Interactive Visual Exploration of Neighbor-Based Patterns in Data Streams *

Di Yang, Zhenyu Guo, Zaixian Xie, Elke Rundensteiner, Matthew Ward
Worcester Polytechnic Institute
diyang|zyguo|xiezx|rundenst|matt@cs.wpi.edu

ABSTRACT

We will demonstrate our system, called $ViStream$, supporting interactive visual exploration of neighbor-based patterns [7] in data streams. $Vistream$ does not only apply innovative multi-query strategies to compute a broad range of popular patterns, such as clusters and outliers, in a highly efficient manner, but it also provides a rich set of visual interfaces and interactions to enable real-time pattern exploration. In our demonstration, we will illustrate that with ViStream, analysts can easily interact with the pattern mining processes by navigating along the time horizons, abstraction levels and parameter spaces, and thus better understand the phenomena of interest.

Categories and Subject Descriptors

H.0 [**Information interfaces and presentation**]: General

General Terms

Algorithm, Management, Human Factor.

Keywords

Streaming Data, Pattern mining, Visual Interaction.

1. INTRODUCTION

The discovery of complex patterns such as clusters, outliers, and associations from huge volumes of streaming data has been recognized as critical for domains from moving object monitoring to stock transaction analysis. For example, huge numbers of pattern mining requests, such as the queries asking for intensive-transaction areas (clusters), are submitted against the transaction stream from NYSE by a large population of financial analysts every day.

*This work is supported under NSF grant CCF-0811510, IIS-0812027, IIS-0119276 and IIS-00414380. We thank our collaborators at MITRE Corporation, J.Casper and P. Leveille, for providing us the GMTI data stream generator.

SIGMOD'10, June 6–11, 2010, Indianapolis, Indiana, USA.

In order to help analysts in finding and interpreting the patterns hidden in data streams, two major challenges have to be conquered. First, advanced computational methods have to be designed to efficiently extract patterns from data streams in real time. While previous work has studied the problem of stream pattern mining for single queries [7, 1, 2], we have developed the first shared execution strategy for handling multiple neighbor-based pattern mining requests in streaming environments [8]. Second, a visual interactive platform is essential to enable analysts to directly interact with the pattern mining processes. Reporting large numbers of detected patterns using a simple console or a file output make it hard for analysts to understand the patterns and even impossible to make real time decisions.

Second, since the pattern mining process itself is driven by expert knowledge, highly dynamic streaming environments make real-time feedback from analysts and quick adjustment to the computation a necessity. When the characteristics of the input stream significantly change, analysts need interactive methods to exploratorily adjust the parameters settings to keep the pattern extraction process optimally tuned.

Unfortunately, state-of-art pattern mining systems [1, 2], although well equipped with computational components, have paid little attention to applying visualization or interaction techniques on detected patterns. On the other hand, traditional visualization systems [4, 5], which usually have sophisticated visualization support yet limited computational capabilities, do not typically support interactive pattern mining in high speed data streams.

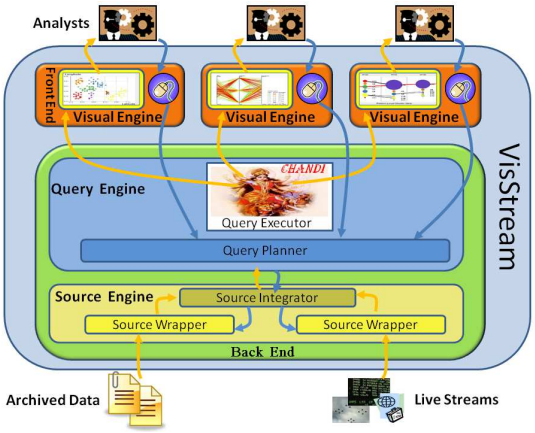


Figure 1: System Architecture of $ViStream$

Over the past 13 years the XMDV team at WPI, composed of visualization, HCI and database experts, supported by a series of five NSF grants, has developed a freeware visual tool suite XmdvTool ($http://davis.wpi.edu/xmdv/$) to facilitate interactive data exploration. In our current project effort, we now focus on extending this tool by supporting interative neighbor-based pattern exploration in streaming data. In particular, we make the following contributions: Compute neighbor-based patterns from data streams for multiple queries using highly efficient shared execution strategies. Model and extract pattern evolution over time. Organize and display detected patterns as well as their evolution over time for multiple analysts. Allow analysts to visually explore the detected patterns by navigating along time axis, abstraction levels and different queries. Facilitate multiple analysts to exploratorily adjust parameter settings of queries at run time.

2. ARCHITECTURE OF VISTREAM

Vistream is composed of three integrated components: source engine, query engine and visual engine. At the computational back end, the source engine wraps and integrates data sources either from live streams or from archived data. The query engine then efficiently compute the patterns for multiple queries. The query engine has two major subcomponents: 1) A query planner which generates optimized query plans by grouping similar queries and constructing an integrated plan for each group. 2) A query executor which executes the query plans, by using the shared query execution strategy we proposed in [1], called *Chandi* [1] . At the visual front end, our visual engine supports visual stream analytics. It visualizes the detected patterns using the visualization pipelines proposed in the steaming version of XmdvTool [6], and also provides various visual interactions for analysts to explore the detected patterns.

3. THE BACK END: EFFICENT PATTERN AND EVOLUTION EXTRACTION

Vistream employs the shared pattern extraction strategy, **Chandi**[8], as computational method of pattern extraction for multiple queries with different parameter settings. The neighbor-based pattern mining queries over streaming data, such as density-based clusters [3] or distance-based outliers [7] detection, have parameter settings for specifying pattern definitions, such as minimum requirement for density, and those for specifying query window characteristics, such as window size *win* and slide size *slide*.

To share computation among queries with arbitrary parameter settings, We first characterize the conditions under which the patterns identified by different queries can be incrementally stored in a single compact structure. For example, the density-based clustering queries [3] specified on a dataset are defined by two parameter settings, namely a range threshold θ^{range} and a count threshold θ^{cnt}. The range threshold θ^{range} defines the concept of "neighborship" between tuples in the dataset, indicating that any pair of tuples which has a distance smaller or equal to this threshold between them are considered as "neighbors" of each other. On the other hand, the count threshold θ^{cnt} defines the minimum number of neighbors that the tuples are required to have to qualify as "core objects", which compose the "skeleton" of each cluster. By analyzing the characteristics of the cluster sets identified by density-based clustering queries with different parameter settings, we made the following observation. Given two queries Q_i and Q_j, if $Q_i.\theta^{range} \geq Q_j.\theta^{range}$ and $Q_i.\theta^{cnt} \leq Q_j.\theta^{cnt}$, then Q_i is more "relaxed" than Q_j. This indicates that the clusters identified by Q_i will be the "expansion" or "merge" of the clusters identified by Q_j or completely the "new clusters" (not identified by Q_j before) [8].

Generally, if any two data points are identified to be in the same cluster by a query Q_j, they are guaranteed to be identified to be in the same cluster by a more "relaxed" query Q_i. Given this "growth property", *Chandi* organizes the cluster structures identified by a group of queries with arbitrary pattern parameter settings into a single tree-based structure. It thus realizes incremental cluster structure storage and maintenance for all queries. This is a significant improvement to the independent cluster maintenance strategy for multiple queries, because both the CPU and memory utilization of such strategy increase dramatically as the number of queries increases, and thus it does not scale for large number of queries.

Second, *Chandi* proposes a meta-query strategy which utilizes a single query to answer multiple queries with different window-specific parameters. In particular, by leveraging the potential overlaps among sliding windows, *Chandi* composes a single meta-query whose meta-information (progressive clusters) maintained is sufficient to answer a group of queries with arbitrary window-specific parameter settings, namely the arbitrary window size *win* and slide size *slide*.

By elegantly combining these two techniques above, *Chandi* achieves significant savings of computational and memory resources due to shared execution among large numbers of queries (in the order of hundreds or even thousands).

Beyond extraction of patterns, our query engine also extracts the evolution of patterns over time using our evolution extraction technique proposed in [9]. In particular, it incrementally tracks the changes of the detected patterns across windows. By doing so, it builds a temporal context for detected patterns and thus help analysts to understand the interrelationship between patterns detected in different time periods. This evolution tracking process is complimentary with our pattern mining process, indicating that the determination of these changes described by our proposed evolution semantics requires very modest additional computational costs.

4. THE FRONT END: VISUAL DISPLAY AND INTERACTION

At the front end, *ViStream* provides one visual engine for each analyst to interact with the detected patterns as well as the pattern mining process.

Visualization of and Interaction in Pattern Space. For a single pattern mining query, *ViStream* organizes the detected patterns and their evolution information into a multi-dimensional pattern space. In particular, our current system supports a two dimensional pattern space, with the dimensions representing the changes over time and across abstraction levels respectively.

Along the dimension of abstraction levels, we support views of detected patterns at the tuple and the pattern level.

[1]Name of a god with multiple hands in hindu theology.

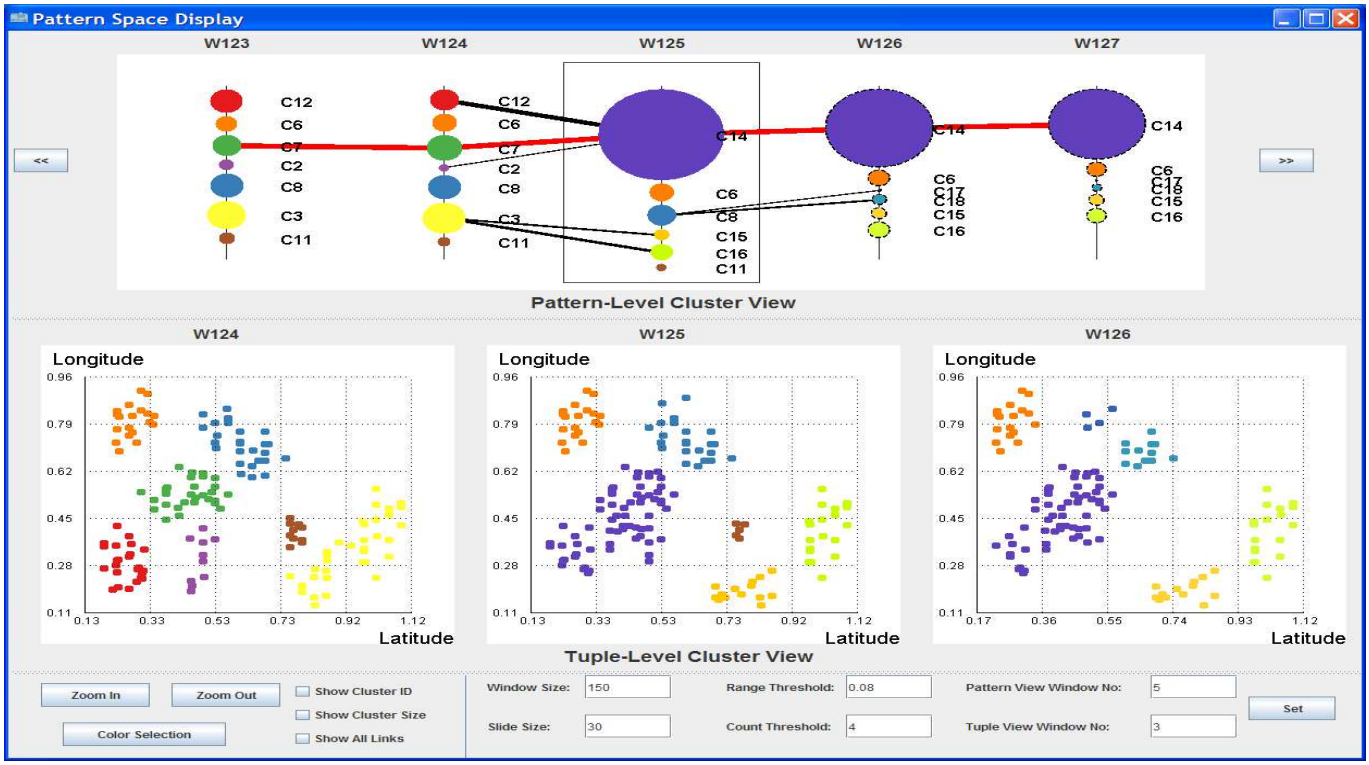


Figure 2: A screen shot of our visualized pattern space.

At the pattern level, each pattern in a window is abstracted as a single object. Taking density-based clusters as example (Figure 2), each cluster is depicted using a colored circle. Two important characteristics of a cluster, namely its size (population) and its position, are conveyed by its radius and relative position in the view respectively. In the tuple level view, specific information about pattern members (tuples) is displayed.

Along the time dimension, our pattern space provides a sequence of views representing the clusters identified in different portions of the data stream. More specifically, each view in the pattern space is a snapshot of the patterns identified in a single window. Those views are organized in the order of their recentness, and more importantly, "connected" based on the evolution relationships among each other [9]. First, we correlate the same patterns in different windows by mapping them to the same color. Second, we propose to customize the "river metaphor" technique initially designed to visualize frequency changes to express the evolution of patterns. In particular, the derivation of each cluster from the one window to the next is presented by "rivers" (links) between them. More specifically, for a given pattern P_i in window W_{n+1}, if it is considered to be (partially) derived from a pattern P_j in the last window W_n, our system draws a "river" between P_j and P_i (as shown in Figure 2). In addition, the volume of each (width of each link) represents the number of pattern members that the pattern in the later window inherits from the previous window. Such "river metaphor" lineages visualization [9] helps analysts to easily understand the evolution of patterns over time.

ViStream also provides a rich set of interactions for monitoring patterns and their evolution. First, analysts can navigate along both dimensions of the pattern space to observe the patterns and evolution in any past, present or near future window [1]. Second, analysts can choose to focus on any pattern of interest by zooming into, and then retrieving information about individual pattern members. Third, analysts can visually adjust the parameter settings of any query.

Multiple Query Visual Exploration. *ViStream* provides two visualization mechanisms for applications in which an analyst needs to monitor the pattern detection process of multiple queries simultaneously through a single interface, namely "juxtaposed" and "integrated visualization" [6]. We aim to empower the analysts to compare and contrast patterns detected by queries with different parameter settings.

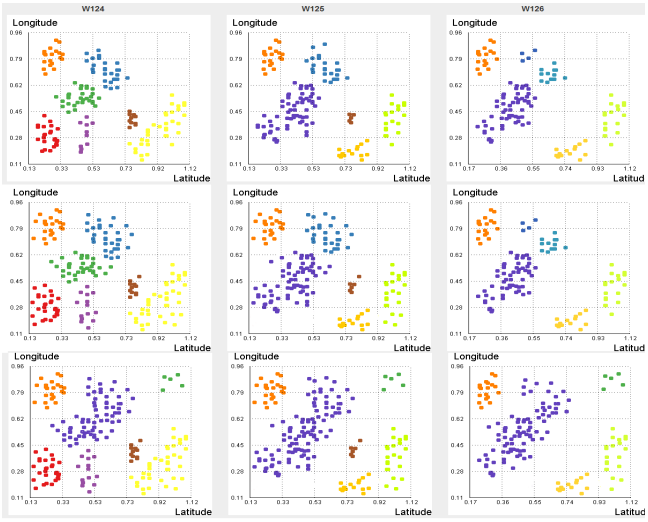
In the *juxtaposed* visualization, the patterns detected by different queries are visualized separately as shown in Figure **3.a** for three density-based clustering queries. The key advantage of this technique is that it minimizes the interference between the visualization of different queries.

The *integrated* visualization visualizes the patterns detected by similar queries in an integrated fashion, and thus more explicitly shows their commonalities and differences in a single display. In particular, taking density-based clusters as example, *ViStream* exploits the hierarchical relationships among cluster structures identified by similar queries, called "growth property" [8], to incrementally visualize them in a single display. For a group of similar yet differently parameterized queries QG, *ViStream* adopts a hierarchical color coding to incrementally display the cluster members iden-

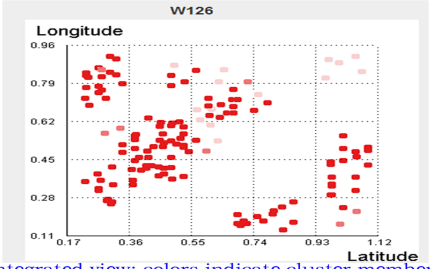
[1]Techniques regarding pattern prediction in near future windows can be found in [9].

tified by different subsets of QG, as shown in Figure **3.b**. In this example, tuples in the lightest color are identified as cluster member by only a single query $Q1$, while darker ones are those identified by more and more queries, such as both $Q1$ and $Q2$, etc. This is because other queries, such as $Q2$, is more "relaxed" than $Q1$ and will thus identify a superset of tuples as cluster members in the same dataset. This integrated visualization highlights how cluster structures change ("grow") as the parameter settings change (get more "relaxed").

ViStream supports interactions for monitoring mining results from multiple queries. Interactions for multiple query visual exploration include: First, analysts can re-arrange the layout of the pattern display to decide which queries to show and in which sequence. Second, analysts can manually group the queries with similar parameter settings for the purpose of shared computation or integrated visualization. Third, analysts can terminate existing queries, launch new queries and adjust parameter settings for active queries.



(a) juxtaposed view: colors indicate members of differnt clusters



(b) integrated view: colors indicate cluster members identified by differnt query sets

Figure 3: Jaxtaposed vs. Integrated Visualization.

5. VISTREAM DEMONSTRATION

In our demonstration, we will illustrate not only the efficient computational methods, but also the visualization and interaction mechanisms supported by *ViStream*. We will let the audience first hand experience its impact on the exploration of neighbor-based patterns in data streams. The demonstration will be based on two data streams, namely, the STT data recording stock transactions from NYSE and the GMTI data recording information about moving objects from MITRE. Demonstrations include, but are not limited to:

- 1) Visualization of proposed pattern space showing the detected patterns, including intensive transaction areas in NYSE stock trades and congestions formed by moving objects in the GMTI stream, at different abstraction levels and their evolution over time (Section 4). The audience will be able to not only observe these complex patterns in each window, but also track how they evolve (appear, split, merge and terminate).
- 2) Illustration of the strengths of juxtaposed and integrated visualizations for displaying various pattern types. (Section 4). The audience will be able to compare and contrast the patterns suggested or of their own interest using either technique.
- 3) Interactions on the pattern space including pattern space re-arrangement and adjustment to parameter settings (Section 4). The audience will be able to interact with the pattern mining process by re-arranging the layout of juxtaposed and integrated visualization, adjusting the query parameter settings, and so on.
- 4) Demonstration of the efficiency of our algorithms for extracting neighbor-based patterns and their evolution by contrasting them against alternative methods in [7] and [8] (Section 3). The audience will be able to experience the performance differences between alternative algorithms using a monitoring console.

6. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

[2] A. Bifet and R. Gavaldà. Mining adaptively frequent closed unlabeled rooted trees in data streams. In *KDD*, pages 34–42, 2008.

[3] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[4] M. C. Hao, U. Dayal, D. A. Keim, and T. Schreck. Multi-resolution techniques for visual exploration of large time-series data. *EuroVis*, pages 27–34, 2007.

[5] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *InfoVis*, 3(1):1–18, 2004.

[6] Z. Xie, M. O. Ward, and E. A. Rundensteiner. Exploring multivariate data streams using windowing and sampling strategies. *Interacting with temporal data workshop, CHI*, 2009.

[7] D. Yang, E. A. Rundensteiner, and M. O. Ward. Neighbor-based pattern detection for windows over streaming data. In *EDBT*, pages 529–540, 2009.

[8] D. Yang, E. A. Rundensteiner, and M. O. Ward. A shared execution strategy for multiple pattern mining requests over streaming data. *PVLDB*, 2(1):874–885, 2009.

[9] D. Yang, E. A. Rundensteiner, and M. O. Ward. A unified framework supporting interactive exploration of density-based clusters in streaming windows. *WPI Technical Report WPI-CS-TR-10-04*, 2010.