Sequence Synopsis: Optimize Visual Summary of Temporal Event Data

Yuanzhe Chen, Panpan Xu and Liu Ren

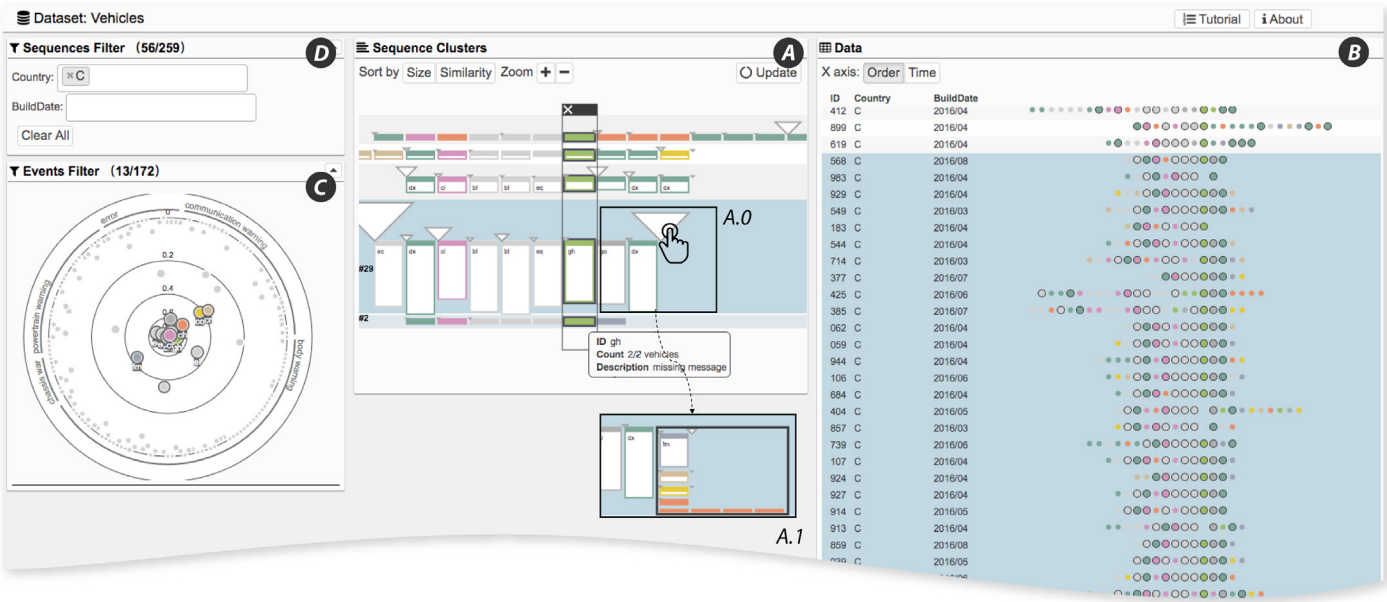

Fig. 1. A screenshot of the proposed visual analytics system for event sequence data exploration. The system contains an overview (A) which shows a set of sequential patterns that can best summarize the entire dataset based on the Minimum Description Length (MDL) principle. It also supports level-of-detail exploration (A.0 → A.1). A tabular display (B) shows the detailed information of the sequences linked with the summary view. Two panels (C and D) support data filtering. The event filter (C) shows the co-occurrence of events with a focus event at the center and allows users to select a set of highly correlated events. The sequence filter (D) supports sequence filtering based on their attribute values. The usage scenario in this figure is described in Section. 6.

Abstract— Event sequences analysis plays an important role in many application domains such as customer behavior analysis, electronic health record analysis and vehicle fault diagnosis. Real-world event sequence data is often noisy and complex with high event cardinality, making it a challenging task to construct concise yet comprehensive overviews for such data. In this paper, we propose a novel visualization technique based on the minimum description length (MDL) principle to construct a coarse-level overview of event sequence data while balancing the information loss in it. The method addresses a fundamental trade-off in visualization design: reducing visual clutter vs. increasing the information content in a visualization. The method enables simultaneous sequence clustering and pattern extraction and is highly tolerant to noises such as missing or additional events in the data. Based on this approach we propose a visual analytics framework with multiple levels-of-detail to facilitate interactive data exploration. We demonstrate the usability and effectiveness of our approach through case studies with two real-world datasets. One dataset showcases a new application domain for event sequence visualization, i.e., fault development path analysis in vehicles for predictive maintenance. We also discuss the strengths and limitations of the proposed method based on user feedback.

Index Terms—Time Series Data, Data Transformation and Representation, Visual Knowledge Representation, Visual Analytics

1 Introduction

Event sequence data, i.e., multiple series of timestamped or ordered events, is increasingly common in a wide range of domains. Website click streams, user interaction logs in software applications, electronic health records (EHRs) in medical care and vehicle error logs in automotive industry can all be modeled as event sequences. It is crucial to reason about and derive insights from such data for effective decision making in these domains. For example, by analyzing vehicle error logs, typical fault development paths can be identified, which can inform better strategies to prevent the faults from occurring or alert drivers in advance, and therefore improve driver experience and reduce warranty cost. Similarly, by analyzing users' interaction log with software applications, usability issues and user behavior patterns can be identified to inform better designs of the interface.

With the growing importance of event sequence analysis, a variety of visualization techniques have been proposed in the past years [5, 8, 9, 15, 23, 29, 33, 36, 41, 43, 52, 54, 55, 57, 60]. Recent

Yuanzhe Chen is with Hong Kong University of Science and Technology. E-mail: ychench@ust.hk. Panpan Xu and Liu Ren are with Bosch Research North America, Palo Alto, CA. E-mail: panpan.xu, liu.ren@us.bosch.com. This work was done while Yuanzhe Chen was an intern at Bosch Research North America, Palo Alto, CA.

Manuscript received 31 Mar. 2017; accepted 1 Aug. 2017.
Date of publication 28 Aug. 2017; date of current version 1 Oct. 2017.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2017.2745083

solutions further integrate sequential pattern mining (SPM) or sequence clustering techniques to facilitate sequential pattern identification from large and complex real-world data [21, 24, 26, 32, 48, 53].

However, despite great progress, it still remains a challenging task to create intuitive, simple, yet comprehensive overviews for real-world event sequence data. Methods such as Sankey diagrams [54] can not effectively handle noisy data with high event cardinality. SPM algorithms generate a long list of potentially redundant patterns and the analysts have to rely on certain interestingness metrics to prune or rank them in the visualization, which may result in partial coverage of the data and leave out some insights [21, 26, 32]. Visualization techniques based on sequence clustering can give an overview of the data [48, 53]. However the algorithms can produce clusters that are difficult to interpret.

In this paper, we describe a novel event sequence visualization technique using an *information-theoretic* approach. The goal is to construct a coarse-level overview of the data with a good balance between the simplicity of the visual representation and its information content. The approach we propose is based on a two-part representation of the data which consists of a set of sequential patterns and a set of corrections. The original sequences are mapped to the patterns and the corrections part specifies the edits (e.g., insertions and deletions of events) needed to transform the patterns to the individual sequences. The two-part representation can be regarded as a *lossless* compression of the data: the original sequences can always be fully recovered with the corresponding patterns and corrections.

Given a two-part representation, we visually summarize the original data with the sequential patterns and in the meanwhile model the information loss in the visualization with the corrections part. The challenge is to identify a set of patterns for a concise visual summary without introducing significant information loss. To tackle this challenge, we introduce the minimum description length (MDL) principle [13, 14]. The MDL principle is a general criterion for model selection in inductive inference which trades off between 1) the complexity of the model and 2) the description length of the original data with the help of the model. It aligns inherently with what we are trying to achieve with the overview, considering 1) the set of sequential patterns is the model and 2) the corrections describe the original data in combination with the set of patterns.

We develop efficient algorithms to identify a set of sequential patterns for optimized visual summary of data based on the MDL principle. The method supports simultaneous pattern extraction and sequence clustering. Since the method only imposes 'soft' pattern matching constraints [11], it is highly robust to handle noisy data with missing or additional events. Furthermore, it is also a generic framework that can incorporate various editing operations (e.g., swapping the positions of adjacent events). We further design a visual analytics system to support level-of-detail exploration of event sequence data with the identified patterns. We apply the method to two real-world datasets. One dataset showcases a new application domain for event sequence visualization, i.e., fault development path analysis in vehicles for predictive diagnostics. Another is a public dataset containing user interaction logs with a visualization tool [10]. It has been released to provide a common basis for evaluating event sequence visualization techniques.

To summarize, the main contribution of this work include:

- A generic two-part representation of event sequences consisting of a set of sequential patterns and a set of corrections. In combination with the MDL principle, the patterns can be used to construct informative overviews of data.
- Efficient algorithms to identify an optimal set of patterns to summarize the data based on the MDL principle.
- A visual analytics system that supports level-of-detail exploration of event sequence data.
- Case studies with real-world datasets and expert interviews which demonstrate the usability and effectiveness of the approach.

2 Related Work

2.1 Event Sequence Visualization

There is a large variety of event sequence visualization techniques. One straightforward approach is placing events or event episodes along a horizontal time axis as in Lifelines [33], CloudLines [20] and TimeSlice [59]. The method can reveal detailed information of each event. However, identifying temporal patterns in multiple sequences can be difficult for the substantial cognitive load to scan them simultaneously.

To tackle this issue, many visualization and interaction techniques have been proposed in the recent years. Lifelines2 [51] summarizes the frequency of events in different temporal granularities to help spot trends over time. EventFlow [29, 55], TrailExplorer [40, 41] and CoreFlow [25] extract and visualize tree-like branching structures from event sequence data. Lu et al. [27] extracts the sentiment trend of events as time series and visualizes them with stacked area charts. Outflow [54], CareFlow [31], DecisionFlow [12] and commercial products such as Google Analytics [1] condense event sequences into transition graphs where nodes represent events at different stages and edges connect successive events in the sequences. The graphs are usually visualized as Sankey diagrams to show the common transition pathways. A more recent approach, MatrixWave [60], uses matrix based visualization to display the graph, in order to avoid visual clutter caused by dense edges in Sankey diagrams.

Interaction is an essential part for event sequence data analysis. (S|qu)eries [58], COQUITO [19] and DecisionFlow [12] provide visual query interfaces to help users select a subset of sequences for focused analysis. Lifelines2 [50, 51] supports interactive alignment of data on selected events that the users can easily spot precursor, co-occurring, and aftereffect events. Wongsuphasawat and Shneiderman [56] and Du et al. [7] propose techniques for users to select a sequence and identify those similar to it based on certain distance metrics. Recently, Du et al. [8] also compiles a series of strategies for analyzing large datasets.

The techniques provide powerful analytic support for temporal pattern analysis. We also adopt existing interaction techniques in our system such as interactive alignment on selected events. However, with increased volume and complexity of the datasets, these techniques will have limited capability. Sankey diagram will suffer from visual clutter with an increasing number of transition paths. Visual query interfaces for reducing data volume and complexity can not provide full overview of the data and the analysts could miss important insights. We present an information-theoretic approach for event sequences summarization. The method reduces the visual clutter in the overview and in the meanwhile minimize the information loss in it. This allows analysts to identify salient patterns even within noisy and complex datasets. We further propose a novel visual representation that not only shows the sequential patterns, but also hints on the information missing from the display to guide the users in detail-on-demand exploration.

2.2 Event Sequence Mining and Visualization

Recently, an increasing number of visualization systems apply data mining techniques for event sequence data analysis.

SPM based methods applies frequency-based sequence mining algorithms and uses the identified patterns to guide event sequence data exploration. FP-Viz [44] is one of the early works that visualize the mined results with Sunburst visualization. TimeStitch [35] use the results of SPM models to help users discover, construct and compare cohorts in medical care data. Both Frequence [32] and Peekquence [21] directly visualize mined patterns to help users understand the data. Liu et al. [26] proposes a three-stage analytics pipeline to explore the patterns and sequences. The pipeline includes a pattern pruning algorithm which can filter redundant patterns mined from SPM models. Besides using automatic mining algorithms for patterns discovery, Vrotsou et al. [47] proposes an interactive approach for sequential pattern mining and visualization. SPM algorithms can generate large amount of patterns and the analysts usually need to rely on certain thresholds or interestingness metrics to make the result manageable and presentable. This may result in missing insights since the trimmed patterns can only give a partial view of the data. In comparison, our method gives an overview of the data by aggregating event sequences and identify the representative sequential pattern for each aggregated group.

Sequence clustering based techniques aggregates the data for an overview. LogView [28] uses treemaps to visualize the hierarchical clustering results while the sequential information is not directly

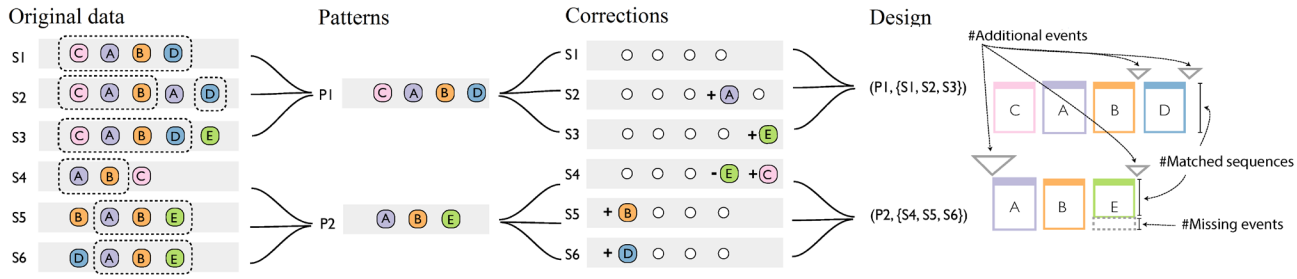

Fig. 2. An example two-part representation of multiple event sequences consists of 1) a set of patterns and 2) a set of corrections which can recover the original sequences from the corresponding patterns by inserting (+) or deleting (-) events. In the visual summary, the height of the rectangles is proportional to the number of sequences containing the corresponding event. Triangular glyphs encode the number of event insertions. Note that this is a *lossy* representation of the original data: it is not possible to recover the original sequences without detailed information about each edit. The size of the triangle glyphs and the height of the rectangles visually indicate the amount of information loss.

displayed. Cadez et al. [3] separates and visualizes sequences into different clusters for comparison. Wang et al. [48] uses unsupervised clustering on clickstream data and use packed circles to show the cluster hierarchy. Wei et al. [53] uses a self-organizing map to cluster and visualize clickstream data. However, the users still need to look at the original unaggregated data to understand why they form groups and verify the results. Our method supports simultaneous pattern extraction and sequence clustering. Each event sequence cluster is characterized with a representative sequential pattern which can greatly facilitate interpretation of the grouping results.

The MDL principle has been applied to construct [17] and evaluate [4] data models for event sequence analysis in data mining research. Our method combines the MDL principle with event data visualization by introducing a novel two-part representation and the corresponding algorithms to identify an optimal visual abstraction of the data.

3 MDL for Event Sequences

A high-level overview often plays a critical role in explorative data analysis, as emphasized in the well-known *visual information seeking mantra* "overview first, zoom and filter, details on demand" [42] and manifested in the design of numerous visualization systems. For event sequence data, an overview can serve as a starting point for descriptive analysis [34] and help users identify interesting patterns or subsets of data that are worth further exploration.

3.1 A Generic Method to Summarize Event Sequences

Our approach to visually summarize multiple event sequences is based on a two-part representation of the original data. The two-part representation consists of a set of sequential patterns and a set of corrections. Each event sequence is mapped to a pattern and the corrections specify the edits needed to transform the pattern to the original sequence. The edits may include insertion or deletion of events from the pattern. Fig. 2 gives an example. It shows six event sequences $\{S_1, S_2, ..., S_6\}$ together with the corresponding patterns and corrections. There are two sequential patterns P_1 and P_2. The original sequences can be reconstructed from either P_1 or P_2 by removing (-) or adding (+) events. For instance, S_2 can be recovered from P_1 by adding event A while S_3 can be recovered from P_1 by adding event E. S_4 can be recovered from pattern P_2 by removing event E and inserting event C. S_1 is exactly the same as pattern P_1, therefore no correction is needed.

The intuition of this two-part representation is to exploit the similarity of event sequences and identify a set of sequential patterns that can give a concise visual summary of the data. In the example in Fig. 2, P_1 and P_2 can roughly characterize the six sequences by representing $\{S_1, S_2, S_3\}$ and $\{S_4, S_5, S_6\}$ respectively. This is common in many application scenarios. For example, a series of interdependent faults can happen in the same sequential manner in multiple vehicles. Visitors of a commerce website may follow a similar sequence of pages to complete their orders.

The corrections part, on the other hand, specifies the information loss if the original data is visually represented by the sequential patterns. To reduce information loss, more elaborated patterns can be introduced. For example, in Fig. 2, the information loss can be reduced by mapping S_4 to a new pattern [A, B, C] instead of P_2. However, such changes can increase the visual complexity of the overview with more patterns to be displayed. The extreme case is when each individual sequence is treated as a pattern: there is no information loss, however severe visual clutter could occur when plotting all the sequences in a single visualization, making it much more challenging to identify high-level patterns. Essentially, we need to consider a trade-off between the readability of the visualization and the completeness of the information communicated through it.

3.2 The MDL Principle

We introduce the MDL principle [13, 14] to identify a set of sequential patterns for an overview of the data while balancing the information loss in it. MDL is a well known information criterion for statistical model selection. It has been adopted for constructing optimized layout of hierarchical visualizations [45]. The MDL principle basically states that the best model for a dataset results in minimized description length of it. Like most authors, we apply the more 'practical' or crude version of MDL instead of the ideal MDL. The ideal MDL tries to find the shortest program in a general-purpose computer language that can print the data. On the other hand, in the crude version, the description length of a dataset is composed of two parts: (a) the encoding of the model $L(\mathcal{M})$ and (b) the encoding of the data with the help of the model $L(\mathcal{D}|\mathcal{M})$. The best model $\hat{\mathcal{M}}$ should minimize the total description length, which is $L(\mathcal{M}) + L(\mathcal{D}|\mathcal{M})$.

For event sequences, we consider the sequential patterns as the model. The original sequences are coded by specifying the edits to the corresponding patterns. The total description length is the sum of the pattern lengths and the corrections length. The best set of sequential patterns that represents the original data should minimize the sum.

The MDL principle applied in this scenario is directly connected to the goal we intend to achieve in the overview. Simplifying the overview which shows the sequential patterns corresponds to minimizing $L(\mathcal{M})$, whereas $L(\mathcal{D}|\mathcal{M})$, the corrections length, is added to the objective function to penalize the information loss in it.

3.3 Denotations and Formal Problem Definition

Now we start to introduce the denotations and formally define the description length of the two-part representation. An event sequence is an ordered list of events $S = [e_1, e_2, ..., e_n]$ where $e_i \in \Omega$, an event alphabet. Given a set of event sequences $\mathcal{S} = \{S_1, S_2, ..., S_m\}$, the goal is to identify a set of patterns $\mathcal{P} = \{P | P = [e_1, e_2, ..., e_l]\}$ and a mapping $f : \mathcal{S} \to \mathcal{P}$ from the event sequences to the patterns that can minimize the total description length:

$$L(\mathcal{P}, f) = \sum_{P \in \mathcal{P}} L(P) + \sum_{S \in \mathcal{S}} L(S|f(S)) \quad (1)$$

In this equation, $L(P)$ is the description length of pattern P and $L(S|f(S))$ is the description length of S given its pattern $f(S)$.

Considering that 1) a pattern can be described by simply listing the events in it[1] and 2) an edit can be fully specified by the position and the event involved and its description length can be roughly treated as a constant, Eqn. 1 can be rewritten as:

$$L(\mathcal{P},f)=\sum_{P\in\mathcal{P}} len(P)+\alpha\sum_{S\in\mathcal{S}}\|edits(S,f(S))\|+\lambda\|\mathcal{P}\| \qquad (2)$$

where $len(P)$ is the number of events in the pattern and $edits(S,f(S))$ is a set of edits that can transform $f(S)$ to S. We further introduce the parameter α in Eqn. 2 to control the importance of minimizing information loss over reducing visual clutter in the overview, following the practice used by Veras and Collins [46]. The third term with the parameter λ is added to directly control the total number of patterns. Increasing λ will reduce the number of patterns in the optimized result. Therefore the scalability of the overview can be improved by setting λ properly.

The mapping f clusters the event sequences: sequences mapped to the same pattern can be considered as in the same cluster. We denote a cluster as a tuple $c=(P,G)$ where $G=\{S|S\in\mathcal{S}\wedge f(S)=P\}$ is the set of sequences mapped to pattern P. We denote the set of tuples for all the clusters as $\mathcal{C}=\{(P_1,G_1),(P_2,G_2),...,(P_k,G_k)\}$, where $\{G_1,G_2,...,G_k\}$ forms a partition of $\mathcal{S}$. Therefore finding $\hat{f}$ and $\hat{\mathcal{P}}$ that minimize $L(\mathcal{P},f)$ is equivalent to finding $\hat{\mathcal{C}}$ that minimize $L(\mathcal{C})$:

$$L(\mathcal{C})=\sum_{(P,G)\in\mathcal{C}} len(P)+\alpha\sum_{(P,G)\in\mathcal{C}}\sum_{S\in G}\|edits(S,P)\|+\lambda\|\mathcal{C}\| \qquad (3)$$

To summarize, our goal is to identify a partition/grouping of the sequences and a representative pattern for each group that can minimize the total description length. Conceptually it seems to be similar to sequence clustering algorithms. However the other methods do not follow an information-theoretic approach. Furthermore, the patterns can give an interpretable coarse-level summary of the original data, which is not possible with the existing sequence clustering techniques.

4 Computing MDL Representation

We introduce the algorithm to identify a grouping of the sequences and a representative pattern for each group that minimize $L(\mathcal{C})$ in Eqn. 3.

4.1 Basic Algorithm

We now present our first algorithm called *MinDL*. Since the optimization problem itself entails a rather large search space, we adopt a heuristic approach using a bottom up strategy. Initially, each sequence starts in its own cluster and is treated as a sequential pattern by itself. Starting from that, we iteratively merge pairs of clusters and compute the representative sequential patterns for the new clusters. The merges are determined in a greedy manner: the algorithm always choose to combine the pair that leads to the maximum description length reduction in each iteration. The algorithm stops when it can no longer find a pair to further reduce L.

MinDL is described formally in Algorithm 1. The algorithm is subdivided into two phases - *Initialization* and *Iterative merging*. In the *Initialization* phase, $\mathcal{C}$ is set to contain all the sequences in $\mathcal{S}$ as individual clusters. The algorithm then computes the description length reduction ΔL for all the pairs in $\mathcal{C}$ and uses a standard priority queue Q to store the pairs with a positive ΔL. With that the algorithm can efficiently retrieve the pair with the maximum ΔL in constant time. The subroutine *Merge* in line 4 returns not only ΔL by merging c_i and c_j, but also $c^*=(P^*,G_i\cup G_j)$ where P^* is the optimal sequential pattern for the merged group which minimizes the description length for the sequences in $G_i\cup G_j$. c^* is stored together with ΔL and (c_i,c_j) in Q to avoid recomputation.

[1] In this work, minimizing the description length is not an end goal, but a means to extract meaningful sequential patterns for a succinct visual display. Therefore we do not use compression schemes such as Huffman coding [38] to shorten the code lengths for the events and we consider the events are described with a constant length code. Same for encoding the edits.

Input: sequences $\mathcal{S}=\{S_1,S_2,...,S_n\}$
Output: pattern and cluster tuples $\mathcal{C}=\{(P_1,G_1),(P_2,G_2),...,(P_k,G_k)\}$

```
   /* Initialization phase                                          */
1  C = {(P,G)|P = S, G = {S} for all S ∈ S};
2  PriorityQueue Q = ∅;
3  for all pairs c_i, c_j ∈ C and i ≠ j do
4  |  ΔL, c* = Merge(c_i, c_j);
5  |  if ΔL > 0 then
6  |  |  insert (ΔL, c*, c_i, c_j) into Q;
7  |  end
8  end
   /* Iterative merging phase                                       */
9  while Q ≠ ∅ do
10 |  retrieve (ΔL, c*, c_i, c_j) from Q with the largest ΔL;
11 |  c_new = c*;
12 |  remove c_i, c_j from C, add c_new to C;
13 |  remove all pairs containing c_i or c_j from Q;
14 |  for c ∈ C − c_new do
15 |  |  ΔL, c* = Merge(c, c_new);
16 |  |  if ΔL > 0 then
17 |  |  |  insert (ΔL, c*, c, c_new) into Q;
18 |  |  end
19 |  end
20 end
21 return C
```

Algorithm 1: MinDL

In the *Iterative merging* phase, the algorithm picks the pair (c_i,c_j) with the maximum ΔL from Q. It updates $\mathcal{C}$ by removing c_i, c_j and inserting c^*. Q is updated by adding new pairs of clusters containing c^* with a positive ΔL. This process is repeated until Q is empty. The remaining tuples in $\mathcal{C}$ specify a grouping of the sequences along with the representative patterns which give an optimized description length of the original data.

The core subroutine in *MinDL* is *Merge*, which appears in line 4 and line 15. It is described in Algorithm. 2. *Merge* calculates the cost reduction ΔL when combining a pair of clusters $c_i=(P_i,G_i)$ and $c_j=(P_j,G_j)$. It also returns the sequential pattern P^* after merging. The algorithm initializes P^* as the Longest Common Subsequence (LCS) of P_i and P_j and iteratively add the remaining events in P_i and P_j to it, starting from those that appear most frequently in G_i and G_j. The iteration stops when ΔL no longer increases or is less than 0. The intuition behind this procedure is that the pattern P^* should be a mixture of P_i and P_j. Starting from the LCS of P_i and P_j can greatly reduce the efforts needed to build the sequential pattern P^* from scratch.

The function *edits* in line 7 calculates the *minimum* number of edits that can transform a pattern P to a sequence S. Different types of edits can be supported in the algorithm, given that the minimum number of edits are computed accordingly. For example, if we allow missing or additional events in the pattern, the minimum number of edits can be obtained by computing the LCS distance between P and S. Adding event substitution into the available types of edits, we get Levenstein distance. The algorithm can support even more editing types such as swapping the positions of adjacent events. In this paper we mostly consider event insertion and deletion as the available editing operations. In Section 4.3 we use an example to illustrate how swapping adjacent events can be supported in the same framework.

4.2 Speedup with Locality Sensitive Hashing (LSH)

An analysis on the time complexity of *MinDL* shows that it can be quite time consuming even for a moderate amount of data. Given n event sequences, the *MinDL* algorithm runs the subroutine *Merge* for $O(n^2)$ times to calculate ΔL for all pairs of clusters. The subroutine *Merge* itself has a time complexity of $O(kmd^2)$, where m is the number of sequences in the combined cluster, k is the number of iterations in the pattern buildup phase (line 5-13 in Algorithm. 2) and d is the

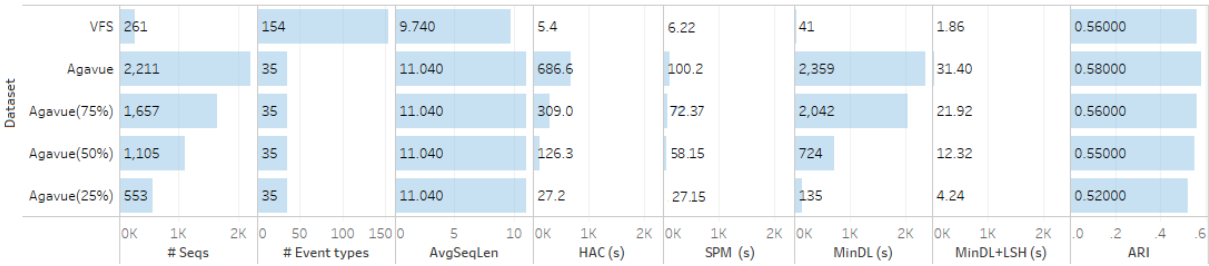

Fig. 3. Algorithm performance comparison on two real-world datasets, vehicle fault sequences (VFS) and Agavue [10]. We sample the Agavue dataset to create test data with different number of sequences. We run algorithms on a PC with 2.5GHz Intel dual-core i5 CPU with 4GB RAM. The algorithms are implemented in Python except that HAC in scikit-learn and weighted LSH in datasketch use external C libraries.

Input: $c_i = (P_i, G_i), c_j = (P_j, G_j)$
Output: ΔL and $c^* = (P^*, G_i \cup G_j)$ by merging c_i and c_j
/* Initialization phase */
init pattern $P^* = P = LCS(P_i, P_j)$;
candidate events $E_c = P_i - P \cup P_j - P$;
sort E_c by frequency in desc order;
$\Delta L = -1$;
/* Pattern buildup phase */
for e *in* E_c **do**
  $P = Add(P, e)$;
  $\Delta L' = len(P_i) + len(P_j) - len(P) + \alpha \sum_{S \in G_i} edits(S, P_i) + \alpha \sum_{S \in G_j} edits(S, P_j) - \alpha \sum_{S \in G_i \cup G_j} edits(S, P) + \lambda$;
  if $\Delta L' < 0$ *or* $\Delta L' < \Delta L$ **then**
    break;
  else
    $\Delta L = \Delta L'$, $P^* = P$;
  end
end
return $\Delta L, c^* = (P^*, G_i \cup G_j)$

Algorithm 2: Merge

average length of the sequences. We assume the minimum number of edits can be computed efficiently through dynamic programming, hence the time complexity of computing *edits* is $O(d^2)$.

To tackle this challenge, we propose a fast randomized approximation algorithm utilizing *Locality Sensitive Hashing (LSH)* [22]. The intuition of this approach is that pairs of sequences/patterns which share very few common events or no common event at all (regardless of the order) can be skipped when searching for candidate pairs to merge. If we can design a method that can quickly filter out such pairs, the times of calling function *Merge*, the most time consuming routine, can be significantly reduced.

Based on this observation, we integrate weighted LSH [16] into the *MinDL* algorithm. Weighted LSH takes a predefined threshold th within the range $(0.0, 1.0)$ as a parameter. If two multisets have a weighted Jaccard similarity larger than th, they will have the same hash value with a sufficiently high probability. We use weighted LSH to quickly identify pairs of sequences/patterns with similar sets of events regardless of their exact order. This allows us to prioritize the clusters when searching for candidates to merge.

When applying LSH, a higher threshold th can filter out more candidates and make the algorithm faster. However, the risk of missing potential candidates also becomes higher. We follow the strategy proposed by Koga et al. [18] and run *MinDL* for multiple iterations while gradually decreasing th. In the first few runs, we use higher th so each time fewer pairs need to be checked in *MinDL*. To ensure no possible merge is missed due to the usage of LSH, we gradually decrease the threshold in the later runs such that more candidate pairs could be considered. Since the number of clusters decreases quickly during the first few runs, this method still can significantly reduce the running time. In this work, the threshold setting is guided by the experimental result.

The *MinDL+LSH* algorithm is described in detail in the Appendix.

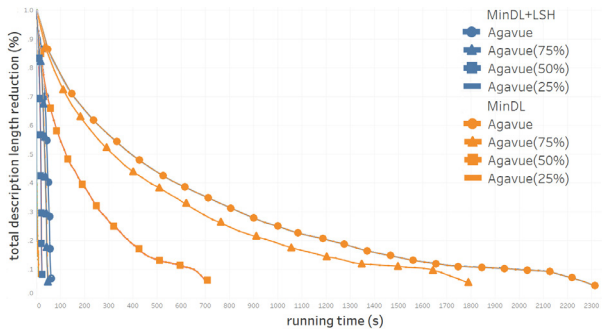

Fig. 4. Change of the description length ($L(\mathcal{C})$) over time as *MinDL* and *MinDL+LSH* progress, tested on the Agavue dataset. $L(\mathcal{C})$ is normalized with its initial value when each sequence is treated as an individual pattern.

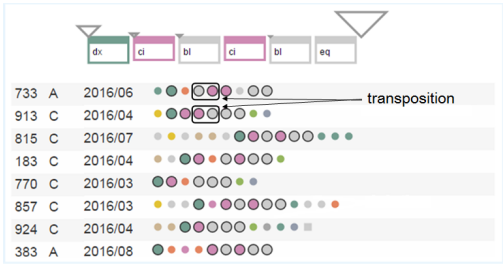

Fig. 5. An example sequence cluster where events may have different orders when compared to the pattern. By adding transposition operation in possible edits, the algorithm allows small perturbations in event order.

We conduct experiments to evaluate the effectiveness of the speedup strategy on two datasets, vehicle fault sequences (VFS) and Agavue [10]. Detailed information of the two datasets are described in Section. 6. Some basic statistics about the datasets and the experimental results are displayed in Fig. 3. It can be observed that we can reduce 95% to 99% running time of *MinDL* with the help of LSH. We also test the running time of a standard clustering algorithm *Hierarchical Agglomerative Clustering (HAC)* and a SPM algorithm [49]. For fair comparison, we use editing distance as the metric in the HAC algorithm, and use the minimum cluster size in our *MinDL* algorithm as the support threshold in the SPM algorithm. From the result, we can observe that *MinDL+LSH* is much faster than both *HAC* and *SPM*, especially when number of sequences becomes larger.

Fig. 4 compares *MinDL* and *MinDL+LSH*. It shows how the total description length L decreases over time as the two algorithms progress. Besides the dramatic difference in the decreasing speed, we also observe that the resulted description length is similar for the two

algorithms. This means that the two algorithms can achieve similar optimization results. To further validate this conclusion, we also compare the clustering results before and after embedding LSH with the Adjusted Rand Index (ARI). ARI is a common metric to compare clustering results. It ranges from -1 to 1 where 0 means random clustering and 1 means identical results. An ARI larger than 0.5 means that the results are very similar [39]. Fig. 3 shows that all the results have an ARI larger than 0.5. Therefore, we can safely conclude that adding LSH does not have significant negative effect on the clustering results.

4.3 Soft Pattern Matching

In the algorithm, the individual sequences may deviate from the patterns with missing or additional events, given that they do not add too much to the corrections part. Therefore the method is quite robust to noises, common in real-world data. The algorithm is also generic and can support more editing operations such as swapping the positions of adjacent events, thus allowing small perturbations in event order. Fig. 5 shows an example when we include insertion, deletion and transposition between two successive events as the possible edits. In the algorithm, the minimum number of edits can be determined by following the method to calculate the Damerau-Levenshtein distance [2]. Fig. 5 shows that the events in the patterns appear in the individual sequences in different order.

5 The Visual Analytics System

5.1 Analysis Tasks

In a recent paper, Plaisant and Shneiderman [34] summarize a set of high-level analytic tasks for event sequence data. To identify the most common tasks across various application domains in order to design a generic tool for event sequence analysis, we survey design studies for different kinds of data (e.g., website click streams and EHR data) and gather requirements from experts in vehicle data analytics, the new application domain we introduce in this paper. Table. 1 (Appendix) summarizes the result of the survey and the expert interview. We conclude the four high level tasks ***T1, T2, T5, T7*** to be the most common ones and centered our design around these tasks. The four tasks are:

T1. Review in detail a few records.
T2. Compile descriptive information about the dataset or a subgroup of records and events (esp. through aggregated views).
T5. Identify a set of records of interest.
T7. Study antecedents or sequelae of an event of interest.

We design the system to support the aforementioned tasks while following the general guideline of showing multiple levels of detail [42]. Starting from an overview of the sequential patterns (***T2***), the analyst can identify a subset for further investigation (***T1***). The analyst can also filter records by their attribute values or filter events by their co-occurrences (***T1***, ***T2***, ***T5***). The system also supports interactive alignment on a selected event to study its causes and effects (***T7***).

5.2 Event Filter

The event filter (Fig. 1 (C)) shows the events' co-occurrences in the sequences and allows users to select a few highly interdependent ones for further study. Similar to the design by Chuang et al. [6], we show explicitly the co-occurrence of all the events with a focus event in the visualization. The co-occurrence is measured by Jaccard Index and is encoded as the radial distances to the focus event at the center of the display. The analyst can change the focus interactively and the distances will change accordingly. The sizes of the circles represent how frequent the events occur overall. The events are arranged around the circle based on their category. The radial angles separate different categories of events as in [6]. The categorical labels are displayed along the sectors.

The color of the cirle encode the type of the event. Using color to encode event type is a common practice in event sequence visualization [26, 29]. It is also proved as relatively effective in a recent study [37]. The color encoding is shared across multiple views for consistency.

In the visualization, events that frequently co-occur with the focus event are close to the center. The analyst can use a lasso tool to select a set of highly relevant events and focus on the sequential patterns containing those events.

An alternative way to visualize the events' interdependency is Multidimensional Scaling(MDS), which can project the events to a 2D plane based on their co-occurrences. We use the radial design to show *undistorted* distances to a *focus* event. By observing the radial distance to the center, users can easily estimate the frequency of co-occurrence between any event and the focus event. Compared with the MDS layout, it can display more accurate and interpretable information to the analysts. Besides that, the radial layout is also suitable when the analyst wants to focus on a particular type of event.

5.3 Summary View

In the summary view (Fig. 1 (A)), we vertically list all the sequential patterns identified by the algorithm. Each pattern represents a cluster of sequences. For each pattern, we layout the events from left to right and display them as rectangles. The color of the rectangles encodes the type of the event.

Besides displaying the sequentially ordered events in the patterns, the summary view also shows the number of edits in the corrections part. Fig. 2 illustrates the visual encoding in the summary view. Triangular glyphs are placed between adjacent events or at the beginning/end of the pattern. Their sizes are proportional to the number of insertions at the corresponding position, accumulated over all the sequences in the cluster. The height of the rectangles is proportional to the number of sequences containing the corresponding event in the pattern. It implicitly shows the deletions as the 'missing' parts compared to the others. The event insertions and deletions are obtained by backtracking the dynamic programming algorithm which computes the minimum editing distances between the individual sequences and the patterns.

The design itself is simple in the sense that at most $O(\sum_{(P,G)\in\mathcal{C}} len(P))$ visual elements appears in it (counting the rectangles and triangles). It is a *lossy* representation of the original data since detailed information of each edit is missing and it is not possible to recover all the original sequences with this visual representation. The sizes of the triangles and the heights of the rectangles visually indicate the amount of information loss. This design also helps viewers identify clusters with high/low intra-cluster similarity, which can guide them to a more detailed exploration of the data. For example, Fig. 1 (A.0 $\rightarrow$ A.1) shows how the user can expand a triangle and get a summary view of the subsequences in it. One potential drawback of this design is that missing events are not explicitly encoded. In certain application scenarios (e.g. EHR data analysis), missing events may also need to be highlighted with additional visual cues.

5.4 Sequence View

The sequence view (Fig. 1 (B)) organizes the detailed information of each record in a tabular form. The attribute values and the original event sequences are displayed. The events are placed along a horizontal axis. Each event is represented by a glyph. The events matched in the patterns are displayed in larger sizes. Event sequences in the same cluster are placed together.

5.5 User Interaction

User interactions are designed to support the exploration of event sequence data. We summarize the interactions into three categories.

Basic interactions. Filtering, tooltip and linked-highlighting are the three basic interactions supported in our system. The system support two types of filtering. First, as mentioned in Section 5.2, users can filter events with the lasso selection tool in the event filter (Fig. 1 (C)). The analysts can also filter the sequences through the attribute values as shown in Fig. 1 (D). Note that the event filter will always be updated accordingly to reflect the co-occurrences of events in the filtered sequences in Fig. 1 (D). The filtering function is especially helpful when the analysts have prior knowledge about what kind of sequences or events worth further analysis. A tooltip is designed to show the detailed description of each event when analysts hover over any event in the system. Furthermore, linked-highlighting is supported to help users associate the information displayed in the detailed view (Fig. 1 (B)) with the summary view (Fig. 1 (A)). When users click and

highlight patterns in the summary view, individual sequences in the detailed view will be ordered and highlighted accordingly.

Detail on demand. As mentioned in Section 5.3, inserted events are aggregated and visualized as triangular glyphs in the summary view. However, users may still want to examine the details, especially when the size of the glyphs indicate that there are many inserted events. In the prototype, the users can double click the glyphs to expand them for detailed analysis. To be more specific, since the inserted events are also a set of subsequences, we apply the same summarization approach in Section 4 to these subsequences and show a set of patterns and corrections in the expanded view. Fig. 1 (A.0 $\rightarrow$ A.1) shows an example of such expansion.

Temporal relation exploration. To support efficient temporal relation exploration, we allow users to align sequences at a selected event. By default, the sequences in summary view and the detail view are aligned at the first event. Users can select one event in the summary view and both views will be aligned to the selected event through animated transition. Fig. 1 (A, B) shows an example where the events are aligned at the event 'gh'. In this way, users can easily identify subsequences occur before and after a given event. Besides, the horizontal scale in the detailed view can be changed to show accurate temporal information instead of only sequential orders. By changing the scale, the users can analyze the temporal distribution of events. Users can also combine alignment and changing the horizontal scale in the system. In this way, they can easily observe the how other events distribute with respect to a selected event. Fig. 6 shows an example.

The system also has other interactive features such as reordering the patterns in the summary view. The analyst can sort the sequential patterns by 1) the number of sequences in the corresponding cluster and 2) the similarity between the patterns measured through the editing distance. To reorder by similarity, we first perform a hierarchical clustering of the patterns and then sort them by the leaf order.

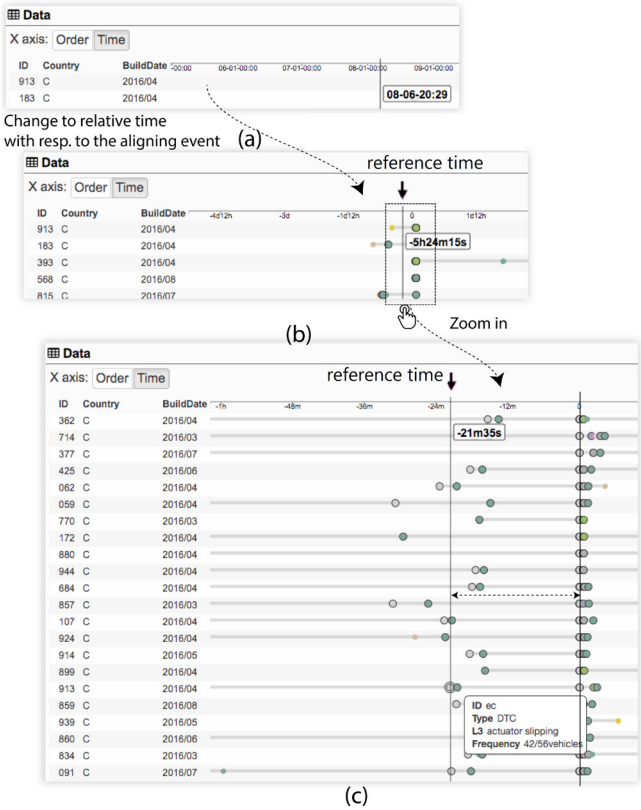

Fig. 6. Switching X axis to timestamps. (a) by default absolute time is displayed (b) aligning at an event changes the X axis to relative time (c) zooming in on the timeline shows that most events in the pattern occurred within a 20 minutes time range.

6 Example Usage Scenarios

We present example usage scenarios with real-world datasets from two application domains to showcase the utility of our approach.

6.1 Vehicle Fault Analysis for Predictive Diagnostics

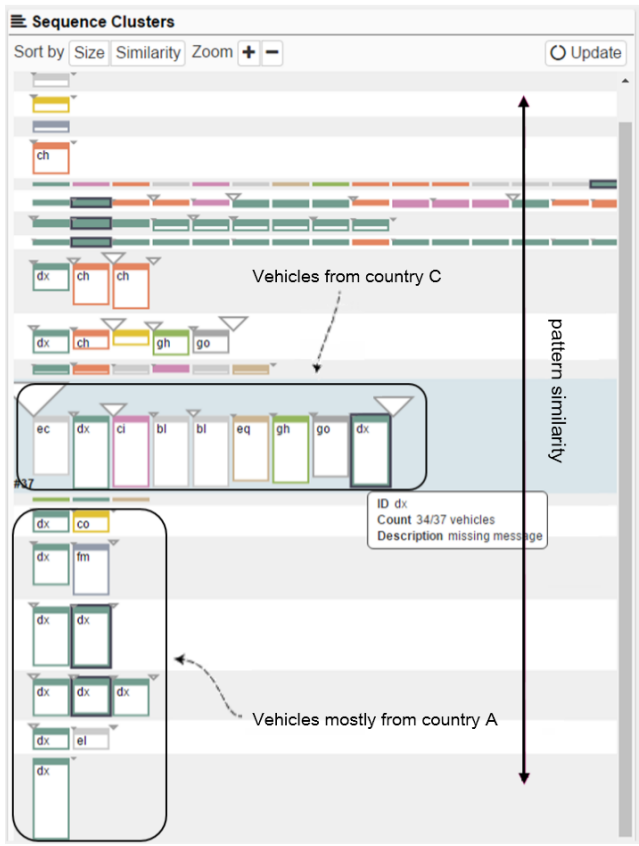

Fig. 7. Visual summary of all sequence data. A cluster (A) with a similar pattern compared to the dominant one in Fig. 1. It contains vehicles sold in country C. Some other clusters (B) contain vehicles sold in country A.

Our first usage scenario involves an expert in the automotive industry. The expert is interested in vehicle data analytics, especially analyzing the development paths of faults in vehicles. We conduct the case study together with the expert. Today's vehicles are complex machines with interconnected modules and the faults have a significant history of development over the vehicles' lifetime. Understanding that history help with predictive diagnostics, i.e., prevent the fault from occurring or mitigate its effects in advance. Eventually this could improve the driving experience and lower the warranty cost for the car manufacturers.

The fault events in vehicles are automatically recorded along with the timestamp information. We obtain a sample dataset (VFS) from the expert. The dataset contains the fault sequences of 261 vehicles together with information such as their vehicle identification numbers(VINs), build dates and the countries they were sold to. The data is collected in one year. In total we count 154 different types of faults. The average length of the event sequence is 9.74. The maximum length is 145. Each fault also has an associated timestamp. The VIN number, the description of the events and the country names are anonymized for privacy concerns.

Data filtering. The analyst started the analysis by filtering sequences. Since the analyst was particularly interested in the vehicles sold to Country C, she got a subset of the data by selecting Country C in the sequence filter (Fig. 1 (D)). The event filter shows that most of the frequently occuring faults are close to the focus event at the center (Fig. 1 (C)). With the lasso tool, the analyst selected these events for further study. The summary view was updated to show the patterns

within the filtered data. It could be observed that there is a dominant cluster with a pattern of 9 events, as highlighted in Fig. 1 (A).

Temporal relation exploration. To further investigate the temporal distributions of the events in the pattern, the analyst switched the X axis in the sequences view to accurate timestamps (Fig. 6 (a)). By default the visualization shows the absolute time, i.e., the exact date and time of the events. To further study how the cause and effect relationship took place over time, the analyst aligned all the sequences at event *gh* in the pattern (Fig. 6 (b)). This changed the X axis to relative time with respect to 'gh'. A reference axis is shown with the movement of the mouse to indicate the time gap between the reference bar and the aligned event. After zooming in the X axis Fig. 6 (c), it could be observed that the events all happened within a short time range (around 20 minutes), indicating a causal relationship that took effect pretty fast.

Detail-on-demand. The summary view shows that quite a few events happened after the pattern ends (Fig. 1 (A.0)). The analyst therefore double clicked on the triangle to look into the next level-of-detail. Fig. 1 (A.1) shows that the corrections part actually contained a large proportion of subsequences with error *fm*. It could be hypothesized that *fm* is also closely related to the events included in the sequential pattern, although it may not have happened yet for some of the vehicles in the cluster.

Insight validation. Now the analyst was curious about whether vehicles sold to other countries also exhibit the same sequential fault pattern. She cleared the filtering conditions and included all the vehicles in the analysis. The summary view (Fig. 7) shows the updated results and order the patterns by their similarity. The analyst observed that there is a cluster with the same sequential pattern when compared to the major cluster in Fig. 1. Hovering over the cluster also highlights the corresponding entries in the table, where the analyst observed that all the vehicles within the cluster were sold in country C. Meanwhile, the analyst also found that there is another group of clusters which contains vehicles mostly sold in country A. This observation leaded to further hypothesis about the potential root causes of these faults, such as the climate characteristics in different geographic areas or faulty parts used in producing the particular batches of cars.

6.2 Application Log Analysis for UI Design Optimization

Our second usage scenario is application log analysis. Desktop or web applications can collect large amount of usage log data recording user interactions and many other events in the system. Log data analysis has the potential to provide important insights about users' behavioral patterns and help optimize the user interface design.

We use a public dataset named Agavue [10]. The dataset logs the user interactions and function calls in a data visualization application in Excel. The sample dataset contains 2211 unique user sessions and 35 distinct event types. An additional preprocessing step is used to merge adjacent events of the same type. After preprocessing, the average sequence length is 11.04 and the maximum sequence length is 146.

Overview. The analyst started with an overview of the data (Fig. 8) and aligned the patterns at the *appInit* event. Not surprisingly, most patterns (e.g., Fig. 8 (A, B)) contain a typical sequence of operations including initializing the app (*appInit*, *create*), resizing the window, binding data (*bindFromPrompt*, *readBoundData*). The analyst can click on the patterns to review the sequences in each group (Fig. 8 (C, D)). Pattern B represents a group of sequences with better consistency, indicated by the smaller sizes of the triangles. Pattern A represents a group of sequences that are consistent in the first few events however have more significant deviations afterwards. This observation can be easily verified by looking at the detailed views (Fig. 8 (C, D)).

Cause and effect relation analysis. Since error messages popping up in an app can interrupt the users' analytic workflow and have a negative effect on user experience, it is important to understand the context in which the error messages occur and based on that, redesign the application to reduce the error messages if possible. To this end the analyst aligned the patterns at the error event (Fig. 9) to study its antecedents. She observed that most errors occur after users trying to bind data to the visualization (*bindFromPrompt*). One possible explanation is that the users may not be familiar with the data format requirements associated with the visualizations. This observation indicates that better interface for data binding can be designed to further improve user experience.

7 Expert Interview

We demonstrated how the prototype could be applied to analyze the vehicle fault sequence data to three groups of analysts from the automotive industry. The analysts all dealt with similar data in their daily work and they were very familiar with the usage scenario. One group of analysts was interviewed remotely and interacted with the system through our web server. The other two interviews were conducted face to face. For each interview, we first introduced the visual designs and the interactions in the system, and then asked analysts to explore the system on their own for about half an hour. After that, we had discussion sessions with the domain experts focusing on three different aspects of the system, e.g., system usability, required additional features and other potential uses (besides vehicle fault analysis) of the system. The analysts commented positively on the system and were intrigued by the idea of fuzzy pattern matching and sequence clustering. Most of the experts think that one of the most powerful features in the system is the interactive alignment of the sequence clusters. Furthermore, one analyst commented that "the system shows clearly the seriousness of some faults as it might later lead to other faults [based on the summary view and the detailed view]", "the correlation among the faults are very clear to see [in the radial graph]" and "with more data it would be a powerful tool to spot patterns of fault occurrences". Seeing the great potential value of the system, the analysts have already arranged follow-up discussions with us about offering the visual analytics solution as part of their vehicle data analytics software.

Besides that, the analysts also requested additional features in the system. For example, now the system only supports aligning on a single event and they recommended to generalize this feature to support aligning at two or even more events to identify what happened between those anchor points.

One analyst mentioned that vehicles from many car manufacturers record error logs in the same manner. Therefore, the system could benefit different car brands. The analyst pointed out that although the current system was demonstrated with a small sample dataset, the features in the system could become more powerful with large scale data.

During the demonstration we also mentioned that the algorithm and the system were generic and could be used to analyze other datasets such as website click streams/application logs as well. One analyst immediately recalled that they also collect click stream data for vehicle diagnostics software used in repair shops and suggested that "the system can help optimize the interface, [and] shorten the time [for the repairers] to find information". After that he/she asked for further follow-up to fully assess the feasibility of this approach and showed great interest to also continue pursuing this particular usage scenario. This demonstrated that the principle underlying the system can be easily grasped and it has the versatility to be adapted to different application scenarios.

8 Limitations and Future Work

Scalability. The current visual design of the summary view can display 20∼30 patterns without too much visual clutter. In the experiment, we tested a dataset containing up to about 2200 individual sequences and the result shows that dozens of patterns can effectively summarize the data. However some datasets may inherently contain more distinct patterns and the proposed approach could suffer from scalability issues. Further experiments need to be conducted to assess the applicability of the framework to large datasets. Besides that, one way to improve the scalability of the system is to extend the current framework to support hierarchical visual summary of event sequences. For example, starting from a high-level overview, users can select one cluster and split it into several low-level clusters. To support this extension, the algorithms need to maintain a hierarchical structure of the data, and the system features also need to be adjusted to support smooth user interaction.

Multiple patterns. The current framework only supports matching individual event sequences to a specific sequential pattern. In real-world applications, an event sequence may contain several patterns, especially when it is very long. We plan to tackle this issue from two

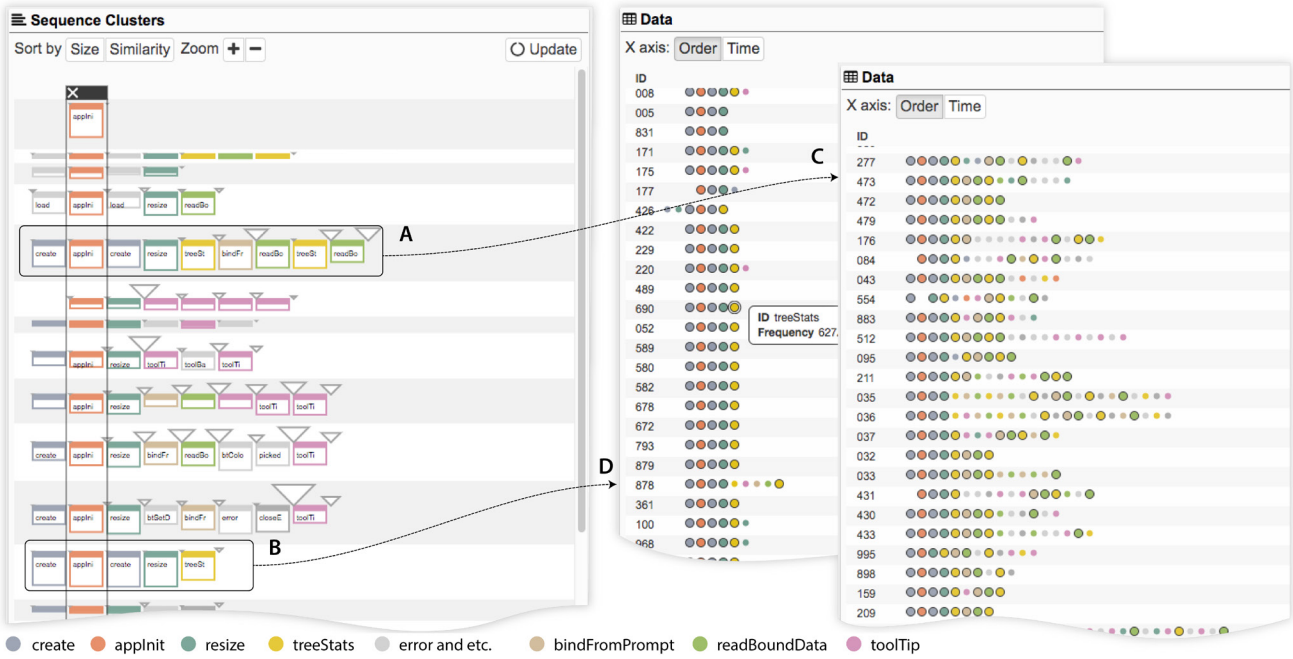

Fig. 8. The system screenshot for analyzing the Agavue dataset. Most patterns (e.g., A and B) show typical sequence of operations including initializing the app (*appInit*, *create*), resizing the window, binding data (*bindFromPrompt*, *readBoundData*, *treeStats*). Pattern B represents a more homogeneous group of sequences (the triangles are quite small). Pattern A represents a group of sequences that is more similar in the first few events however has more significant deviations later.

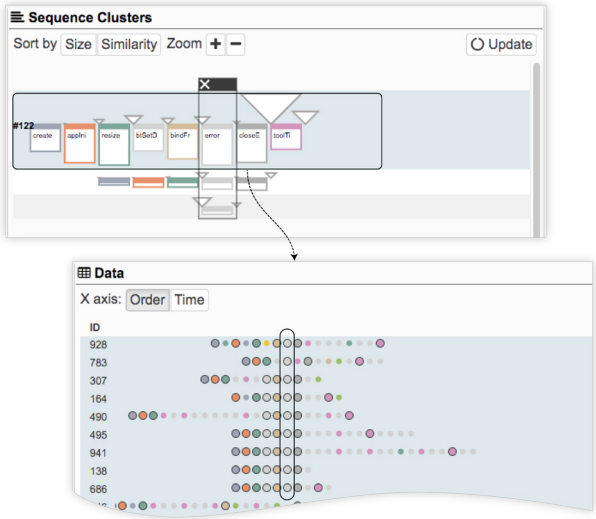

Fig. 9. When aligned at the *error* event, the summary view shows the most frequent antecedents (binding data) and sequelae (close error message window).

directions. One direction is to extend the two-part representation and support matching individual sequences to multiple sequential patterns. Another direction is to design algorithms or support user interactions to segment long sequences into meaningful shorter ones and use the subsequences as input to the MDL framework.

Event importance. Although we treat all the events equally in the paper, it is common in real-world usage scenarios that some events are more critical compared to the others. For example, some faults in the vehicles may indicate failures in engine start and would require immediate repair services whereas others might not have major effect on vehicle operation. In such cases aggregating the critical events into the triangle may not be a good option. We will continue to explore the potential approaches to signify such differences in event importance.

Pattern query. In many usage scenarios, users are interested in certain sequences based on domain knowledge. To improve the usability of our system and keep users in the loop, the system should support pattern query and present the queried patterns in the summary view.

General MDL based visual summary. The method proposed in the paper addresses a fundamental trade-off in visualization design: reducing visual clutter vs. increasing the information content in the visualization. We believe our approach successfully showcases how this trade-off can be directly quantified and optimized to construct an informative and concise overview of the data. We envision that similar approaches can also be designed for other types of data, e.g., graphs/networks and time series data. For example, Navlakha et. al. [30] has applied MDL for graph summarization. However, they did not address the tradeoff between the visual clutter and information content. For visualization community, there is still a lot of room for exploration.

9 Conclusion

In this paper, we present a novel visual analytics approach to visualize event sequence data. First, we propose an information-theoretic method based on the minimum description length principle to construct an overview of data. This method can extract sequential patterns and cluster event sequences simultaneously. We further demonstrate that it supports soft pattern matching and it is a generic approach that can incorporate different editing operations. We then propose a comprehensive visual analytics system with multiple levels-of-detail to facilitate interactive data exploration. We also conduct case studies with two real-world dataset and collect feedback from end users to demonstrate the effectiveness of the proposed approach. We also introduce a new application domain for event sequence visualization, which is fault development path analysis for predictive diagnostics in vehicles.

Acknowledgments

We would like to thank Kelsey Hoggard for supporting the video editing. We would also like to thank the VAST reviewers for their valuable comments. This work is also supported by RGC GRF 16208514.

REFERENCES

[1] Google analytics. https://analytics.google.com/.

[2] E. Brill and R. C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pp. 286–293. Association for Computational Linguistics, 2000.

[3] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Model-based clustering and visualization of navigation patterns on a web site. *Data Mining and Knowledge Discovery*, 7(4):399–424, 2003.

[4] T. Calders, C. W. Günther, M. Pechenizkiy, and A. Rozinat. Using minimum description length for process mining. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 1451–1455. ACM, 2009.

[5] N. Cao, Y.-R. Lin, F. Du, and D. Wang. Episogram: Visual summarization of egocentric social interactions. *IEEE computer graphics and applications*, 36(5):72–81, 2016.

[6] J. Chuang, D. Ramage, C. Manning, and J. Heer. Interpretation and trust: Designing model-driven visualizations for text analysis. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pp. 443–452. ACM, New York, NY, USA, 2012.

[7] F. Du, C. Plaisant, N. Spring, and B. Shneiderman. Eventaction: Visual analytics for temporal event sequence recommendation. *Proceedings of the IEEE Visual Analytics Science and Technology*, 2016.

[8] F. Du, B. Shneiderman, C. Plaisant, S. Malik, and A. Perer. Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus. *IEEE Transactions on Visualization and Computer Graphics*, PP(99):1–14, 2016.

[9] J. A. Ferstay, C. B. Nielsen, and T. Munzner. Variant view: Visualizing sequence variants in their gene context. *IEEE transactions on visualization and computer graphics*, 19(12):2546–2555, 2013.

[10] D. Fisher. Agavue event data sample: Full dataset. version of october 20, 2016. microsoft research. retrieved from http://eventevent.github.io.

[11] D. Gotz. Soft patterns: Moving beyond explicit sequential patterns during visual analysis of longitudinal event datasets. In *Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis*, 2016.

[12] D. Gotz and H. Stavropoulos. Decisionflow: Visual analytics for high-dimensional temporal event sequence data. *IEEE transactions on visualization and computer graphics*, 20(12):1783–1792, 2014.

[13] P. Grunwald. A tutorial introduction to the minimum description length principle. *arXiv preprint math/0406077*, 2004.

[14] P. D. Grünwald. *The minimum description length principle*. MIT press, 2007.

[15] S. Haroz, R. Kosara, and S. L. Franconeri. The connected scatterplot for presenting paired time series. *IEEE transactions on visualization and computer graphics*, 22(9):2174–2186, 2016.

[16] S. Ioffe. Improved consistent sampling, weighted minhash and l1 sketching. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pp. 246–255. IEEE, 2010.

[17] J. Kiernan and E. Terzi. Constructing comprehensive summaries of large event sequences. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(4):21, 2009.

[18] H. Koga, T. Ishibashi, and T. Watanabe. Fast agglomerative hierarchical clustering algorithm using locality-sensitive hashing. *Knowledge and Information Systems*, 12(1):25–53, 2007.

[19] J. Krause, A. Perer, and H. Stavropoulos. Supporting iterative cohort construction with visual temporal queries. *IEEE transactions on visualization and computer graphics*, 22(1):91–100, 2016.

[20] M. Krstajic, E. Bertini, and D. Keim. Cloudlines: Compact display of event episodes in multiple time-series. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2432–2439, Dec 2011.

[21] B. C. Kwon, J. Verma, and A. Perer. Peekquence: Visual analytics for event sequence data. In *ACM SIGKDD 2016 Workshop on Interactive Data Exploration and Analytics*, 2016.

[22] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[23] S. Liu, W. Cui, Y. Wu, and M. Liu. A survey on information visualization: recent advances and challenges. *The Visual Computer*, 30(12):1373–1393, 2014.

[24] Z. Liu, H. Dev, M. Dontcheva, and M. Hoffman. Mining, pruning and visualizing frequent patterns for temporal event sequence analysis. In *Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis*, 2016.

[25] Z. Liu, B. Kerr, M. Dontcheva, J. Grover, M. Hoffman, and A. Wilson. Coreflow: Extracting and visualizing branching patterns from event sequences. 2017.

[26] Z. Liu, Y. Wang, M. Dontcheva, M. Hoffman, S. Walker, and A. Wilson. Patterns and sequences: Interactive exploration of clickstreams to understand common visitor paths. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):321–330, 2017.

[27] Y. Lu, M. Steptoe, S. Burke, H. Wang, J.-Y. Tsai, H. Davulcu, D. Montgomery, S. R. Corman, and R. Maciejewski. Exploring evolving media discourse through event cueing. *IEEE transactions on visualization and computer graphics*, 22(1):220–229, 2016.

[28] A. Makanju, S. Brooks, A. N. Zincir-Heywood, and E. E. Milios. Logview: Visualizing event log clusters. In *Privacy, Security and Trust, 2008. PST'08. Sixth Annual Conference on*, pp. 99–108. IEEE, 2008.

[29] M. Monroe, R. Lan, H. Lee, C. Plaisant, and B. Shneiderman. Temporal event sequence simplification. *IEEE transactions on visualization and computer graphics*, 19(12):2227–2236, 2013.

[30] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pp. 419–432. ACM, 2008.

[31] A. Perer and D. Gotz. Data-driven exploration of care plans for patients. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pp. 439–444. ACM, 2013.

[32] A. Perer and F. Wang. Frequence: interactive mining and visualization of temporal frequent event sequences. In *Proceedings of the 19th international conference on Intelligent User Interfaces*, pp. 153–162. ACM, 2014.

[33] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: visualizing personal histories. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 221–227. ACM, 1996.

[34] C. Plaisant and B. Shneiderman. The diversity of data and tasks in event analytics. In *Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis*, 2016.

[35] P. J. Polack, S.-T. Chen, M. Kahng, M. Sharmin, and D. H. Chau. Timestitch: Interactive multi-focus cohort discovery and comparison. In *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*, pp. 209–210. IEEE, 2015.

[36] A. C. Robinson, D. J. Peuquet, S. Pezanowski, F. A. Hardisty, and B. Swedberg. Design and evaluation of a geovisual analytics system for uncovering patterns in spatio-temporal event data. *Cartography and Geographic Information Science*, 44(3):216–228, 2017.

[37] R. A. Ruddle, J. Bernard, T. May, H. Lücke-Tieke, and J. Kohlhammer. Methods and a research agenda for the evaluation of event sequence visualization techniques. In *Proceedings of the IEEE VIS 2016 Workshop on Temporal & Sequential Event Analysis*. Leeds, 2016.

[38] D. Salomon and G. Motta. *Handbook of data compression*. Springer Science & Business Media, 2010.

[39] J. M. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International Conference on Artificial Neural Networks*, pp. 175–184. Springer, 2009.

[40] Z. Shen and N. Sundaresan. Trail explorer: Understanding user experience in webpage flows. *IEEE VisWeek Discovery Exhibition*, pp. 7–8, 2010.

[41] Z. Shen, J. Wei, N. Sundaresan, and K.-L. Ma. Visual analysis of massive web session data. In *Large Data Analysis and Visualization (LDAV), 2012 IEEE Symposium on*, pp. 65–72. IEEE, 2012.

[42] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pp. 336–343. IEEE, 1996.

[43] B. Shneiderman and C. Plaisant. Sharpening analytic focus to cope with big data volume and variety. *IEEE computer graphics and applications*, 35(3):10–14, 2015.

[44] J. Stasko and E. Zhang. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *Information Visualization, 2000. InfoVis 2000. IEEE Symposium on*, pp. 57–65. IEEE, 2000.

[45] R. Veras and C. Collins. Optimizing hierarchical visualizations with the minimum description length principle. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):631–640, 2017.

[46] R. Veras and C. Collins. Optimizing hierarchical visualizations with the minimum description length principle. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):631–640, Jan 2017.

[47] K. Vrotsou, J. Johansson, and M. Cooper. Activitree: interactive visual exploration of sequences in event-based data using graph similarity. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):945–952,

2009.

[48] G. Wang, X. Zhang, S. Tang, H. Zheng, and B. Y. Zhao. Unsupervised clickstream clustering for user behavior analysis. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pp. 225–236. ACM, 2016.

[49] J. Wang, J. Han, and C. Li. Frequent closed sequence mining without candidate maintenance. *IEEE Transactions on Knowledge and Data Engineering*, 19(8), 2007.

[50] T. D. Wang, C. Plaisant, A. J. Quinn, R. Stanchak, S. Murphy, and B. Shneiderman. Aligning temporal data by sentinel events: discovering patterns in electronic health records. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 457–466. ACM, 2008.

[51] T. D. Wang, C. Plaisant, B. Shneiderman, N. Spring, D. Roseman, G. Marchand, V. Mukherjee, and M. Smith. Temporal summaries: Supporting temporal categorical searching, aggregation and comparison. *IEEE transactions on visualization and computer graphics*, 15(6), 2009.

[52] F. Wanner, A. Stoffel, D. Jäckle, B. C. Kwon, A. Weiler, D. A. Keim, K. E. Isaacs, A. Giménez, I. Jusufi, T. Gamblin, et al. State-of-the-art report of visual analysis for event detection in text data streams. In *Computer Graphics Forum*, vol. 33, 2014.

[53] J. Wei, Z. Shen, N. Sundaresan, and K.-L. Ma. Visual cluster exploration of web clickstream data. In *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on*, pp. 3–12. IEEE, 2012.

[54] K. Wongsuphasawat and D. Gotz. Exploring flow, factors, and outcomes of temporal event sequences with the outflow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2659–2668, Dec 2012.

[55] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. Lifeflow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1747–1756. ACM, 2011.

[56] K. Wongsuphasawat and B. Shneiderman. Finding comparable temporal categorical records: A similarity measure with an interactive visualization. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pp. 27–34. IEEE, 2009.

[57] J. Wood. Visualizing personal progress in participatory sports cycling events. *IEEE Computer Graphics and Applications*, 35(4):73–81, 2015.

[58] E. Zgraggen, S. M. Drucker, D. Fisher, and R. Deline. (s|qu)eries: Visual regular expressions for querying and exploring event sequences. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, pp. 2683–2692, 2015.

[59] J. Zhao, C. Collins, F. Chevalier, and R. Balakrishnan. Interactive exploration of implicit and explicit relations in faceted datasets. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2080–2089, 2013.

[60] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 259–268. ACM, 2015.