An Ordered Chronicle Discovery Algorithm

Alexandre Sahuguède[1], Euriell Le Corronc[1], and Marie-Véronique Le Lann[1]

LAAS-CNRS, Université de Toulouse, CNRS, INSA, UPS, Toulouse, France
{alexandre.sahuguede, euriell.le.corronc, marie-veronique.lelann}@laas.fr

Abstract. Chronicles are temporal patterns well suited to capture dynamic process thanks to an event abstraction of the information of interest. Designing chronicles from a journal log is not a trivial task considering the huge amount of data generated by highly-advanced systems. Chronicle discovery is a mean to help expert design chronicles that are representative of a system behavior from direct observations. In this paper, a clustering approach to the chronicle discovery problem is considered. To improve the discovered chronicle quality, an order in the design of interesting pattern is introduced. This allows a better robustness to small perturbations in the input journal log. The efficiency of the ordered chronicle discovery algorithm is evaluated on a real dataset.

Keywords: Temporal data mining · Chronicle discovery · Clustering.

1 Introduction

In recent years, both academic and industrial communities have been interested in the study of the timed discrete event models called chronicles. Chronicles are temporal patterns well suited to capture dynamic process by means of an event abstraction of the information of interest. Chronicles are used to model care-pathways [3], web-services [7], or alarms [6]. However, designing chronicles is not a trivial task, a task that often requires the knowledge of an expert. The need for a chronicle discovery algorithm to help the expert to analyze the huge amount of data that modern systems generate is clear.

Several approaches to the chronicle discovery problem exist. Cram [2] proposes an interactive approach to this problem. Based on the *APriori* algorithm [1], this algorithm requires the user to check the result on each iteration and stops if the discovered chronicles satisfy the user. This algorithm uses a single temporal sequence for this task. Another approach [3] consists in mining several sequences where a known phenomenon appears once in each sequence. In previous work [8], we try to use a clustering approach to reduce the need of an expert. In this algorithm, a density-based clustering algorithm is used to cluster patterns by similarity. However, our first approach is not sufficiently robust to small perturbations in the input data. In this paper, we try to tackle this problem with the introduction of an order in the pattern construction.

The rest of this paper is organized as follows. In Section 2, concepts used in this paper are described. Section 3 presents the ordered chronicle discovery algorithm. In Section 4, our algorithm is evaluated on real data. Section 5 concludes this work.

2 Concepts and definitions

An *event* is defined by $x = (e, t)$ with an event type $e \in E$, and a time instant $t \in \mathbb{N}$. A *temporal sequence* is a time-ordered set of events denoted $\mathcal{S} = \{x_1, \ldots, x_n\}$ with n a finite number of events, and $t_j < t_{j+1}$, $j = 1, \ldots, n-1$. The set of all event types occurring in $\mathcal{S}$ is called $E_{\mathcal{S}}$.

A *chronicle* $\mathcal{C}$ is a pair $(\mathcal{X}, \mathcal{T})$ where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of partially ordered events with n a finite number of events, and $\mathcal{T} = \{\tau_{ij}\}_{1 \le i < j \le n}$ is a set of temporal constraints on $\mathcal{X}$. A *temporal constraint* is a tuple $\tau_{ij} = (x_i, x_j, t^-, t^+)$, also noted $\tau_{ij} = x_i[t^-, t^+]x_j$, with $0 \le t^- \le t^+$. $\tau_{ij} = x_i[t^-, t^+]x_j$ is said satisfied by a couple of events $x_i = (e_i, t_i)$, $x_j = (e_j, t_j)$ if and only if $(t_j - t_i) \in [t^-, t^+]$. $E_{\mathcal{C}}$ denotes the set of all event types of $\mathcal{C}$. A n-length chronicle is a chronicle with n events.

A *chronicle instance* $\mathcal{I}_{\mathcal{C}}(\mathcal{S})$ is a subset of events of $\mathcal{S}$ such that their event types are those of $\mathcal{X}$ and their time occurrences satisfy all the temporal constraints $\mathcal{T}$ of $\mathcal{C}$. The frequency of a chronicle $\mathcal{C}$ in a temporal sequence $\mathcal{S}$ is the number of instances of $\mathcal{C}$ in $\mathcal{S}$ and is named $f_{\mathcal{C}}(\mathcal{S})$.

Let $\mathcal{S}$ be a temporal sequence and let (a, b) be a pair of event types such that $a, b \in E_{\mathcal{S}}$. The set $\mathcal{O}_{ab}$ is the set of all the occurrences of (a, b) in $\mathcal{S}$ such that b follows a:

$$\mathcal{O}_{ab} = \{\langle (a, t_i), (b, t_j) \rangle \mid \forall i, j,\ t_i < t_j, (a, t_i), (b, t_j) \in \mathcal{S}\}. \tag{1}$$

The set $\mathcal{D}_{ab}$ is all the temporal distances between each occurrence of the pair (a, b):

$$\mathcal{D}_{ab} = \{(t_j - t_i) \mid \langle (a, t_i), (b, t_j) \rangle \in \mathcal{O}_{ab}\}. \tag{2}$$

Example 1. Let $\mathcal{C} = (\mathcal{X}, \mathcal{T})$ be a 3-length chronicle with $\mathcal{X} = \{x_1 = (a, t_1), x_2 = (b, t_2), x_3 = (b, t_3)\}$, and $\mathcal{T} = \{\tau_{12} = x_1[10, 20]x_2, \tau_{13} = x_1[3, 4]x_3\}$. The set of all event types of $\mathcal{C}$ is $E_{\mathcal{C}} = \{a, b\}$. Let $\mathcal{S} = \{x_1, x_2, x_3, x_4, x_5\}$ be a temporal sequence with $x_1 = (a, 1)$, $x_2 = (b, 4)$, $x_3 = (a, 10)$, $x_4 = (b, 13)$, and $x_5 = (b, 27)$. Events x_1 and x_3 are different occurrences of event type a. Two instances of $\mathcal{C}$ appear in $\mathcal{S}$: $\mathcal{I}^1_{\mathcal{C}}(\mathcal{S}) = \{x_1, x_2, x_4\} = \{(a, 1), (b, 4), (b, 13)\}$, and $\mathcal{I}^2_{\mathcal{C}}(\mathcal{S}) = \{x_3, x_4, x_5\} = \{(a, 10), (b, 13), (b, 27)\}$. The frequency of $\mathcal{C}$ in $\mathcal{S}$ is $f_{\mathcal{C}}(\mathcal{S}) = 2$ and corresponds to the total number of instances. About the pair (a, b): $\mathcal{O}_{ab} = \{\langle (a, 1), (b, 4) \rangle, \langle (a, 1), (b, 13) \rangle, \langle (a, 1), (b, 27) \rangle, \langle (a, 10), (b, 13) \rangle, \langle (a, 10), (b, 27) \rangle\}$ and $\mathcal{D}_{ab} = \{3, 12, 26, 3, 17\}$.

Proposition 1. *Let $\mathcal{D}_{ab}$ be a set of temporal distances for a pair (a, b). The 2-length chronicle $\mathcal{C} = (\mathcal{X}, \mathcal{T})$ can be obtained from $\mathcal{D}_{ab}$ with $\mathcal{X} = \{x_1 = (a, t_1),\ x_2 = (b, t_2)\}$. $\mathcal{T} = \{\tau_{12} = x_1[\min\{\mathcal{D}_{ab}\}, \max\{\mathcal{D}_{ab}\}]x_j\}$ is given by the lower and upper bounds of $\mathcal{D}_{ab}$[1]. All instances $\mathcal{I}^i_{\mathcal{C}}(\mathcal{S})$ are occurrences of $\mathcal{O}_{ab}$. The frequency $f_{\mathcal{C}}(\mathcal{S})$ is the size of $\mathcal{D}_{ab}$.*

[1] When $\mathcal{D}_{ab}$ contains only one element, $t^- = t^+ = \mathcal{D}_{ab}$.

Proof. Directly from the chronicle definition and Equation (2).

Example 2. Let (a,b) be a pair of event types with $\mathcal{D}_{ab} = \{3, 12, 26, 3, 17\}$. The 2-length chronicle $\mathcal{C} = (\mathcal{X}, \mathcal{T})$ is obtained with $\mathcal{X} = \{x_1 = (a,t_1), x_2 = (b,t_2)\}$ and $\mathcal{T} = \{\tau_{12} = x_1[3,26]x_2\}$. The instances of $\mathcal{C}$ in the temporal sequence $\mathcal{S}$ seen in Example 1 are $\mathcal{I}^1_{\mathcal{C}}(\mathcal{S}) = \{(a,1),(b,4)\}$, $\mathcal{I}^2_{\mathcal{C}}(\mathcal{S}) = \{(a,1),(b,13)\}$, $\mathcal{I}^3_{\mathcal{C}}(\mathcal{S}) = \{(a,1),(b,27)\}$, $\mathcal{I}^4_{\mathcal{C}}(\mathcal{S}) = \{(a,10),(b,13)\}$, and $\mathcal{I}^5_{\mathcal{C}}(\mathcal{S}) = \{(a,10),(b,27)\}$. They correspond to the set $\mathcal{O}_{ab}$ of the pair (a,b).

3 Ordered chronicle discovery

In this section, the chronicle discovery algorithm proposed in our previous work [8] is explained. Then, our new approach is presented. With a temporal sequence as input, this is a 2 step algorithm.

The first step is the discovery of a set of 2-length chronicles from a temporal sequence $\mathcal{S}$. First, the set of temporal distances $\mathcal{D}_{ab}$ is computed for each pair of event types from $E_{\mathcal{S}}$. Then, a cluster analysis is performed on each of these sets with DBSCAN[2] [4]. This clustering algorithm builds clusters thanks to the point density in the data. Clusters found depend on some parameters ε and $minPts$. Points that are not included in clusters are determined as noise. This analysis takes advantage of the natural clusters present in the sets $\mathcal{D}$. Finally, 2-length chronicles are generated by means of Proposition 1 for each cluster. Since clusters found by DBSCAN are homogeneous, the frequencies $f_{\mathcal{C}}(\mathcal{S})$ of the 2-length chronicles generated from it is exactly the number of temporal distances in $\mathcal{D}$. Chronicles with a frequency less than $minPts$ are not generated by this step.

The second step of the chronicle discovery algorithm is the chronicle synthesis from the 2-length chronicles previously discovered. This step uses a similarity index on the events of the 2-length chronicles called the Jaccard index.

Definition 1 (Jaccard index). *Let $\mathcal{S}$ be a temporal sequence and $\mathcal{C} = (\mathcal{X}, \mathcal{T})$ be a chronicle with x_i one of its event. The set of time occurrences of x_i in all chronicles instances $\mathcal{I}_{\mathcal{C}}(\mathcal{S})$ is determined by the following formula:*

$$\mathcal{O}_i = \{t_i \mid \forall \mathcal{I}_{\mathcal{C}}(\mathcal{S}), x_i = (e, t_i) \in \mathcal{X}\}. \tag{3}$$

With x_j another event of $\mathcal{C}$ and $\mathcal{O}_j$ its set of time occurrences, the Jaccard index between x_i and x_j is calculated by the following formula:

$$S(x_i, x_j) = \frac{|\mathcal{O}_i \cap \mathcal{O}_j|}{|\mathcal{O}_i \cup \mathcal{O}_j|}. \tag{4}$$

The Jaccard index quantifies the frequency at which the occurrence of two events appears at an identical time. An index equals to 1 shows that the two events always appear at the same time, whereas an index lower than 1 shows that some occurrences do not appear at the same time.

[2] Any clustering algorithm that can be used on 1-dimensional data could be applied instead of DBSCAN.

The major difference between the approach in this paper and the approach in [8] is that we can use all values of the Jaccard index and not only values of 1. The use of the Jaccard index only on values of 1 allows most of the chronicle construction problems to be solved. A chronicle solution for each frequency is created. Then, each 2-length chronicle is combined to the chronicle solution of its frequency without any ordering constraints. The Jaccard index between the events of the 2-length chronicle and the events of the chronicle solution is then computed. If the Jaccard index is equal to 1, those events are merged. Once all the 2-length chronicles are treated, the learned chronicles correspond to the chronicles solution.

Unfortunately, this previous approach does not take into account all the problems that occur with the use of the Jaccard index on a percentage of similarity. Another method for this chronicle synthesis step should be considered.

First, 2-length chronicles are sorted by their frequency in the temporal sequence mined. Then, for each frequency in the set of all possible frequencies of the 2-length chronicles discovered $\mathcal{F}$, a chronicle solution will be synthesized. This chronicle will be the synthesis of the set of all candidate chronicles. A chronicle is candidate when its frequency is higher than the product between the current frequency and the $minJac$ parameter. The $minJac$ parameter is the minimum threshold for which the Jaccard index is taken into account. The candidate 2-length chronicles are then sorted by the value of the interval of their temporal constraints. Finally, for each pair of candidate chronicles, the Jaccard index between each event is computed.

Once all the Jaccard indexes are compiled, one chronicle solution is initialized with all the candidate 2-length chronicles. Then, operations are sorted according to the value of the Jaccard index: the higher the index, the sooner the operations will be applied to the chronicle solution. An operation is the fusion of two events in the chronicle solution. An operation with a Jaccard index below the threshold $minJac$ is not taken into account. Furthermore, since candidate chronicles are sorted by interval, the operations (with the same Jaccard index) between chronicles with small intervals will be treated sooner than those with high intervals. With the operations sorted, they are checked sequentially. If an operation satisfies the rules of a chronicle, it is applied on the chronicle solution. Once all operations are applied, the independent sub-chronicles in the chronicle solution are extracted and added to the set of learned chronicles. The use of the Jaccard index on different values than 1 means that several solutions can be found depending on the order of the operations. This step could be repeated with a different order of operations that could result in a different chronicle solution. In some cases, a different solution could be interesting for the users to explore. This is the object of future work.

An analysis of the algorithmic complexity of the presented algorithm shows that a polynomial complexity could be achieved. The overall algorithmic complexity of this algorithm is given by $O(n^4 m \log(m))$, with n the number of events in the input temporal sequence and m the length of the longest discovered chronicle. This complexity could be largely reduced by well-chosen parameters of the clustering step.

4 Experiments

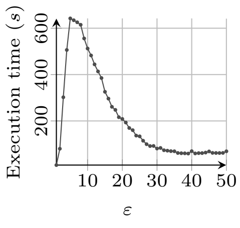
In this section, the efficiency of the ordered chronicle discovery algorithm is evaluated on real data. The dataset used in this experiment, called *Blocks*, comes from BIDE-D [5] and is summarized in Table 1. In this dataset, the temporal sequences describe visual primitives obtained from video of an hand stacking colored blocks. Events describe block contacts and actions of the hand. Each temporal sequence represents a specific scenario.

Table 1: The dataset Blocks with the number of events and event types for each temporal sequence.

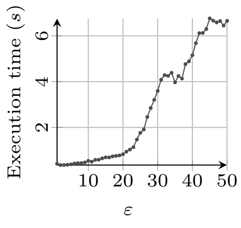
Temporal Sequence	Events	Event types
assemble	528	12
disassemble	486	12
move-left	150	10
move-right	154	10
pick-up	184	6
put-down	186	6
stack	362	10
unstack	364	10

Figure 1 illustrates the execution times for different values of the ε parameter with the parameter $minPts$ set to 3 and $minJac$ set to 0.9, whereas Figure 2 illustrates the number of learned chronicles. These plots compile results of 4 temporal sequences (*assemble*, *move-left*, *pick-up*, and *stack*) out of the 8 available. Most of these results show a peak in both the execution time and the number of chronicles learned with a value of ε around 10. This tends to indicate that the execution time is correlated to the number of chronicles learned. For the expert, a good value of ε could be the one that maximizes the number of chronicles learned. On the plot for the temporal sequence *move-left*, no peak appears, this could indicate that a good value of ε for this sequence is higher than 50. Table 2 shows the number of 2-length chronicles discovered, the number of chronicle learned, and execution time for each temporal sequences with the parameter ε set to 10, $minPts$ set to 3, and $minJac$ set to 0.9.

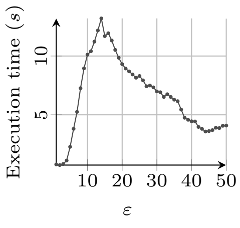
In order to evaluate the chronicle learned by the chronicle discovery algorithm presented in this paper, the notion of chronicle complexity is used. The chronicle complexity is defined by its length n and the number of its temporal constraints m. This performance metric is computed by $cc = \frac{2m}{n-1}$. The more constrained are the events of a chronicle, the bigger its complexity will be, whereas the less constrained the events of a chronicle, the smaller its complexity.



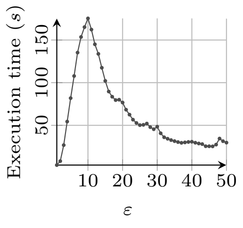
(a) assemble



(b) move-left

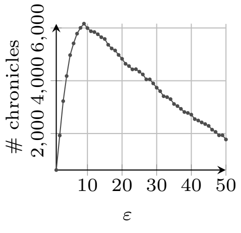


(c) pick-up

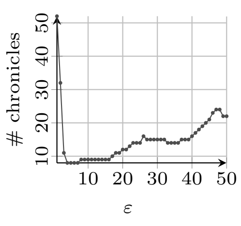


(d) stack

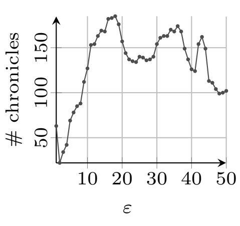
Fig. 1: Execution times for different values of ε with $minPts = 3$ and $minJac = 0.9$.



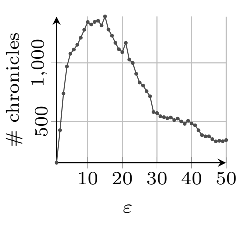
(a) assemble



(b) move-left



(c) pick-up



(d) stack

Fig. 2: Number of chronicles learned for different values of ε with $minPts = 3$ and $minJac = 0.9$.

Table 2: Number of 2-length chronicles discovered, number of chronicles learned, and execution time for each temporal sequence with $\varepsilon = 10$, $minPts = 3$, and $minJac = 0.9$.

Temporal Sequence	# 2-length chronicles discovered	# chronicles learned	Execution time (s)
assemble	47706	5995	512.451
disassemble	53171	3147	482.022
move-left	9568	9	0.54228
move-right	10039	10	0.408546
pick-up	10460	127	10.1122
put-down	10561	180	9.11633
stack	33774	1349	175.352
unstack	35666	1119	131.214

To determine the most relevant chronicles learned, we analyze the frequency w.r.t. complexity plot. An example of such a plot is given Figure 3 and represents the 180 chronicles learned from the sequence *put-down* with the parameters $\varepsilon = 10$ and $minJac = 0.9$. On this plot, few chronicles stand out: $\mathcal{C}_{30}$, $\mathcal{C}_{31}$, and $\mathcal{C}_{32}$ have a high frequency and a good complexity. $\mathcal{C}_{30}$ is graphically represented in Figure 4. This chronicle represents the behavior of the scenario *put-down* where an hand puts a red block on a green block.

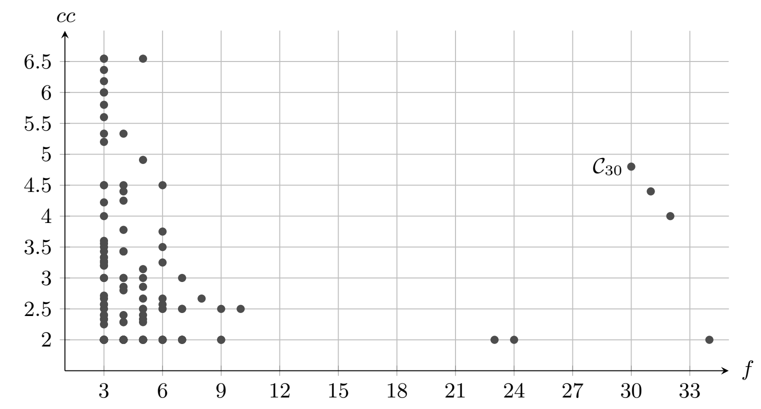


Fig. 3: The frequency w.r.t. complexity plot of the 180 chronicles learned in the sequence *put-down* with $\varepsilon = 10$, $minPts = 3$, and $minJac = 0.9$. The highlighted chronicle $\mathcal{C}_{30}$ is graphically represented in Figure 4.

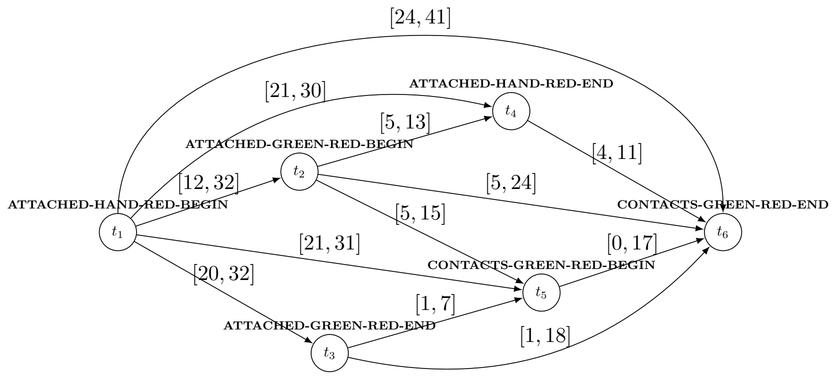


Fig. 4: Chronicle $\mathcal{C}_{30}$ of frequency 30 found in the sequence *put-down* with $\varepsilon = 10$, $minPts = 3$, and $minJac = 0.9$.

5 Conclusion

In this paper, an extension of the clustering approach to the chronicle discovery presented in [8] is provided. This extension introduces an order in the chronicle synthesis step that allows the Jaccard index to be exploited on a percentage of similarity and not only on 0 or 1. The use of the Jaccard index on a fuzzy value allows noisy data to be exploited. This chronicle discovery algorithm is evaluated on a real dataset. In future work, the algorithm introduced in this paper should be compared to others works, such as [2]. Furthermore, other datasets, synthetic as well as real, should be explored to confirm our results.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases. pp. 487–499 (1994)
2. Cram, D., Mathern, B., Mille, A.: A complete chronicle discovery approach: application to activity analysis. Expert Systems **29**(4), 321–346 (2012)
3. Dauxais, Y., Guyet, T., Gross-Amblard, D., Happe, A.: Discriminant chronicles mining - application to care pathways analytics. In: 16th Conference on Artificial Intelligence in Medicine, AIME 2017. pp. 234–244 (2017)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96). pp. 226–231 (1996)
5. Mörchen, F., Fradkin, D.: Robust mining of time intervals with semi-interval partial order patterns. In: Proceedings of the 2010 SIAM International Conference on Data Mining. pp. 315–326 (2010)
6. Morin, B., Debar, H.: Correlation of intrusion symptoms: An application of chronicles. In: Recent Advances in Intrusion Detection: 6th International Symposium, RAID 2003. pp. 94–112 (2003)
7. Pencolé, Y., Subias, A.: A chronicle-based diagnosability approach for discrete timed-event systems: Application to web-services. Journal of Universal Computer Science **15**(17), 3246–3272 (nov 2009)
8. Sahuguède, A., Le Corronc, E., Le Lann, M.V.: Chronicle discovery for diagnosis from raw data: A clustering approach. In: 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes, SAFEPROCESS 2018 (2018)