_q). \quad (3)$$

Note that we do not use the distance based on $G_S$, since it may provide misleading information. For example, consider the simple heterogeneous graph with three nodes shown in Figure 2 (b). $v'_2$ connects to $v'_1$ and $v'_3$, since one streamline in $v'_2$ connects to a critical point in $v'_1$, and the other connects to another critical point in $v'_3$. However, none of these two streamlines links to critical points in both $v'_1$ and $v'_3$, That is, starting from any critical point in $v'_1$, we cannot follow a streamline in $v'_2$ to reach any critical point in $v'_3$, and vice versa. Therefore, $d_{G_H}(v'_1, v'_3) = \infty$ while $d_{G_S}(v'_1, v'_3) = 2$. As a contrast, for Figure 2 (a), we have $d_{G_H}(v_1, v_3) = d_{G_S}(v_1, v_3) = 2$.

**Connector Inspection.** A connector is a node that serves as a bridge by building the connection between two selected nodes. Given a set of selected nodes $\tilde{V} \subset V$, connector inspection finds all connectors $v_i$ that connect any two nodes $v_j$ and $v_k$ in $\tilde{V}$. We define the set of connectors as

$$C(\tilde{V}) = \{v_i \in N(\tilde{V}) \mid \exists\, v_j, v_k \in \tilde{V}, d_{G_H}(v_i, v_j) = d_{G_H}(v_i, v_k) = 1\}. \quad (4)$$

Similar to neighborhood inspection, connectors are determined based on $G_H$ and then mapped back to the nodes in $G_S$ for visualization. We highlight the edges between selected nodes and their connectors. Note that $C(\tilde{V}) \subset N(\tilde{V})$. For each connector $v_i \in C(\tilde{V})$, we draw a circular percentage bar to indicate the percentage of objects that connect to objects in at least two different selected nodes $v_j$ and $v_k$. We compute the connector percentage as

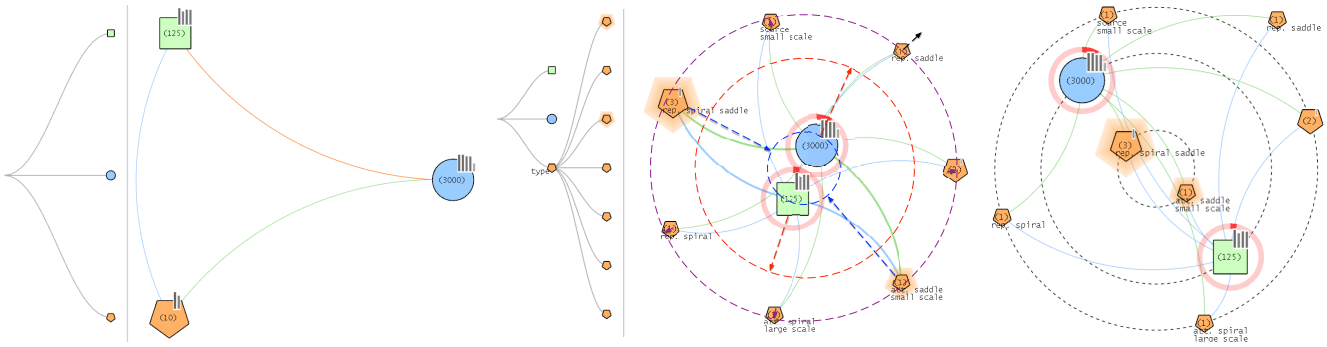
$$CP(v_i) = \frac{|\{o \in O(v_i) \mid \exists\, v_j, v_k \in \tilde{V}, o \in N(O(v_j)) \cap N(O(v_k))\}|}{|O(v_i)|}. \quad (5)$$

This operation helps users understand the relationships among selected objects through other objects, which is especially beneficial when direct relationship is difficult to define. For example, two critical points can be connected by a set of streamlines; and two streamline clusters can be connected by a common region. In Figure 2, $v_2$ shown in (a) is a connector but $v'_2$ shown in (b) is not. For the same reason, connectors are computed at the object level.

Note that connectors may not exist for every pair of selected nodes. To identify the most prominent connectors that connect pairs of selected nodes, we further filter these node pairs by their strength. We define the *connection strength* between a pair of nodes $v_j$ and $v_k$ as the number of objects through which they can be connected, i.e., $|N(O(v_j)) \cap N(O(v_k))|$. If the strength is smaller than a user-specified threshold, the objects connected to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2773071, IEEE Transactions on Visualization and Computer Graphics






(a) (b) (c)

Fig. 3: Node split and neighborhood inspection. (a) The initial ontology graph. (b) The splitting of the P-node using the "type" attribute. Two nodes are selected for neighborhood inspection. The dashed circles and arrows indicate node movements for generating the egocentric layout. (c) The resulting egocentric layout of the same SFG shown in (b).

only $v_j$ and $v_k$ will be removed from connector inspection results. Furthermore, a node will be removed from the queried nodes $\tilde{V}$, if its connection strength to any other node in $\tilde{V}$ is smaller than the threshold. Using this filtering function, users can first select a relatively large set of nodes to query the connectors, and then gradually narrow down to the significant ones.

Since our visualization displays the connections among all queried nodes in $\tilde{V}$, it is impossible for users to distinguish connectors between different pairs of nodes. We further provide *connector iterator* to go through each of the node pairs in descending order of their strength. Users can use the mouse wheel to switch back and forth among all the node pairs.

## 4 Visualization and Interaction Design

Our SFG interface consists of three views: the graph view, history view, and volume view. All these three views are connected through brushing and linking: when users interact with one view, the other two views will be dynamically updated. In this section, we describe the design of each view and the interactions provided in the context of steady flow fields.

### 4.1 Graph View

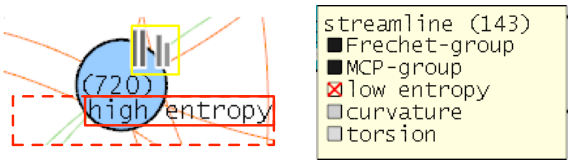
The graph view shows the current SFG, as shown in Figure 3. The three types of nodes are displayed in different styles: blue circles for L-nodes, orange pentagons for P-nodes, and green squares for R-nodes. The three types of edges are displayed in different colors: blue for R-P edges, orange for L-R edges, and green for L-P edges. We design two node highlighting schemes: the halos are used to indicate the selected nodes, while the circular percentage bars are used to highlight the neighbors (connectors) when neighborhood inspection (connector inspection) is performed. To better represent the nodes at different graph distances to the selected nodes, an *egocentric layout* is used in neighborhood inspection; while in other cases, a *standard layout* is used to reveal the connections among all nodes.

**Standard Layout.** For the standard layout, we implement the algorithm presented by Gansner et al. [4] using stress majorization. This algorithm determines node positions in the 2D screen space by preserving their graph-theoretical distances. To better maintain the user's mental map when SFG changes, we further develop an incremental layout update. The basic idea is to introduce an additional energy term that forces the previously existing nodes to stay close to their original positions after an update. We refer interested readers to online graph drawing approaches [3], [14] for more sophisticated solutions.

**Egocentric Layout.** The egocentric layout applies when neighborhood inspection is performed. This layout provides a guidance for users to determine how to further discover the relationships among the selected nodes and other nodes. It aligns the nodes at different layers of concentric circles according to their distances to the selected nodes (Equation 3). The selected nodes are placed at the innermost layer, their 1-hop neighbors are placed at the second layer, 2-hop neighbors are placed at the third layer, and the rest of nodes are placed at the outermost layer. The 1-hop neighbors contain the objects directly related to the selected nodes while the 2-hop neighbors are candidates to inspect connectors between them and the selected nodes. For the rest of nodes, their relations to the selected nodes are often not interesting, since they are not directly related to the selected nodes, and no connector can be found between any of them and the selected ones. For example, in Figure 3 (c), we can see that both the R-node and the L-node are directly connected to the two types of critical points. No other types of critical points are in the 1-hop neighborhood, as no direct connection exists between critical points in our graph definition. But connectors exist between four types of critical points and the two selected types. Only one type of critical point cannot be connected to the selected nodes through streamlines or regions, and it is placed at the outermost layer. We use dashed circles to indicate different layers of nodes. However, we do not draw a dashed circle explicitly for the outermost layer. This is to avoid the possible misunderstanding that those nodes have the same distance to the selected nodes.

Starting from the original layout, we generate the egocentric layout by moving the nodes to the desired circle along the direction connecting itself and the center of the circle, as shown in Figure 3 (b). To avoid clutter, the nodes are placed evenly on the circle, while preserving their relative order along the circle. This maintains a smooth transition from the original layout to the egocentric layout. For example, in Figure 3 (b), the four nodes in the 2-hop neighborhood move along the purple arrows to the purple circle. For a more balanced layout, the nodes are evenly spaced on the circle, while preserving their relative order along the circle, as shown in Figure 3 (c).

**Node Labels.** For large enough nodes, we display labels to provide their basic information, as shown in Figure 4 (a). Each label contains the number of objects in the node, and the associated semantic information. The number of objects is always displayed




(a) (b)

Fig. 4: Node label. (a) The node with text label and split guidance for attributes. (b) The detailed label information of the same node when the node is hovered.

at the center of the node, if not overlapped with other labels. With the limited screen space, only the most important semantic information is displayed, including the types of critical points, and the low or high values for scalar attributes. To place the label of a node, we first compute its size, as indicated by the red rectangle in Figure 4 (a). Then, a compact neighborhood (the red dashed rectangle) below the center of the node is searched to place the label, so that it will not overlap with existing labels or exceed the boundary. The labels are placed in descending order of node size. A label will not be displayed if no location that fulfills the requirement can be found in the neighborhood.

When a node is hovered, we show its label with detailed information including the node type, the number of objects within, and all associated semantic information. If the objects in the node share the same value category for an attribute, a predefined name for that value will be displayed (e.g., "low entropy" in Figure 4 (b)). Otherwise, only the attribute name will be shown, indicating that the node can be further split using that attribute.

**Split Guidance.** Even the number of nodes can be greatly reduced after semantic abstraction, the cognitive efforts for users could still be huge as they need to decide how to split the nodes and expand the graph. Following the idea of scented widgets [34], we provide split guidance which gives navigation cues to users about the amount of information added when splitting a node $v$ using each attribute. Assume that $\{v_1,\ldots,v_q\}$ are the nodes generated by splitting a node $v$ according to some attribute $a$, the added information is measured by the entropy of object distribution over the newly-generated nodes. Specifically, let $p_i$ be the probability of an object in $v_i$ after the split, given by $p_i = |v_i|/|v|$. The entropy of splitting node $v$ using attribute $a$ is computed as $H(v,a) = -\sum p_i \log p_i$.

We design two ways to visualize split guidance. First, it is drawn as a bar chart at the upper-right corner of a node, as shown in the yellow rectangle of Figure 4 (a). The size of the bar chart is proportional to the radius of the node. The bar chart will not be displayed if it is too small. Second, it is indicated by multiple grayscale squares on the left side of the label when the node is hovered, as shown in Figure 4 (b). The black (white) implies that the entropy is high (low). A box with a red cross indicates that all objects in the node share the same attribute value category, so the node cannot be further split according to that attribute.

**Edge Filtering.** The SFG can be dense and cluttered, since two nodes are connected if any two of their objects are linked. This hinders the observation of important connections. Therefore, we introduce two types of edge filtering to reduce the number of edges displayed. First, users may manually specify a threshold for each type of edges to indicate the maximum number of edges to display, or choose to completely hide a type of edges. Edges of each type are ordered by their weights to ensure that the most important connections will be visible. By default, we display 20 edges with the largest weights. Second, we automatically determine the edge types of interest based on the operation being performed. Specifically, when a node is split or neighborhood inspection is performed on a set of nodes of the same type, we only display the edges related to this type. For example, when an L-node is split or the neighborhood of a set of L-nodes is inspected, we consider the connections between the L-nodes and the other two types of nodes to be more interesting, and the connections between P-nodes and R-nodes to be less interesting. In this case, we will only display L-P edges and L-R edges.

**Graph Exploration.** The exploration of SFG always starts with the ontology graph, as shown in Figure 3 (a). Users may select a node and an attribute to perform attribute-based split. For example, we split the P-node using the "type" attribute, and the graph is updated accordingly. Users may select a set of nodes to perform other operations. If structure-based split is performed, we update the SFG by highlighting the edges connecting the new nodes and the selected nodes, so that their relationships are easy to track. If neighborhood inspection is performed, we highlight the neighbors with circular percentage bars and rearrange all nodes into the egocentric layout. If connector inspection is performed, we highlight the connectors with circular percentage bars but the layout stays the same for stable viewing. For the two inspection operations, the volume view will also be updated to show the objects connected to the selected nodes. After each operation, if the graph layout is updated, we show an animated transition that gradually transits from the previous layout to the new one.

### 4.2 History View

The history view visualizes the node split history as a tree. We display different types of nodes following their styles shown in the graph view, but in the same size to keep the tree visually organized. Initially, a tree corresponding to the ontology graph is shown. This tree has a pseudo-root with three children representing the three types of nodes. After each split, the newly-generated nodes will be added into the tree as the children of the node being split. The criterion to split a node will be displayed: if a node is split based on an attribute, the name of that attribute will be shown; if a node is split based on structure, we display "structure" on that node. The children of an internal node are ordered according to the split criterion as well. Specifically,

- if the internal node is split based on a *nominal* attribute (e.g., the type of critical points or the cluster index of streamlines), the children are arranged in descending order of their numbers of objects, with the child containing the most number of objects placed at the top;
- if the internal node is split based on an *ordinal* attribute, the children are arranged in descending order of that attribute's category, with the child containing objects with the highest attribute values placed at the top; and
- if the internal node is split based on structure, the children are ordered by the lexicographic order of their connecting subsets, with the child connecting to all of the selected nodes placed at the top.

Note that all the leaves in this tree are the nodes of the current SFG. The history view allows users to track, step by step, how the nodes are split from the ontology graph to form the current SFG.

**Focus+Context Visualization.** We implement a focus+context lens using a one-dimensional fisheye view [17] to stretch different regions vertically when users mouse over nodes in the history

view. Usually, the height of the tree along the horizontal direction is not expected to be large, otherwise the nodes could be trivial with too many split operations. Therefore, visual clutter is most likely to happen around the leaves overlapping each other vertically. If the minimum distance between any two leaves is smaller than the node size, we will enable the lens and enlarge the focal region centered at the $y$-coordinate of the mouse position.

**Interactions.** Selecting or splitting a leaf in the history view has the same effect as that operation in the graph view. Selecting an internal node will select all the leaves of the subtree rooted at that node. This provides an efficient way to select a group of nodes in many scenarios. In addition, users can specify a set of nodes to be hidden in the graph view. The hidden nodes will be shown in gray in the history view. This simplifies the graph so that users can focus on the relationships of interest. The hidden nodes can be displayed again in the graph view at any time. Another important feature of the history view is that the split operations can be reverted by merging all descendants back to a single node. The order to merge previously split nodes is flexible, i.e., users do not need to follow exactly the reverse order of split operations to cancel each of them. Finally, users may split an internal node using other criteria as well. The effect is the same as first merging back to that internal node and then splitting it again.

### 4.3 Volume View

The volume view displays streamlines, critical points, and spatial regions in 3D. They are drawn as tubes, spheres, and wireframe boxes, respectively. If only objects in one node are selected for a node type, the objects are colored according to the type. Specifically, for streamlines, the color on a point along a streamline indicates the velocity magnitude at this point (see Figure 5 (d)); for critical points, we use the same orange color as that of P-nodes in the graph view; and for regions, their colors indicate the corresponding velocity entropies. If objects in multiple nodes are selected, we assign different colors to objects of different nodes, so that they can be distinguished.

In addition, users can select any streamline, critical point, or spatial region in 3D, and link it back to the graph view and history view. Since the objects displayed in the volume view are already selected in the other two views, this allows users to further filter the previous selection or inspection results. The selection is performed at the node level, similar to the graph view. To start the selection in the volume view, users can click on "select streamlines", "select critical points", or "select regions" from the pop-up menu. When nodes of one type is selected, objects in the other two types of nodes will be deemphasized, so that users can focus on the selected objects, and still receive the contextual information of objects in other types. For example, when users select critical points, the radius of streamline tubes will shrink, and the line width for region wireframes will decrease. Users can select a node by click on any of its object in the volume view. Multiple nodes can be selected before clicking "finish selection". The objects in the selected nodes will remain in the volume view, with other objects removed.

## 5 Results

In this section, we present five case studies with SFG followed by a comparison of the features/tasks supported by the SFG and previous approaches. The case studies are: finding the connections among critical points (Case Studies 1 and 2), discovering the relationships between flow fields and associated scalar volumes (Case Study 3), discovering the relationships between streamlines, critical points, and vortex cores (Case Study 4), and revealing the evolution of flow and features (Case Study 5). For all the case studies, we produce 3000 field lines and use two voxels as the distance threshold to determine whether a critical point and a field line is connected. The grid resolution is determined by the cell size. We use a cell size of $5 \times 5 \times 5$ for most of the data sets, except the five critical points data set ($10 \times 10 \times 10$) and the combustion data set ($20 \times 20 \times 20$).

### 5.1 Case Studies

**Case Study 1: Five Critical Points.** We explore the five critical points data set and demonstrate the results in Figure 1. This data set is simulated by randomly placing five critical points in the domain, including two spirals, two saddles, and one source. Starting from the ontology graph, we split the R-node using the "entropy" attribute, the L-node using the "entropy" attribute, and the P-node using the "type" attribute, respectively. These split operations are recorded and demonstrated in (a), and the SFG produced by applying these operations is shown in (b). From the labels of P-nodes in (b), we find that ten critical points are detected instead of five. This implies that additional critical points are generated during the simulation. The connections among these critical points are revealed by connector inspection with all the P-nodes selected. From the percentage bars in (b), we can see that the different types of critical points are mostly connected through high-entropy streamlines and regions. This is expected, since streamlines associated with the critical points normally present more complex flow patterns. The original 3D streamlines and regions connecting the critical points are shown in (c), which provides us the details on how the critical points are connected and helps to confirm that the information revealed by the SFG is accurate and reliable.

**Case Study 2: Two Swirls.** Our exploration of the two swirls data set is shown in Figure 5. The two swirls data set contains two major swirling patterns together with several smaller ones on the side, as shown in (d). The current SFG is shown in (b) and the corresponding node split history is shown in (a). Starting from the ontology graph, we first split the P-node using the "type" attribute. Then, structure-based split is performed with all P-nodes selected. The L-node and R-node in the ontology graph are split based on their connections to the P-nodes, so that objects in each new L-node and R-node correspond to the same types of critical points.

We select all P-nodes and perform neighborhood inspection. The inspection result is shown without and with the egocentric layout in (b) and (c), respectively. Note that the same set of nodes (all P-nodes) is used for both structure-based split and neighborhood inspection. Therefore, all circular percentage bars are full, since if a node connects to a P-node, all its objects should connect to some object in the P-node. In addition, this indicates that connectors of P-nodes can be directly observed at the node level, and streamlines connecting to different sets of P-nodes can be distinguished by their colors in the volume view. The relationships between streamlines and critical points, and between spatial regions and critical points can be clearly observed in (b). 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 arrows in (b). We find that the P-nodes form two groups through the L-nodes. The largest P-node with ten attractive

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2773071, IEEE Transactions on Visualization and Computer Graphics



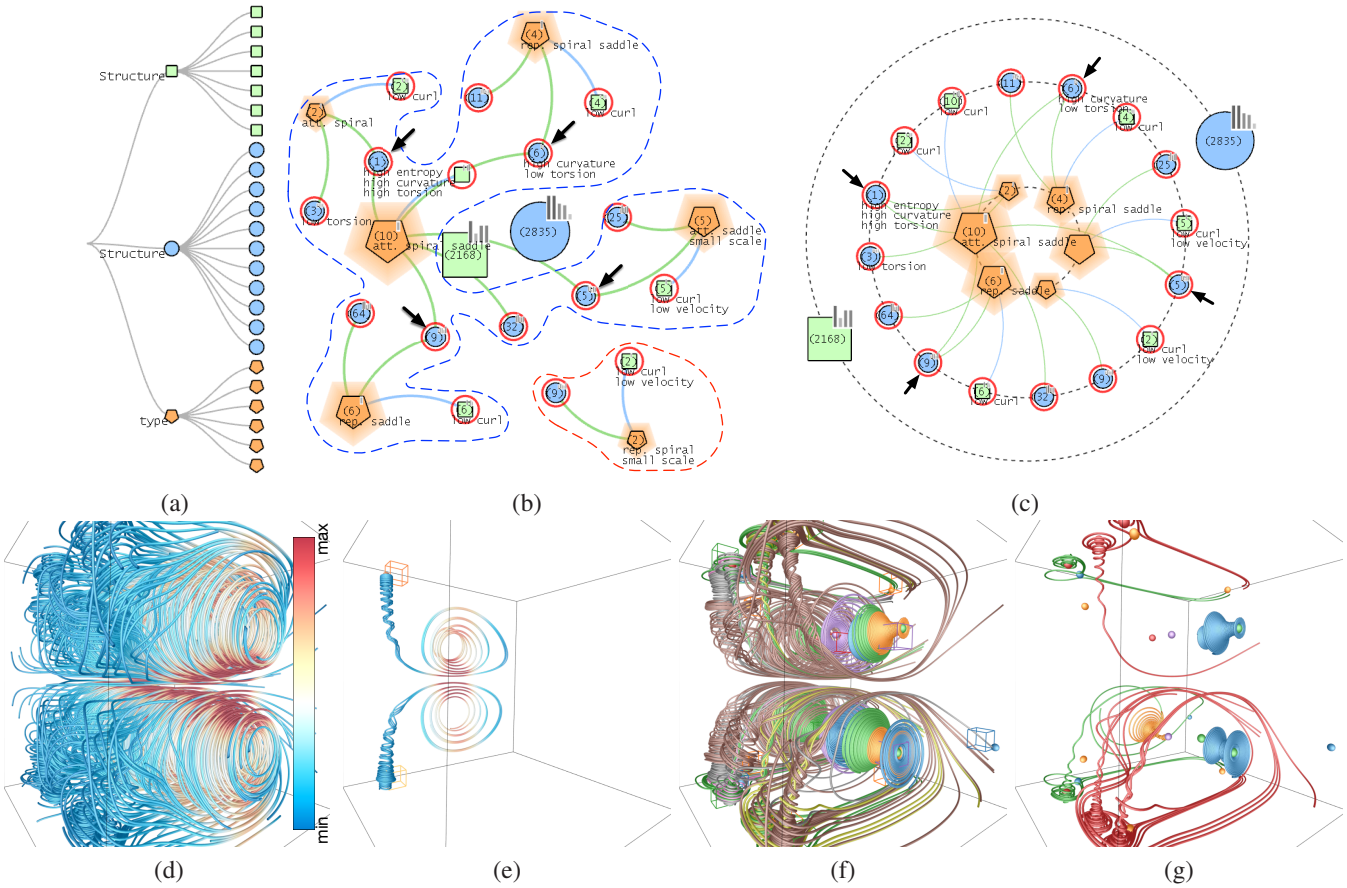



Fig. 5: Exploration of the two swirls data set. (a) The node split history. (b) Neighborhood inspection for all P-nodes without the egocentric layout. Each P-node represents a type of critical points. (c) The same inspection result of (b) with the egocentric layout. (d) 300 streamlines randomly selected from the entire pool of 3000 streamlines. The velocity magnitude is mapped to streamline colors. (e) and (f) The objects in the red and blue dashed boundaries of (b), respectively. (g) The connectors of the P-nodes.

spiral saddles connects to four other types of critical points (i.e., repelling saddles, repelling spiral saddles, attractive saddles, and attractive spirals) through L-nodes. These five P-nodes and their neighbors are marked by the blue dashed boundary in (b). The streamlines related to these four P-nodes are shown in (f). Since the L-nodes are generated by structure-based split, the color of a streamline actually indicates which types of critical points they are related to. The streamlines in the four L-node connectors are shown in (g). The two L-nodes connecting the attractive spirals, repelling spiral saddles, and attractive spiral saddles have the label "high curvature", since these three P-nodes exhibit spiral patterns leading to high-curvature streamlines. The other type of P-node (i.e., repelling spirals) is isolated, as no connector can be found between this P-node and other nodes. The streamlines related to this P-node are shown in (e). Compared to a randomly selected subset of 300 streamlines in (d), the streamlines in (e) and (f) reveal more inner structure of the flow field. We can only see the two larger swirling patterns and some smaller ones in (d), but their connections, and the small spirals inside the two large swirls can hardly be found.

In terms of observing the overall pattern of connections among nodes, the standard layout in (b) seems to be more effective than the egocentric layout in (c). Although the egocentric layout shows clearly the 1- and 2-hop neighborhoods, the two groups of critical points are still difficult to observe. However, finding the node pairs is relatively easy with the egocentric layout, as users can simply scan through the 1-hop neighborhood to see which nodes have two edges connecting to them.

**Case Study 3: Atmosphere.** The atmosphere data set contains 121 time steps. Each time step consists of one vector field indicating the wind directions and two associated scalar fields of PM10 and cloud fraction (CLDFRA). We first investigate a single time step (59) in the middle. Figure 6 shows the exploration of this time step. We split the R-node in the ontology graph using "PM10", since its distribution is of particular interest. We further split the R-node with "high PM10" and each of the eight disjoint regions becomes one R-node, as shown in (a) and (b). The resulting SFG shows a strong relationship between PM10 and cloud fraction, since every R-node labeled "high PM10" has "low CLDFRA". In addition, we find that the critical points are less relevant to "high PM10" regions, since only one of the eight "high PM10" R-nodes connects to the P-node. To investigate into the regions that may be affected by these "high PM10" regions, we perform neighborhood inspection on "high PM10" R-nodes. In (c), we can see that the streamlines passing these regions cover mostly the lower part of the volume which is close to the ground. This indicates that the PM10 pollution is more likely to propagate onto the ground by the wind. We further examine the connections among these "high PM10" regions by performing connector inspection on the corresponding R-nodes. In (d), the streamlines following a 'C'-shape at the lower-right part pass through several "high PM10" regions. This indicates that the other regions passed by these

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2773071, IEEE Transactions on Visualization and Computer Graphics



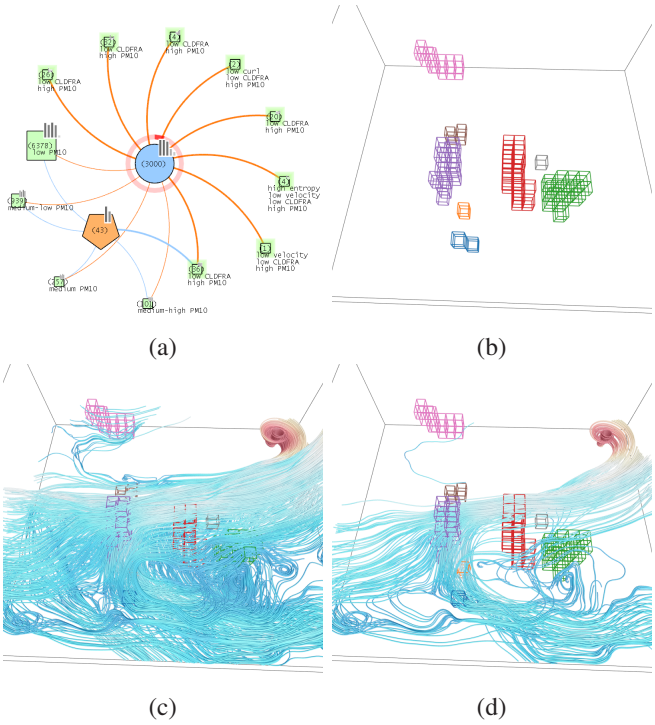



Fig. 6: Exploration of the atmosphere data set at time step 59. (a) Current SFG with connectors of high PM10 regions. (b) High PM10 regions in the volume view. (c) and (d) Resulting streamlines of neighborhood inspection and connector inspection on high PM10 regions.

streamlines may be under greater influence of the PM10 pollution, since they are affected by different "high PM10" regions.

**Case Study 4: Supercurrent.** Figure 7 shows the exploration of the supercurrent data set, which is from a time-dependent Ginzburg-Landau (TDGL) superconductor simulation. The simulation has two outputs: the vector-valued supercurrent field and complex-valued scalar order parameter field. The supercurrent field characterizes the electric current in the material; the order parameter field is used to derive magnetic flux vortices, or simply vortices, which are 3D curves and are the most important features in the data [5]. Because of the Lorentz force from the magnetic flux, supercurrents swirl over the vortices.

We create C-nodes, a new type of feature nodes for the vortex cores, as shown by the red pentagons in (a). The rationale is based on the swirling supercurrent over the vortices. The edges between C-nodes and other types of nodes are defined as follows: an L-C edge is formed between a streamline and a vortex core if the distance between the closest points on them is smaller than a threshold; a P-C edge is formed between a critical point and a vortex core if the distance between the critical point and the closest point on the vortex core is smaller than a threshold; and an R-C edge is formed between a spatial region and a vortex core if the vortex core passes the region.

In this data set, the vortex cores are mostly evenly spaced in the domain, as shown in (b). We apply structure-based split with the C-node in the ontology graph selected to distinguish the objects that are related to the vortex cores and those that are not. The interactions between the vortex cores and the related streamlines and critical points are particularly interesting. Therefore, we hide the other nodes, split the C-node into nodes containing individual vortex cores, and split the P-node related to the vortex cores according to their types. The resulting SFG is shown in (a).

We demonstrate the streamlines that are related to vortex cores (c), related to critical points and vortex cores (d), and connecting multiple vortex cores (e), respectively. In (c), using neighborhood inspection, the streamlines related to the vortex cores confirm that supercurrents swirl over the vortices. Most of the streamlines swirl over individual vortices, but some of them swirl over multiple vortices. Besides, vortices are mainly related to saddles, as revealed by the SFG in (a). Saddles are formed, because supercurrents move in opposite directions between neighboring vortices. The relationships between vortices and critical points can also be observed by inspecting the neighbors of the critical points, as shown in (d). In (d), streamlines that are not related to the critical points are removed, thus the figure provides a clear visualization of the relationships between critical points and vortices. In (e), we inspect the connectors between the C-nodes to identify the streamlines related to multiple vortex cores. None of the connector streamlines is related to the vortex cores at the centers. Instead, most of the streamlines connect the cores on the two sides. Our tool enables further discovery of the complex relationship between vortices and streamlines based on the interactive exploration.

**Case Study 5: Combustion.** The combustion data set has nine time steps and four associated scalar fields: namely, pressure (P), density (RHO), reaction progress (PROG), and temperature (Temp). The propagation of a premixed flame into an unburned fuel/air mixture in a homogeneous isotropic turbulent flow was simulated [7]. Each spatial region is now associated with a "time step" attribute. A region is connected to a pathline if the pathline passes that spatial region at the corresponding time step. To understand the temporal behaviors of this data set, we also include the finite-time Lyapunov exponents (FTLE) as an additional attribute of the spatiotemporal regions. The FTLE field at a specific time step is computed by tracing particles from that time step for a time span. Since this data set has only nine time steps, we trace the particles until the last time step. To investigate the evolution of critical points, we introduce the feature flow lines as a new type of nodes. The feature flow lines are traced in the feature flow fields [27] to indicate the movement of critical points over time.

We first explore the first time step. We split the P-node and R-node using "time step" and hide the nodes at the other time steps. Since the critical points are related to "low PROG" regions, we split the R-node at the first time step using "PROG" and perform neighborhood inspection with the P-node selected. The inspection result is shown in Figure 8 (a). Note that the feature flow lines are hidden as well for clearer observation. We can see that most of the critical points reside in the blue regions ("low PROG"). In the combustion data set, "PROG" is the normalized temperature and the "low PROG" regions in Figure 8 (a) correspond to an unburned fuel/air mixture with low temperature. The critical points in this data set are associated with complex turbulence structures that are injected at a boundary. Such turbulence structures disappear as they pass through the flames with high temperature ("high PROG"). The result shown in Figure 8 (a) is consistent with such a phenomenon.

To understand the temporal development of flows, we merge the R-nodes with different "PROG" and split the resulting R-node using "FTLE". In Figure 8 (c) and (d), we perform neighborhood inspection with "high FTLE" and "low FTLE" R-nodes selected, respectively. We find that most of the "high FTLE" regions locate at the boundaries, and their related pathlines are usually more

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2773071, IEEE Transactions on Visualization and Computer Graphics



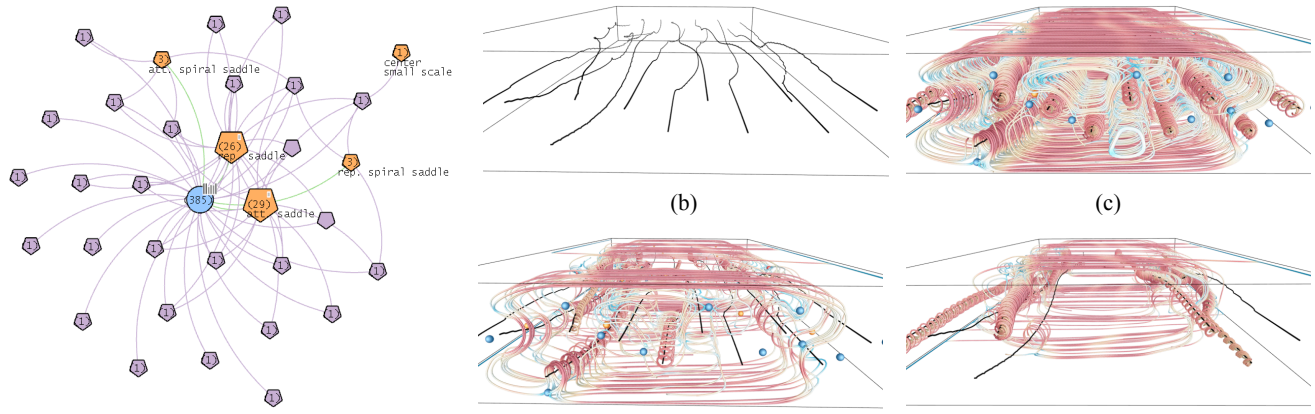



Fig. 7: Exploration of the supercurrent data set. (a) Current SFG to inspect the relationships between the vortex cores, critical points, and streamlines. (b) The vortex cores in the volume view. (c) The vortex cores and the related streamlines and critical points. (d) The critical points and the related streamlines and vortex cores. (e) The vortex cores and their connectors.

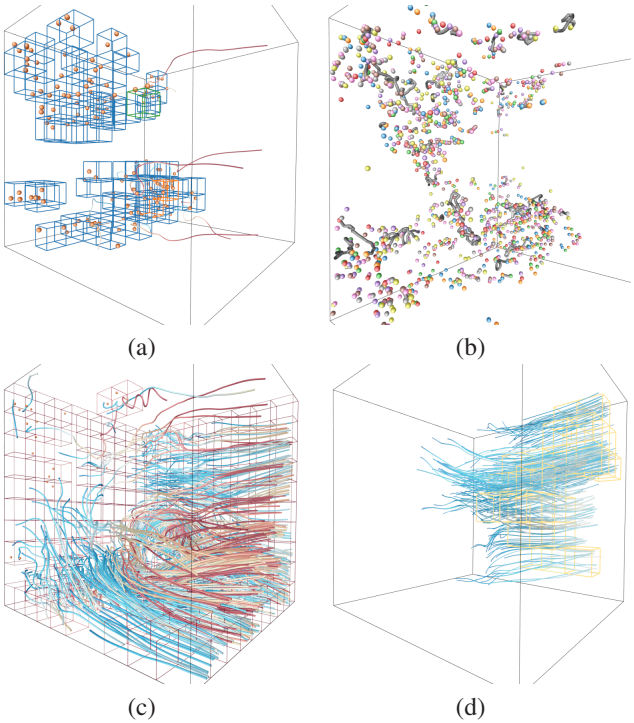



Fig. 8: Exploration of all the time steps of the combustion data set. (a) The critical points at the first time steps and their neighboring objects. (b) All critical points and the feature flow lines connecting them. (c) and (d) High and low FTLE regions and their respective neighboring objects.

diverse at the beginning stage. In contrast, the pathlines related to the "low FTLE" regions are more similar. We then investigate the evolution of critical points. We show the P-nodes at other time steps again, hide the L-nodes and R-nodes, and perform connector inspection. The feature flow line segments that connect critical points at different time steps are shown in Figure 8 (b). There are 1778 critical points in total and each time step contains 148 to 227 critical points. However, only 187 feature flow lines are detected as connectors and most of them are too short to be noticed in the volume view. This indicates the short existence of critical points in the combustion data set, which is caused by a stochastic nature of the turbulent flow field. Most of the critical points at a time step are born as new ones instead of corresponding to critical points at previous time steps.

### 5.2 Comparison to Previous Approaches

We compare our SFG framework with previous approaches based on their ability to handle the following tasks:

- **T1. Discovery of flow patterns**: identifying the common flow patterns or flows with certain characteristics, and locating patterns of interest.
- **T2. Region analysis**: understanding how the spatial regions are connected by flows.
- **T3. Exploration of flow features**: specifying flow features for display and revealing their interactions.
- **T4. Analysis of associated scalar fields**: revealing the relationships between the flow field and associated scalar fields.
- **T5. Support for unsteady flow fields**: revealing the evolution of flows and features in unsteady flow fields.

We compare our approach against four previous approaches: namely, FlowString [25], [26] (FS), flow web [35] (FW), FlowGraph [11], [12] (FG), and saddle connectors [28] (SC). Table 2 shows a summary of this comparison. For the discovery of flow patterns (**T1**), FS fully supports this task by extracting and visualizing the basic patterns for user query. Both FW and FG preserve the flow patterns to some degree in the graph layout, but they may require trial-and-error efforts to identify the exact patterns. SC depicts the flow patterns related to critical points, but other patterns may be missing. Our SFG partially supports this task. Although it allows users to identify flow patterns indirectly through other elements (e.g., critical points) or field line characteristics (e.g., entropy), no visual hint of the patterns is provided by the graph itself. For the region analysis (**T2**), FS and SC do not explicitly encode the regions to support this task. Among these five methods, FW and FG are the best in terms of describing the connections among regions with respect to flows. But these two approaches organize regions in a fixed hierarchy. In contrast, our SFG supports dynamic grouping of regions according to various criteria, but their spatial relationships are less perceivable in the layout. Therefore, we consider this task to be partially supported by FW, FG, and our

SFG. For the exploration of flow features (**T3**), FW and FG do not support this task since the features are not explicitly captured. FS can capture the features that are related to the shape of individual streamlines (e.g., swirls), but it is not capable of capturing other features. SC provides precise descriptions to the critical points and their relationships, but it does not support other types of features or allow users to specify critical points of interest for observation. Our SFG fully supports this task. It provides great extensibility to support the exploration of various types of features by creating multiple types of nodes. The relationships among the features and other objects can be discovered using the operations provided by SFG. For the analysis of associated scalar fields (**T4**) and support for unsteady flow fields (**T5**), our SFG is the only one that supports both tasks. Similar to our approach, FG handles unsteady flow fields by partitioning the entire spatiotemporal domain into 4D blocks. But SFG can be flexibly extended to incorporate established tools (e.g., FTLE and feature flow fields) to study the temporal development of unsteady flow fields, which is not supported by FG.

Overall, we find that the major merit of SFG is its flexibility, thanks to the semantic nature of the graph. By assigning additional attributes and creating extra types of nodes, SFG can be extended to incorporate new features or analysis tools. Meanwhile, the operations allow the connections among these fundamental objects to be discovered, which is critical to support various kinds of tasks.

## 6 Expert Review

To evaluate the effectiveness of our approach, we collaborated with Dr. Seung Hyun Kim and conducted an expert evaluation. Dr. Kim is an expert in computational fluid dynamics with more than 20 years of experience. His research interests include turbulent combustion modeling and simulation. Several tasks were designed to guide him through the exploration procedure, including identifying streamlines or regions with certain properties, discovering relationships between the identified streamlines and regions, locating certain types of critical points, and distinguishing streamlines connecting different types of critical points, etc. Dr. Kim was informed that the comments behind the rating were more important than the accuracy of performing the tasks.

The evaluation was performed in three stages, each of which took around two hours. At the first stage, Dr. Kim was introduced to the tool, including the meaning of the semantic graph and the interactions to perform the tasks. He interacted with the tool and asked questions to ensure that his understanding was correct. The second stage was performed five days later, which gave him enough time to digest the content. At the second stage, he was asked to follow our task list using the five critical points data set, and to freely explore the two swirls and atmosphere data sets. At the third stage, he was introduced the exploration of unsteady flow fields, and freely explore the combustion data set. Instead of measuring the timing and accuracy of performing these tasks, we designed a set of open questions that ask him to comment on different aspects of our approach, which serves as a guideline for him to organize his comments. After performing these tasks, Dr. Kim provided the following comments in written format.

"In terms of SFG operations, attribute-based split is intuitive and easy to use. It facilitates the exploration by categorizing different regions and flow structures. In addition to the attributes currently available in the program, other quantities such as strain rates and dissipation rates could be added. It is also helpful if a user can define an attribute based on primitive variables in the flow field data. Structure-based split, on the other hand, is somewhat complex to understand and use. It would be more logical to select two categories of nodes separately, i.e., a set of nodes to be split and the other set of nodes for which structural equivalence is inspected. The expert also felt that neighborhood inspection and connector inspection are complementary to each other. Together they offer an effective way of investigating the relationship among different flow structures and regions. For instance, neighborhood inspection and connector inspection are effective in identifying flow structures around regions with low or high values of scalars or particular flow characteristics, e.g., high entropy. The use of three views, namely, graph, history, and volume views, is effective in navigating the data and managing the operations. The learning curve seems a little steep especially for those who are only familiar with traditional visualization techniques, but this approach can be beneficial once its functions are fully appreciated."

| | T1 | T2 | T3 | T4 | T5 |
|---|---|---|---|---|---|
| FlowString [26] | ++ | - | + | - | - |
| flow web [35] | + | + | - | - | - |
| FlowGraph [11], [12] | + | + | - | - | + |
| saddle connectors [28] | + | - | + | - | - |
| SFG | + | + | ++ | ++ | ++ |

TABLE 2: Tasks supported by previous approaches and our SFG. "++", "+", and "-" indicate that a task is fully supported, partially supported, and not supported by a method, respectively.

"SFG appears to be a particularly attractive approach for investigating the relationship between scalar transport and fluid flow. For instance, when studying pollutant dispersion, it would be interesting to find vulnerable regions with high levels of pollutant concentration and identify ways to reduce pollutant levels in those regions. The pollutant concentration field is determined by the flow field, and linking the high pollutant concentration regions with particular flow structures will be of primary importance to devise the mitigation strategy. In drug delivery, the concentration of drug should be high at targeted locations. In some drug delivery applications, SFG can be used to study the relationship of scalar transport with flow in complex structures in a human body. SFG is effective in understanding the evolution of features as well. In the study of turbulent mixing and combustion, often useful is a tool that tracks a certain feature, e.g., high scalar dissipation layers. During the evaluation using the unsteady combustion data set, the feature flow lines for the critical points were explored, which revealed that the critical points are short-lived. An extension of this feature to investigating scalar structures and their relationship with flow fields will be of interest."

"An advantage of SFG appears to be its flexibility in extending functionality. While three kinds of objects are currently used, it appears that the addition of new kinds of objects is straightforward and helpful in some applications. Similar to critical points, other pointwise features based on scalar quantities, for instance, local extrema, can be useful. In addition, it will be helpful for a user to select values used for attribute-based split. While automatic selection of values is convenient and should be kept, the flexibility in setting values for attributes may improve the clarity of the graph. For instance, the log-scale is more appropriate than the linear scale for some quantities, especially when derivatives of flow or scalar variables are involved in the evaluation. Similarly, it will also be beneficial to allow finer resolution of regions. Currently, the regions are generated by uniform division of the computational domain for the volume data. Once a user has

identified a particular region of interest, it may be necessary to refine such a region for more thorough investigation or enhanced clarity in the graph representation."

In addition, Dr. Kim gave the following comments to the user interface of SFG. "Structure-based split often generates complicated graphs especially when the current graph configuration already contains a number of nodes. It may be helpful to provide an option so that users can select the objects (nodes) that will be split, as discussed above. It may also be helpful to add the one-click "undo" function to easily recover a previous configuration. For colors used in the SFG visualization, with three kinds of objects, the current practice looks appropriate. As an alternative, it may be good to use the mixed colors of the nodes (objects), gradually changing from one color to the other along an edge. This will be particularly useful when more kinds of objects are used. Finally, the speed of animated transition in the graph view seems appropriate for keeping the mental map of the graphs."

## 7 Conclusions and Future Work

We have presented a novel approach that applies the concept of semantic graph to investigate the relationships among different types of objects in a flow field. We introduce semantic flow graph (SFG) that captures the connections among field lines, features, and spatiotemporal regions as a heterogeneous graph, and leverage semantic abstraction to simplify and explore this graph. We develop a suite of operations (attribute-based split, structure-based split, neighborhood inspection, and connector inspection) to explore the graph. In addition, a history view is designed to inform users how the current SFG is generated starting from the ontology graph, and to enable users to make convenient and swift transitions between any two configurations of SFG. Through brushing and linking, the volume view displays the graph exploration results in the original 3D space for making connections and observations.

In the future, we would like to investigate the following. First, we will extend our current split guidance to visual recommendation with a complete graph exploration. Split guidance provides navigation cues on splitting a node based on a certain attribute. However, this solution only "looks" one step ahead, and may not lead to a globally optimal solution. An ideal recommendation for exploration should balance between added information and visual complexity, and generate the most informative graph with multiple steps of exploration. This may potentially lead to an automatic solution for SFG generation by simply applying the recommended operations. Second, we will study automatic comparison of two SFG configurations. For example, users may split the L-node in the ontology graph using different attributes to generate different SFGs. However, they have to manually compare the two graphs or the corresponding streamline visualizations to examine the differences. Automatic comparison of two SFGs will be beneficial for users to discover the differences between streamline clusters and to figure out how their relationships to the regions and critical points differ from each other. Third, attribute-based split uses a set of predefined attributes and evenly partitions their ranges to group the nodes. As pointed out by the domain expert, allowing users to define their own attributes and pick the values to partition the ranges will be beneficial. These can be implemented by introducing arithmetic and derivative operators that generate new attributes from flow fields and providing an interface that displays the distribution of an attribute for users to manually divide its range. It will also be helpful to design automatic scheme for partitioning attribute ranges. In addition, when an R-node is split based on a scalar attribute, we may partition its corresponding region based on the isosurfaces instead of regular blocks. This will better preserve small or thin structures in the scalar field. Fourth, our current implementation only consider low-order critical points. To explore the relationship among more complex flow features, we will seek more advanced solutions for high-order critical point detection or more general definition of flow features [31].

## Acknowledgements

This research was supported in part by the U.S. National Science Foundation through grants IIS-1456763 and IIS-1455886, the Notre Dame Global Collaboration Initiative Program, China National 973 project 2014CB340301, and the National Natural Science Foundation of China through grants No. 61379088 and No. 61772504. We would like to thank Dr. Hanqi Guo for providing the supercurrent data set and giving comments on the results, Dr. Reid Johnson and Martin Imre for his narration of the accompanying video, and Yunde Su for preparing the combustion data set.

## References

[1] R. Bujack, J. Kasten, I. Hotz, G. Scheuermann, and E. Hitzer. Moment invariants for 3D flow fields via normalization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 9–16, 2015.

[2] C.-M. Chen, L. Xu, T.-Y. Lee, and H.-W. Shen. A flow-guided file layout for out-of-core streamline computation. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 145–152, 2012.

[3] Y. Frishman and A. Tal. Dynamic drawing of clustered graphs. In *Proceedings of IEEE Symposium on Information Visualization*, pages 191–198, 2004.

[4] 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.

[5] H. Guo, C. Phillips, T. Peterka, D. Karpeyev, and A. Glatz. Extracting, tracking, and visualizing magnetic flux vortices in 3D complex-valued superconductor simulation data. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):827–836, 2016.

[6] E. Heiberg, T. Ebbers, L. Wigström, and M. Karlsson. Three-dimensional flow characterization using vector pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 9(3):313–319, 2003.

[7] S. H. Kim. Leading points and heat release effects in turbulent premixed flames. *Proceedings of the Combustion Institute*, 36(2):2017–2024, 2017.

[8] T. Klein and T. Ertl. Scale-space tracking of critical points in 3D vector fields. In H. Hauser, H. Hagen, and H. Theisel, editors, *Topology-Based Methods in Visualization*, pages 35–49. Springer-Verlag Berlin Heidelberg, 2007.

[9] H. Löffelmann, H. Doleisch, and M. E. Gröller. Visualizing dynamical systems near critical points. In *Proceedings of Spring Conference on Computer Graphics*, pages 175–184, 1998.

[10] F. Lorrain and H. C. White. Structural equivalence of individuals in social networks. *Journal of Mathematical Sociology*, 1(1):49–80, 1971.

[11] 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.

[12] 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.

[13] S. Mann and A. Rockwood. Computing singularities of 3D vector fields with geometric algebra. In *Proceedings of IEEE Visualization Conference*, pages 283–290, 2002.

[14] S. C. North. Incremental layout in DynaDAG. In *Proceedings of International Symposium on Graph Drawing*, pages 409–418, 1996.

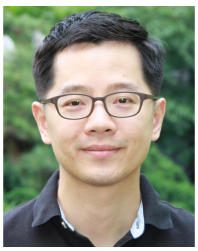
[15] C. Rössl and H. Theisel. Streamline embedding for 3D vector field exploration. *IEEE Transactions on Visualization and Computer Graphics*, 18(3):407–420, 2012.

[16] T. Salzbrunn and G. Scheuermann. Streamline predicates. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1601–1612, 2006.

[17] M. Sarkar and M. H. Brown. Graphical fisheye views of graphs. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 83–91, 1992.

[18] G. Scheuermann, H. Hagen, H. Krüger, M. Menzel, and A. Rockwood. Visualization of higher order singularities in vector fields. In *Proceedings of IEEE Visualization Conference*, pages 67–74, 1997.
[19] M. Schlemmer, M. Heringer, F. Morr, I. Hotz, M.-H. Bertram, C. Garth, W. Kollmann, B. Hamann, and H. Hagen. Moment invariants for the analysis of 2D flow fields. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1743–1750, 2007.
[20] D. Schroeder, D. Coffey, and D. F. Keefe. Drawing with the flow: A sketch-based interface for illustrative visualization of 2D vector fields. In *Proceedings of ACM SIGGRAPH/Eurographics Sketch-Based Interfaces and Modeling*, pages 49–56, 2010.
[21] Z. Shen, K.-L. Ma, and T. Eliassi-Rad. Visual analysis of large heterogeneous social networks by semantic and structural abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1427–1439, 2006.
[22] L. Shi, Q. Liao, H. Tong, Y. Hu, Y. Zhao, and C. Lin. Hierarchical focus+context heterogeneous network visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 89–96, 2014.
[23] B. Shneiderman and A. Aris. Network visualization by semantic substrates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):733–740, 2006.
[24] 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.
[25] J. Tao, C. Wang, and C.-K. Shene. FlowString: Partial streamline matching using shape invariant similarity measure for exploratory flow visualization. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 9–16, 2014.
[26] J. Tao, C. Wang, C.-K. Shene, and R. A. Shaw. A vocabulary approach to partial streamline matching and exploratory flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(5):1503–1516, 2016.
[27] H. Theisel and H.-P. Seidel. Feature flow fields. In *Proceedings of Eurographics/IEEE TCVG Symposium on Visualization*, pages 141–148, 2003.
[28] H. Theisel, T. Weinkauf, H.-C. Hege, and H.-P. Seidel. Saddle connectors - an approach to visualizing the topological skeleton of complex 3D vector fields. In *Proceedings of IEEE Visualization Conference*, pages 225–232, 2003.
[29] C. Wang. Graph-based techniques for visual analytics of scientific data sets. *IEEE Computing in Science & Engineering*, 21(1), 2018. In Press.
[30] C. Wang and J. Tao. Graphs in scientific visualization: A survey. *Computer Graphics Forum*, 36(1):263–287, 2017.
[31] Z. Wang, H.-P. Seidel, and T. Weinkauf. Multi-field pattern matching based on sparse feature sampling. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):807–816, 2016.
[32] M. Wattenberg. Visual exploration of multivariate graphs. In *Proceedings of ACM Conference on Human Factors in Computing Systems*, pages 811–819, 2006.
[33] J. Wei, C. Wang, H. Yu, and K.-L. Ma. A sketch-based interface for classifying and visualizing vector fields. In *Proceedings of IEEE Pacific Visualization Symposium*, pages 129–136, 2010.
[34] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007.
[35] L. Xu and H.-W. Shen. Flow web: A graph based user interface for 3D flow field exploration. In *Proceedings of IS&T SPIE Conference on Visualization and Data Analysis*, 2010.
[36] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3D streamlines. In *Proceedings of IEEE Visualization Conference*, pages 471–478, 2005.

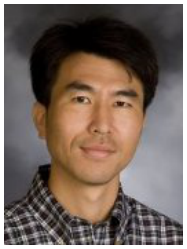
**Jun Tao** is currently a postdoctoral researcher at University of Notre Dame. He received a Ph.D. degree in computer science from Michigan Technological University in 2015. Dr. Tao's major research interest is scientific visualization, especially on applying information theory, optimization techniques, and topological analysis to flow visualization and multivariate data exploration. He is also interested in graph-based visualization, image collection visualization, and software visualization. He received the Dean's Award for Outstanding Scholarship (2015) and the Finishing Fellowship (2015) from Michigan Technological University, and a best paper award at IS&T/SPIE VDA (2013).



**Chaoli Wang** is an associate professor of computer science and engineering at University of Notre Dame. He received a Ph.D. degree in computer and information science from The Ohio State University in 2006. Prior to joining Notre Dame, he was a postdoctoral researcher at University of California, Davis and an assistant professor of computer science at Michigan Technological University. Dr. Wang's main research interest is scientific visualization, in particular on the topics of time-varying multivariate data visualization, flow visualization, and information-theoretic algorithms and graph-based techniques for big data analytics. He is a recipient of the NSF CAREER Award (2014), best paper awards at IS&T/SPIE VDA (2013, 2015), and an honorable mention at IEEE PacificVis (2013).

**Nitesh V. Chawla** is the Frank M. Freimann Professor of computer science and engineering at University of Notre Dame. He received a PhD degree in computer science and engineering from University of South Florida in 2002. At Notre Dame, Dr. Chawla directs the Interdisciplinary Center for Network Science and Applications (iCeNSA), an institute focused on network and data science. His research work has led to many interdisciplinary contributions in social networks, healthcare analytics, environmental sciences, learning analytics, and media. He received prestigious recognitions such as IBM Big Data Award, IBM Watson Faculty Award, IEEE CIS Outstanding Early Career Award, National Academy of Engineers New Faculty Fellowship, and Outstanding Teacher Awards.

**Lei Shi** is an associate research professor at Chinese Academy of Sciences. Before that, he was a research staff member and research manager at IBM Research - China. He received B.S. (2003), M.S. (2006), and Ph.D. (2008) degrees in computer science and technology from Tsinghua University. His current research interests are visual analytics and data mining, with more than 60 papers published in top-tier venues, such as IEEE TVCG, TKDE, TC, VIS, ICDE, ICDM, INFOCOM, ACM SIGCOMM and CSCW. He is the recipient of IBM Research Division Award on "Visual Analytics" and IEEE VAST Challenge Awards (2010, 2012).



**Seung Hyun Kim** is an assistant professor of mechanical and aerospace engineering at The Ohio State University. His research focuses on the modeling of multiscale and multiphysics problems in relation to energy science and technology. He received B.S. (1996), M.S. (1998), and Ph.D. (2003) degrees in mechanical engineering from Pohang University of Science and Technology. He was a postdoctoral researcher and a research associate at Stanford University, and an assistant professor of mechanical engineering at Michigan Technological University.