10 APPENDIX

10.1 MinDL+LSH

Input: $\mathcal{S} = \{S_1, S_2, ..., S_n\}, th_{start}, th_{end}, th_{rate}$
Output: $\mathcal{C} = \{(P_1, G_1), (P_2, G_2), ..., (P_k, G_k)\}$

$\mathcal{C} = \{(S_1, \{S_1\}), (S_2, \{S_2\}), ..., (S_n, \{S_n\})\}$;
PriorityQueue $\mathcal{Q} = \emptyset$;
$th = th_{start}$;
while $th > th_{end}$ **do**
    /* LSH table initialization with threshold th */
    lshInit(th);
    for c_i *in* $\mathcal{C}$ **do**
        lshInsert(c_i)
    end
    /* MinDL: initialization phase */
    for $c_i \in \mathcal{C}$ **do**
        for $c_j \in lshQuery(c_i)$ **do**
            $\Delta L, c^* = Merge(c_i, c_j)$;
            if $\Delta L > 0$ **then**
                push $(\Delta L, c^*, c_i, c_j)$ to $\mathcal{Q}$;
            end
        end
    end
    /* MinDL: iterative merging phase */
    while $\mathcal{Q} \neq \emptyset$ **do**
        pop $(\Delta L, c^*, c_i, c_j)$ from $\mathcal{Q}$;
        remove c_i, c_j from $\mathcal{C}$, add c^* to $\mathcal{C}$;
        $c_{new} = c^*$;
        /* LSH table update */
        lshDelete(c_i), lshDelete(c_j);
        lshInsert(c^*);
        for $c \in lshQuery(c_{new})$ **do**
            $\Delta L, c^* = Merge(c, c_{new})$;
            if $\Delta L > 0$ **then**
                push $(\Delta L, c^*, c, c_{new})$ to $\mathcal{Q}$;
            end
        end
    end
    $th = th \times th_{rate}$;
end
return $\mathcal{C}$

Algorithm 3: MDL+LSH

In the algorithm, *lshInit* creates a hash table for the clusters in $\mathcal{C}$. Essentially the hash table is composed of a set of buckets. Each bucket contains a set of clusters and their representative patterns are hashed to the same value. The hash table needs to be recreated when th changes. It takes $O(n)$ ($n = \|\mathcal{C}\|$) to populate the hash table with the clusters in $\mathcal{C}$. n decreases over iterations as the clusters are merged together. *lshDelete* and *lshInsert* update the hash table when the clusters are merged. *lshQuery* takes a cluster and returns all the other clusters in the same bucket.

Using LSH reduces the need to compute ΔL for all pairs of clusters in $\mathcal{C}$ which takes $O(n^2)$ time, instead, only pairs within the same bucket will be considered. In this way, the method can reduce the running time significantly. This is validated by the empirical results in Fig. 3.

The three parameters th_{start}, th_{end} and th_{rate} need to be manually chosen to control the LSH threshold over iterations. In general, th_{start} should be close to 1 such that fewer candidate clusters need to be checked in earlier iterations while th_{end} should be close to 0 to make sure the description length is effectively minimized. th_{rate} should be a value between 0 and 1 to gradually decrease the threshold over time. We set $th_{start} = 0.9$, $th_{end} = 0.2$ and $th_{rate} = 0.6$ in the experiments.

10.2 Analytic tasks survey

Plaisant and Shneiderman [36] have recently summarized a set of high-level analytic tasks for event sequence data. We survey the existing visual analytic systems for event sequence data and list the tasks they support in Table. 1. Besides that, we also interview the experts to gather the requirements for the new application domain, i.e., vehicle data analytics. The analytical tasks proposed by Plaisant and Shneiderman are listed below. From Table. 1, we identify that ***T1***, ***T2***, ***T5*** and ***T7*** are the most commonly supported tasks across a wide range of visualization systems.

Heighten awareness

- ***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).
- ***T3***. Find and describe deviations from required or expected patterns.

Prepare or select data for further study

- ***T4***. Review data quality and inform choices to be made in order to model the data.
- ***T5***. Identify a set of records of interest.

Understanding impact of event patterns; plan action

- ***T6***. Compare two or more sets of records.
- ***T7***. Study antecedents or sequelae of an event of interest.
- ***T8***. Generate recommendations on actions to take.

Table 1. Summary of high-level tasks in previous design studies and in the new application domain, i.e., vehicle fault sequence analysis.

	T1	T2	T3	T4	T5	T6	T7	T8
Lifelines2 [52]	✓				✓		✓	
ActiviTree [49]		✓			✓	✓	✓	
DecisionFlow [13]		✓			✓			✓
Peekquence [22]	✓	✓					✓	
EventAction [8]	✓	✓		✓	✓	✓		✓
CoCo [30]	✓	✓			✓	✓		
Frequence [34]		✓			✓			
Monroe et al. [31]		✓			✓		✓	
Liu et al. [27]	✓	✓			✓		✓	
Vehicle fault sequence analysis	✓	✓			✓		✓	

10.3 Qualitative Comparison of Data Overview

To evaluate the effectiveness of our approach, we further conduct an experiment to compare the data overview (Fig. 10) generated by EventFlow [1], one of the state-of-the-art event sequence visualization techniques, and the data overview generated with the our approach.

In Fig. 10, the overview on the left shows the result of the VFS dataset. We can see that EventFlow can hardly group sequences and hence there are few patterns can be identified from the overview. Compared with the result of the proposed approach (Fig. 7), we can see that our method is better at discovering salient patterns in this noisy data, while EventFlow tends to generate fragmented results. On the other hand, by comparing the overview on the right with the overivew in Fig. 8, the result of EventFlow can also capture visually salient patterns. The result demonstrates that the proposed approach is particular suitable for the analysis of noisy data.

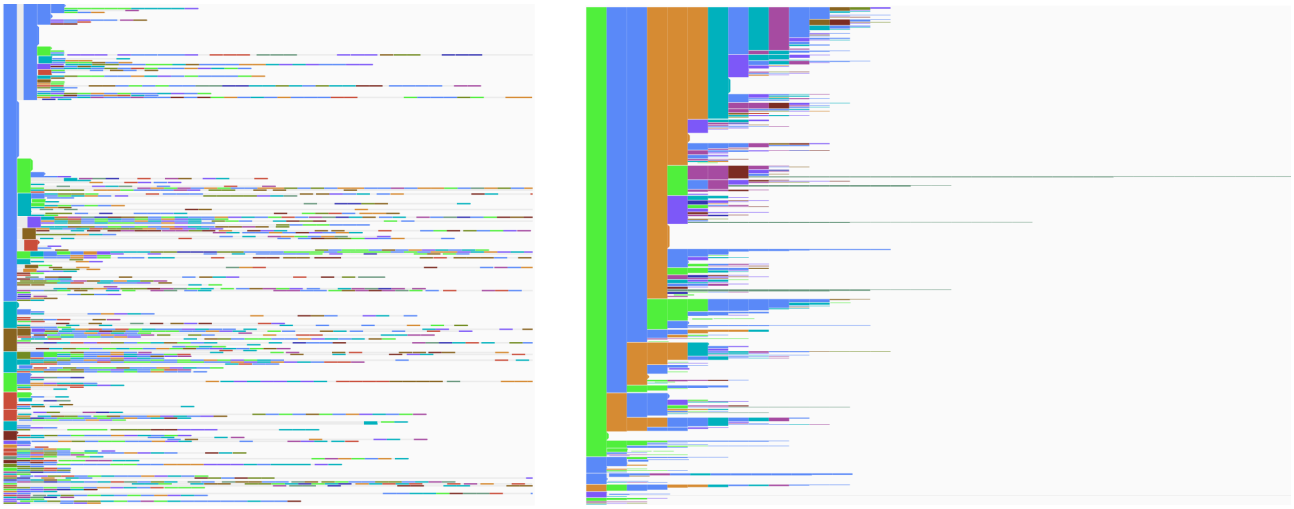

Fig. 10. The data overview generated by EventFlow [1]. Left: the result of the VFS dataset. Right: the result of the Agavue dataset.