IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 21 October 2023, accepted 17 January 2024, date of publication 22 January 2024, date of current version 31 January 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3357139

RESEARCH ARTICLE

Efficient Frequent Chronicle Mining Algorithms: Application to Sleep Disorder

HARETH ZMEZM[1,2], JOSÉ MARÍA LUNA[1], EDUARDO ALMEDA[1], AND SEBASTIÁN VENTURA[1], (Senior Member, IEEE)
[1]Department of Computer Science and Numerical Analysis, Andalusian Research Institute in Data Science and Computational Intelligence (DaSCI), University of Córdoba, 14071 Córdoba, Spain
[2]Department of Biology, University of Kerbala, Karbala 56001, Iraq

Corresponding author: José María Luna (jmluna@uco.es)

This work was supported in part by the Spanish Ministry of Science and Innovation under Project PID2020-115832GB-I00, and in part by the European Fund of Regional Development.

ABSTRACT Sequential pattern mining is a dynamic and thriving research field that aims to extract recurring sequences of events from complex datasets. Traditionally, focusing solely on the order of events often falls short of providing precise insights. Consequently, incorporating the temporal intervals between events has emerged as a vital necessity across various domains, e.g. medicine. Analyzing temporal event sequences within patients' clinical histories, drug prescriptions, and monitoring alarms exemplifies this critical need. This paper presents innovative and efficient methodologies for mining frequent chronicles from temporal data. The mined graphs offer a significantly more expressive representation than mere event sequences, capturing intricate details of a series of events in a factual manner. The experimental stage includes a series of analyses of diverse databases with distinct characteristics. The proposed approaches were also applied to real-world data comprising information about subjects suffering from sleep disorders. Alluring frequent complete event graphs were obtained on patients who were under the effect of sleep medication.

INDEX TERMS Frequent event graphs, chronicle mining, sequence mining, temporal data mining, sleep disorder.

I. INTRODUCTION

Pattern mining techniques aim to discover hidden patterns in large databases [23], and they are useful not only for data understanding but also for decision-making processes [21]. Sequence analysis techniques [16] are required when the sequential order of the items or events is critical, e.g. text analyses where it is often relevant to consider the word order. These techniques, also known as sequential pattern mining [11], focus on extracting frequent event sequences from data. Although these techniques do not require a timestamp and primarily focus on the order of events, in many application domains, like network failure analysis [8], care pathways [7], and human activities analysis [5], the temporal distance between events is as critical as their order. Therefore, new and effective techniques that consider this aspect are required.

To fill this gap, Dousson and Duong [8] proposed the extraction of temporal patterns represented by temporal constraint networks. This challenging task, known as chronicle mining [24], discovers frequently occurring event graphs where the vertices are events, and the edges represent intervals denoting the time between the two linked events. A chronicle offers an organized account of pertinent or historical events in the sequence of their occurrence, facilitating users' comprehension of the timeline needed to navigate from one event to another. Consequently, temporal constraints establish the order of events relative to each other and the duration needed for transitions. This duration is represented as a range since two sequences involving identical events may occur at varying time intervals, as exemplified in the field of activity analysis. Two people (person A and person B) do the same events in a trial but in different time gaps.

The associate editor coordinating the review of this manuscript and approving it for publication was Vijay Mago.

The sequences of events for A and B are the same, but the resulting time gaps highly differ. It denotes a difference in the context, maybe because person A has practiced a lot, or perhaps because person B has disabilities or undiscovered problems to complete the events in a fast way.

Chronicle mining [13] has recently arisen as an alluring research agenda in different application domains such as medicine [2], [6] and predictive maintenance [24], mainly because graphs model the information in a highly expressive way [1]. Recent research works [24] have considered chronicle mining as a promising task for descriptive purposes, similar to pattern mining but including richer information in the form of graphs. This is a computationally hard problem as descriptive analytics requires an accurate description of the information (adjustment of the temporal constraints). The first approaches in the frequent chronicle mining field worked in two phases [24], mining frequent sequences using the CloSpan algorithm [31], and then adjusting the temporal constraints. However, a major problem of this methodology is two sequences may include the same events but they satisfy different transactions. It implies that the resulting graphs are the same (in terms of the events), but they include different temporal constraints. Let us imagine two frequent sequences: $< (A)(B)(C) >$ and $< (B)(C)(A) >$. They may satisfy different data transactions, and subsequently, the final event graphs (including the temporal constraints) may differ. At this point, one might wonder if returning two identical graphs in terms of events, but including different temporal constraints could confuse the end user. Additionally, existing approaches extract incomplete chronicles (absence of some links between pairs of events), and include inconsistencies (the chronicle does not properly adjust the temporal constraints to the satisfied sequences). All these problems mainly appear because frequent chronicle mining approaches were proposed for predictive purposes, where the information could be somehow inaccurate. The primary research question we aim to address is: Can we efficiently extract frequent event graphs that vary based on their events, maintain all event linkages, and contain no inconsistencies? To answer this question, this paper proposes new and efficient approaches for mining frequent event graphs from temporal data. These event graphs should be different, complete and without inconsistencies. The proposed approaches, namely, MEGMRTC (Mining Event Graphs through Minimum Ranges of the Temporal Constraints) and MEGBIA (Mining Event Graphs through a Bio-Inspired Approach) aim to improve the final performance by proposing new heuristic-based methods to find the right temporal constraints. Heuristic-based approaches have already proven to be effective in the field of pattern mining [27]. The experimental stage compares the proposals to CPM [24], which is the most recent and efficient algorithm for frequent chronicle mining. Our experiments compare these approaches based on the quality of chronicles produced, runtime, scalability, and memory requirements across 12 datasets. Last but not least, a case study was conducted to analyze and accurately understand frequent bio-physiological changes that occur during sleep on 100 subjects with sleep disorders [9]. Real-world data gathered by the Sleep Medicine Centre of the Hospital of Coimbra University [17] were considered to demonstrate the usefulness of frequent event graph mining.

The novelty of this research work is summarized as follows:

- The problem of chronicle mining for descriptive purposes is defined, paying special attention to completeness (all the events are linked), and data integrity (lack of inconsistencies in the final graph).
- We present two novel methodologies, MEGMRTC and MEGBIA, for mining frequent event graphs. In contrast to existing techniques, these methods do not take into account the extraction of frequent sequences. This avoids the extraction of event graphs with the same events, which hinders the understanding of the resulting set.
- Heuristic methods are proposed to address the computational hardness of adjusting temporal constraints. The advantages and disadvantages of these methods are analyzed on a large number of datasets. Performance is evaluated in terms of scalability, runtime, and memory requirements.
- The CPM [24] algorithm has been briefly modified to avoid sequences with the same events, so its comparison in terms of the quality of the solutions is fair.
- A case study is performed on real-world data gathered by the Sleep Medicine Centre of the Hospital of Coimbra University [17] to demonstrate the usefulness of using chronicles for descriptive purposes.

The remainder of this paper has the following structure. Section II introduces the background and related works of chronicle mining. Section III describes the chronicle mining problem with definitions and properties, as well as the problem statement. Section IV presents the proposed algorithms. Experimental results are reported in Section V, and a detailed case study is presented in Section VI. Finally, some conclusions and future works are in Section VII.

II. RELATED WORK

Since the early 90s, when the market basket problem was proposed to discover what items were bought together in a transaction, many studies have contributed to an improvement of the efficiency [21] and expressiveness [22] of the proposals. Sequence analysis techniques [11] are some examples of highly expressive proposals required when the sequential order of the items is critical. These are called un-temporal sequence pattern mining approaches [18] as only the item occurrence order is considered. In order to deal with time-related data, some approaches were proposed to consider not only the item occurrence order in a sequence pattern but also the time between successive items in a sequential pattern [32]. As a result, sequential pattern mining methods were classified into three categories [30]: 1) pattern discovery from point-based event sequences; 2) mining patterns from interval-based event sequences; 3) extraction of patterns from hybrid event sequences.

Yoshida et al. [32] defined the temporal sequence patterns of the form $A \xrightarrow{[0,7]} B \xrightarrow{[3,5]} C$. It means that the sequential pattern $A \rightarrow B \rightarrow C$ frequently appears in the data under analysis, and the transition times from A to B, and from B to C are [0, 7] and [3, 5], respectively. Chen et al. [4] presented a type of temporal sequence pattern by considering user-defined time intervals in advance. These predefined time intervals were considered to count the frequency of the temporal sequence patterns in the data. However, as they adopted some user-predefined constraints on the time intervals between successive items, it is difficult for a user to specify optimal constraints related to item interval [18]. Hu et al. [14] proposed the extraction of sequential patterns by modelling the temporal arrangements between pairs of events. Other research works proposed an enhancement of the temporal information such as Bellazzi et al. [3], which aimed at detecting trends in time series and providing descriptions using the temporal algebra: *overlaps*, *finished-by*, *before*, *starts*, etc. Additional researchers focused on time-constrained sequential pattern mining [19]. Times associated with the items (events) were used to restrict the mined sequences (some min-max time spans were predefined). Hence, in this research work, the temporal information is not provided as a descriptive feature, and the resulting knowledge is a row of events as in sequential pattern mining. Some other researchers focused on time-interval pattern mining [20], considering times associated with transactions (not events) to take valid data records. Recent studies [28] have been focused on a mix of both (time-constrained sequential pattern mining and time-interval pattern mining), not only considering the time feature to take valid transactions but also restricting the extracted sequences by considering min-max time spans.

All the previous sequential pattern mining techniques [11] were proposed to find frequent patterns in a database (multiple sequences), but it was frequent episode mining the one proposed to identify all episodes (subsequences of events) [15] that frequently appear in a single long sequence of events. Frequent episode mining works on two types of input sequences: simple (each event has a unique timestamp) and complex (simultaneous events are allowed). This is a computationally expensive task because the search space can be very large, and it is extremely difficult to find the proper frequency threshold. In this regard, the specialized literature includes algorithms for mining the top-k frequent episodes [12]. However, episode mining does not work on multiple sequences where it is required to obtain representative event graphs on all such sequences. Something similar occurs when working with process mining, where the aim is to collect a process log with data about the order of the events [26]. Here, the resulting model is represented through a net with starting/ending tasks, linked by transitions.

Dousson and Duong [8] innovatively applied temporal constraint networks to depict temporal patterns, enabling a user to comprehend the timeline between graph events. These event graphs encapsulate a concept similar to episode mining, wherein vertices represent events, and edges, time intervals between two linked events. The emerging task, known as chronicle mining [24], aims to generate illustrative event graphs from sequence sets, subtly escalating the complexity of frequent episode mining. STP formalism, widely advocated in diverse applications like medicine [2], failure prediction [24], and activity recognition [5], is the backbone of chronicle mining. Various algorithms have been proposed, focusing mainly on predictive [24] and discriminant [6] goals. Dauxais et al. [6] introduced the DCM algorithm to extract discriminant chronicle patterns across two classes or target variables (positive and negative). Sellami et al. [24] focused their research studies on machine failure prediction. They aimed to extract a chronicle related to a given failure with the aim of predicting a breakdown before it happens. Cram et al. [5] presented a heuristic chronicle discovery algorithm for activity recognition. For each pair of event types, a constraint database holds several temporal constraints in a structure called the constraint graph. Nevertheless, since the goal of such approaches was not to provide a useful description of what is happening in the data, the time constraints were not accurately adjusted and the authors did not pay attention to the completeness of the results.

III. PRELIMINARIES

A. FREQUENT CHRONICLE MINING

Frequent chronicle mining consists of extracting all temporal event graphs that appear more than a minimum predefined number of times in a temporal dataset. This section includes the basic definitions to understand the meaning of a chronicle (event graph), the quality measure and the task complexity.

Definition 1 (Event): Let $\mathbb{E}$ be a set of things that may occur (generally important or unusual) in a problem or activity. An event [8] is formally defined as $e \in \mathbb{E}$. Let $\mathbb{T}$ be a temporal domain where $\mathbb{T} \subseteq \mathbb{R}$. A temporal event is a tuple (e, t) such that $e \in \mathbb{E}$ and $t \in \mathbb{T}$. The tuple (e, t) means that the event e occurs at time t.

Example 1. Considering the sample sequential database shown in Table 1, it includes five different events: A, B, C, D and E. A sample temporal event is (A, 1), which appears in the three first transactions. The time associated with an event can be a timestamp represented in any format, e.g. *Tues 21-02-2017 21:35*. However, for a matter of simplicity, we denote it as an integer from now on.

Definition 2 (Sequence of events): Let us assume the set $\mathbb{E}$ of events is totally ordered by $\leq_{\mathbb{E}}$. Given a group of two or more events $\langle (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) \rangle$, and an index SID of such a group that may appear or not, a sequence of events is formally defined as the tuple $\langle SID, \langle (e_1, t_1), (e_2, t_2), \ldots, (e_n, t_n) \rangle \rangle$. Additionally, it is satisfied that, for all $i, j \in [1, n]$, $i < j \Rightarrow t_i \leq t_j$. If $t_i = t_j$ then $e_i <_{\mathbb{E}} e_j$.

TABLE 1. Example of a sequential database.

SID	Sequence
1	(A, 1), (B, 3), (A, 4), (E, 4), (C, 5), (C, 6), (D, 7)
2	(A, 1), (A, 3), (B, 5), (C, 8)
3	(A, 1), (B, 4), (C, 5), (B, 6), (C, 8), (D, 9)
4	(B, 2), (A, 4), (C, 5), (B, 6)
5	(A, 2), (B, 4), (C, 7)
6	(C, 4), (B, 5), (A, 6), (C, 7), (D, 10)

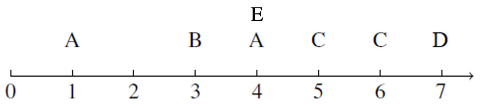


FIGURE 1. Sequence with SID 1 from the sequential database shown in Table 1.

Example 2. The sample sequential database shown in Table 1 includes 6 different sequences of events. A sample sequence is $\langle 1, \langle (A, 1), (B, 3), (A, 4), (E, 4), (C, 5), (C, 6), (D, 7)\rangle\rangle$, which includes all the feasible events (A, B, C, D and E) in data. The meaning of this sequence is the event A occurs at time 1, B at time 3, A and E at time 4, C at time 5, C at time 6, and D at time 7. This same information provided by the first sequence in Table 1 can also be expressed graphically by a timeline as it is illustrated in Figure 1.

Definition 3 (Subsequence of a sequence of events): A subsequence of a given sequence is a sequence that can be derived from the given sequence by deleting some or no elements without changing the order of the remaining elements. Given two groups of two or more events $G_x = \langle (e_i, t_i), \ldots, (e_j, t_j)\rangle$ and $G_y = \langle (e_k, t_k), \ldots, (e_z, t_z)\rangle$, G_y is a subsequence of G_x if and only if $G_y \subseteq G_x$. A subsequence is, at the same time, a sequence of events

Example 3. Taking the sample sequential database shown in Table 1, the sequence $\langle (A, 1), (A, 4), (C, 6), (D, 7)\rangle$ is a subsequence of the first sequence in Table 1, that is, $\langle 1, \langle (A, 1), (B, 3), (A, 4), (E, 4), (C, 5), (C, 6), (D, 7)\rangle\rangle$.

Definition 4 (Temporal constraint): A temporal constraint is defined as a 4-tuple (e_1, e_2, t^-, t^+) where $e_1, e_2 \in \mathbb{E}$, $e_1 \leq_{\mathbb{E}} e2$ and $t^-, t^+ \in \mathbb{T}$, $t^- \leq t^+$. A temporal constraint can also be presented as $e_1[t^-, t^+]e_2$, and it is satisfied by a couple of events $((e, t), (e', t'))$, $e \leq_{\mathbb{E}} e'$, $t \leq t'$ if and only if $e = e_1$, $e' = e_2$ and $t' - t \in [t^-, t^+]$.

Example 4. Considering sample sequential database illustrated in Table 1, and considering the sample temporal constraint $tc = (A, B, 3, 5)$ (also denoted as $A[3, 5]B$), it is satisfied by transactions with SID 2 and 3. Focusing on $\langle 2, \langle (A, 1), (A, 3), (B, 5), (C, 8)\rangle\rangle$, the couple of events $((A, 1), (B, 5))$ satisfy tc since $5 - 1 \in [3, 5]$. Additionally, tc is satisfied by the couple of events $((A, 1), (B, 4)) \in SID = 3$ since $4 - 1 \in [3, 5]$, and also $((A, 1), (B, 6)) \in SID = 3$ since $6 - 1 \in [3, 5]$. On the contrary, this temporal constraint $tc = (A, B, 3, 5)$ is not satisfied by the couple of events $((A, 3), (B, 5)) \in SID = 2$, since $5 - 3 \notin [3, 5]$. It should be noted that the temporal constraint denotes the path from A to B, thus is why the couple of events $((B, 3), (A, 4)) \in SID = 1$ produces a temporal distance of -1.

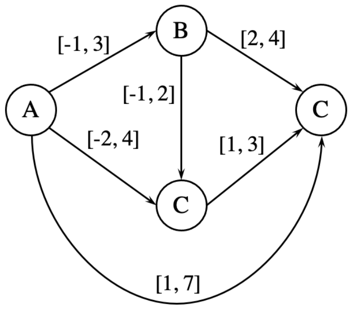


FIGURE 2. Sample chronicle obtained from the sequential database shown in Table 1.

Example 5. A temporal constraint may also be defined in timestamp format. Given the sample temporal constraint $tc = (X, Y,$ *Tues 21-02-2017 21:35*, *Tues 21-02-2017 23:50*), it is satisfied by the couple of events $((X, Tues21-02-201722:00), (Y, Tues21-02-201723:35))$. Additionally, it is possible to consider more general timestamps, e.g. $tc = (X, Y, 21{:}35, 23{:}50)$, which does not consider the date but just the time.

Definition 5 (Chronicle): A chronicle $\mathcal{C}$ or event graph is defined [8] as a tuple $(\mathcal{E}, \mathcal{T})$, where $\mathcal{E} = \{e_1, \ldots, e_n\}$ satisfying that $\forall i, e_i \in \mathbb{E}$ and $e_i \leq_{\mathbb{E}} e_{i+1}$; whereas $\mathcal{T} = \{t_{ij}\}_{1\leq i<j\leq|\mathcal{E}|}$ is a set of temporal constraints on $\mathcal{E}$ such that for all pairs (i, j) satisfying $i < j$, t_{ij} is denoted by $e_i[t_{ij}^-, t_{ij}^+]e_j$. It is important to highlight that since the constraint $e_i[t_{ij}^-, t_{ij}^+]e_j$ is equivalent to $e_j[-t_{ij}^+, -t_{ij}^-]e_i$ it is usually considered the order on events, $\leq_{\mathbb{E}}$, to decide which one is represented in the chronicle. The set $\mathcal{E}$ might contain several occurrences of a same event type.

Example 6. Figure 2 shows a sample chronicle obtained from the sample database shown in Table 1. This chronicle $\mathcal{C}$ is defined by $\mathcal{E} = \{e_1 = A, e_2 = B, e_3 = C, e_4 = C\}$ and $\mathcal{T} = \{e_1[-1, 3]e_2, e_1[-2, 4]e_3, e_1[1, 7]e_4, e_2[-1, 2]e_3, e_2[2, 4]e_4, e_3[1, 3]e_4\}$. Analysing this chronicle, the edge between A and B denotes the temporal constraint $(A, B, -1, 3)$ or $A[-1, 3]B$, representing that the event A occurs between a temporal unit (hours, days, weeks, etc.) after B and 3 temporal units before B. The negative value denotes that B occur before A.

Definition 6 (Incomplete chronicle): A chronicle $\mathcal{C}$ or event graph is defined as an incomplete chronicle if there is at least one temporal constraint with infinity values, i.e. $t_{ij} = \infty$. Formally, a chronicle $\mathcal{C} = (\mathcal{E}, \mathcal{T})$ is an incomplete chronicle if and only if $\exists t_{ij} \in \mathcal{T}$ where $t^- = -\infty, t^+ = \infty$. In other words, there is at least one temporal constraint with no restriction, which is represented as $e_i(-\infty, \infty)e_j$. This type of chronicle is less restrictive than that defined in Definition 5.

Example 7. Figure 3 shows a sample incomplete chronicle obtained from the sample database shown in Table 1. This chronicle $\mathcal{C}$ is defined by $\mathcal{E} = \{e_1 = A, e_2 = B, e_3 = C, e_4 = C\}$ and $\mathcal{T} = \{e_1[-1, 3]e_2, e_1[-2, 4]e_3, e_1(-\infty, \infty)e_4, e_2[-1, 2]e_3, e_2(-\infty, \infty)e_4, e_3[1, 3]e_4\}$. This

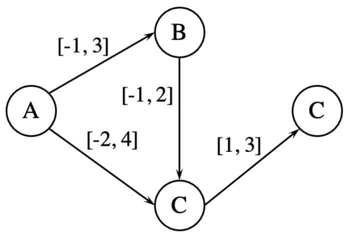


FIGURE 3. Sample incomplete chronicle obtained from the sequential database shown in Table 1.

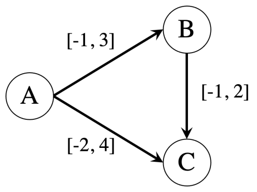


FIGURE 4. Sample sub-chronicle of the one shown in Figure 2.

chronicle includes some undefined temporal constraints, e.g. there is no edge between events B and C. It means that any value is feasible, and the constraint range is considered as $(-\infty, \infty)$.

Definition 7 (Sub-chronicle): A chronicle $\mathcal{C}_1 = (\mathcal{E}_1, \mathcal{T}_1)$ is defined as a subset of a chronicle $\mathcal{C}_2 = (\mathcal{E}_2, \mathcal{T}_2)$, denoted as $\mathcal{C}_1 \subset \mathcal{C}_2$, if and only if $\mathcal{E}_1 \subseteq \mathcal{E}_2$, and $\mathcal{T}_1 \subset \mathcal{T}_2$.

Example 8. Given the sample chronicle $\mathcal{C}_1$ (see Figure 2) defined by $\mathcal{E}_1 = \{e_1 = A, e_2 = B, e_3 = C, e_4 = C\}$ and $\mathcal{T}_1 = \{e_1[-1, 3]e_2, e_1[-2, 4]e_3, e_1[1, 7]e_4, e_2[-1, 2]e_3, e_2[2, 4]e_4, e_3[1, 3]e_4\}$, and the sample chronicle $\mathcal{C}_2$ (see Figure 4) defined by $\mathcal{E}_2 = \{e_1 = A, e_2 = B, e_3 = C\}$ and $\mathcal{T}_2 = \{e_1[-1, 3]e_2, e_1[-2, 4]e_3, e_2[-1, 2]e_3\}$, it is possible to assert that $\mathcal{C}_2 \subseteq \mathcal{C}_1$. In this example, $\mathcal{E}_2 \subset \mathcal{E}_1$, and $\mathcal{T}_1 \subset \mathcal{T}_2$.

Definition 8 (Super-chronicle): A chronicle $\mathcal{C}_1 = (\mathcal{E}_1, \mathcal{T}_1)$ is defined as a superset of chronicle $\mathcal{C}_2 = (\mathcal{E}_2, \mathcal{T}_2)$, denoted as $\mathcal{C}_1 \supset \mathcal{C}_2$, if and only if $\mathcal{C}_2$ is a sub-chronicle of $\mathcal{C}_1$ (see Definition 7).

Definition 9 (Restrictive chronicle): A chronicle $\mathcal{C}_1 = (\mathcal{E}_1, \mathcal{T}_1)$ is denoted as more restrictive than other chronicle $\mathcal{C}_2 = (\mathcal{E}_2, \mathcal{T}_2)$ if and only if $\mathcal{E}_1 \equiv \mathcal{E}_2$, and at least one temporal constraint in $\mathcal{T}_1$ has a smaller range than $\mathcal{T}_2$ for the same pair of events.

Example 9. Given the sample chronicle $\mathcal{C}_1$ defined by $\mathcal{E}_1 = \{e_1 = A, e_2 = B, e_3 = C\}$ and $\mathcal{T}_1 = \{e_1[-1, 3]e_2, e_1[-2, 4]e_3, e_2[-1, 2]e_3\}$, and the sample chronicle $\mathcal{C}_2$ defined by $\mathcal{E}_2 = \{e_1 = A, e_2 = B, e_3 = C\}$ and $\mathcal{T}_2 = \{e_1[0, 3]e_2, e_1[1, 3]e_3, e_2[-1, 2]e_3\}$, it is possible to assert that $\mathcal{C}_2$ is more restrictive than $\mathcal{C}_1$. In this example, $\mathcal{E}_1 \equiv \mathcal{E}_2$, but some of the temporal restrictions in $\mathcal{T}_2$ are more restrictive than those in $\mathcal{T}_1$.

Definition 10 (Quality measure): The quality of a chronicle $\mathcal{C}$ or event graph is defined by its frequency [24], also known as the number of sequences the chronicle satisfies from the database (all the temporal constraints defined by $\mathcal{C}$ are satisfied). Given a database defined as a set of sequences $\mathcal{S}$, the frequency of a chronicle $\mathcal{C}$ in $\mathcal{S}$ is mathematically denoted as $|\{s \in \mathcal{S} : \mathcal{C} \subseteq s\}|$. This frequency can be defined in relative terms as $|\{s \in \mathcal{S} : \mathcal{C} \subseteq s\}|/|\mathcal{S}|$.

Example 10. Given the sample chronicle $\mathcal{C}$ defined by $\mathcal{E} = \{e_1 = A, e_2 = B, e_3 = C, e_4 = C\}$ and $\mathcal{T} = \{e_1[-1, 3]e_2, e_1[-2, 4]e_3, e_1[1, 7]e_4, e_2[-1, 2]e_3, e_2[2, 4]e_4, e_3[1, 3]e_4\}$, its frequency on the sample database shown in Table 1 is 3 (SIDs 1, 3 and 6). In relative terms, this frequency is 3/6=50%.

Definition 11 (Chronicle mining): The task of mining frequent chronicles aims to extract any chronicle $\mathcal{C}$ (see Definition 5) with a frequency (see Definition 10) higher than a minimum predefined value.

Additionally, it is important to highlight that the chronicle mining task presents a series of properties that should be considered.

Property 1. A chronicle $\mathcal{C}$ may satisfy a sequence more than once. As chronicles include a set of events that are subsequences of sequences of events, a subsequence may appear more than once in a sequence. Similarly, the temporal constraints of a chronicle may be satisfied more than once in a sequence.

Example 11. Given the sample chronicle $\mathcal{C}$ defined by $\mathcal{E} = \{e_1 = A, e_2 = B, e_3 = C\}$ and $\mathcal{T} = \{e_1[-1, 3]e_2, e_1[-2, 4]e_3, e_2[-1, 2]e_3\}$, it satisfies the sequence $\langle 1, \langle (A, 1), (B, 3), (A, 4), (E, 4), (C, 5), (C, 6), (D, 7)\rangle\rangle$ shown in Table 1 twice. More specifically, the chronicle $\mathcal{C}$ is satisfied by the subsequences: $\langle (A, 1), (B, 3), (C, 5)\rangle$, and $\langle (B, 3), (A, 4), (C, 5)\rangle$. However, this property does not affect the final frequency (see Definition 10) as this quality measure calculates the number of sequences (transactions in a database) satisfied by the chronicle, not the number of subsequences.

Property 2. The set of sequences from data satisfied by a sub-chronicle is equal to or bigger than its super-chronicle. Similarly, the set of sequences satisfied by a super-chronicle is always equal to or smaller than its sub-chronicle. For the same reason, the frequency (see Definition 10) of a sub-chronicle is equal to or higher than its super-chronicle, whereas the frequency of a super-chronicle is always equal to or lower than its sub-chronicle.

Property 3. Given two chronicles $\mathcal{C}_1$ and $\mathcal{C}_2$, where $\mathcal{C}_1$ is more restrictive than $\mathcal{C}_2$ (see Definition 9), the set of sequences satisfied by $\mathcal{C}_1$ is always equal or smaller than $\mathcal{C}_2$. By the same reason, the frequency (see Definition 10) of $\mathcal{C}_1$ is equal or lower than $\mathcal{C}_2$.

Property 4. Given two chronicles $\mathcal{C}_1 = (\mathcal{E}_1, \mathcal{T}_1)$ and $\mathcal{C}_2 = (\mathcal{E}_2, \mathcal{T}_2)$, where $\mathcal{E}_1 \equiv \mathcal{E}_2$, and $\mathcal{C}_1$ being an incomplete chronicle, then the set of sequences from data satisfied by $\mathcal{C}_1$ is equal or bigger than $\mathcal{C}_2$. By the same reason, the frequency (see Definition 10) of $\mathcal{C}_1$ is equal or higher than $\mathcal{C}_2$.

Property 5. The sum of the temporal constraints of the linked events cannot be considered for unknown temporal constraints. In other words, given three events A, B and C, the time required to move from A to C is not always the time required to move from A to B plus the time required to move from B to C.

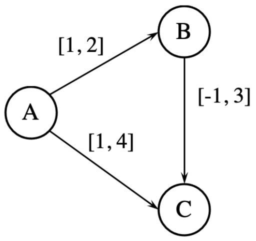


FIGURE 5. **Sample chronicle where the sum of the temporal constraints of the linked events is not the same as the direct link.**

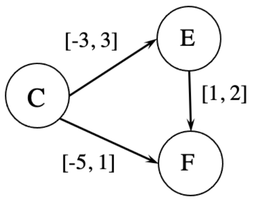


FIGURE 6. **Sample chronicle shown in [5].**

This last property (see Property 5) is of high interest and explains why sequential pattern mining with temporal constraints cannot produce the same information as chronicle mining does. Let us consider a sample chronicle shown in Figure 5. In this sample event graph, to go from event A to event B the user needs between 1 and 2 time units. Additionally, to go from B to C, they need between -1 and 3 time units. However, to go from A to C the user needs between 1 and 4 time units. What is happening? Why it is faster to go from A to C in the worst case (4 time units) than going through B (2+3=5 time units)? Is there any problem with the integrity? The answer is no, as the sum of the paths is not always equal to the direct path (see Property 5). Let us suppose a database containing the sequences $\langle 1, \langle (A, 1), (B, 2), (C, 5)\rangle\rangle$, $\langle 2, \langle (A, 1), (B, 3), (C, 5)\rangle\rangle$, and $\langle 3, \langle (A, 1), (C, 2), (B, 3)\rangle\rangle$. The chronicle illustrated in Figure 5 satisfies all these three sequences and the ranges are properly adjusted. For example, the sequence #1 denotes 1-time unit to move from A to B, 3-time units to move from B to C, and 4-time units to move from A to C. Sequence #2 denotes 2-time units to move from A to B, 2-time units to move from B to C, and 4-time units to move from A to C. Finally, the sequence #3 denotes 1-time unit to move from A to B, 1-time unit to move from C to B, and 1-time unit to move from A to C.

At this point, it is interesting to denote that temporal sequence pattern mining does not satisfy Property 5 as it provides sequences [32] of the form A $\xrightarrow{[0,7]}$ B $\xrightarrow{[3,5]}$ C. The temporal constraint among A and C is unknown, and cannot be inferred from A $\xrightarrow{[0,7]}$ B, and B $\xrightarrow{[3,5]}$ C. Additionally, since this unknown constraint cannot be inferred, the information provided by an event graph is more powerful. This problem is larger when the number of events increases as temporal sequence patterns provide just two links by event (except for the first and the last one). Some algorithms for mining temporal sequences are explained in the related work section.

Property 6. A chronicle well-defined should satisfy the data integrity, and avoid inconsistencies. It means that the chronicle should adjust the temporal constraints to the satisfied sequences.

Let us consider a sample chronicle shown in Figure 6, which was obtained and analyzed in [5]. In this sample event graph, there is no set of sequences that properly adjust the temporal constraints, so this chronicle does not hold the integrity requirement (see Property 6). Here, the event E is always before F (between 1 and 2 time units). It is therefore impossible that F occurs 5 time units before C, but E in the range [-3, 3]. It is important to remark that due to the aim of mining frequent event graphs is to explain what is going on in data, it is crucial to maintain the correctness of the resulting insights. This was not properly addressed by existing proposals in chronicle mining, mainly due to they were not designed for descriptive purposes. It is therefore easy to find chronicles with some inconsistencies in the temporal constraints (as the one previously presented), describing different behaviour depending on the path to follow.

Last, but not least, it is important to remark that the time complexity of this task depends on the number of chronicles in the search space and the cost of the operations for adjusting the temporal constraints. Let t be the number of sequences in a database, n the maximum length of the sequences, and m the number of chronicles, the complexity is equal to $(t \times n \times m)$.

B. PROBLEM STATEMENT

Chronicle mining has recently emerged as a promising research area for descriptive purposes [24], providing a richer graphical representation of information compared to conventional pattern mining. This depth of detail cannot be achieved by any sequential pattern mining technique, even those that mine temporal sequence patterns. Whether they consider user-defined time intervals in advance or not, these techniques derive transition times between pairs of events, with each event serving as the start or end point for only one transition. Consequently, in a sample sequence like $A \rightarrow B$ within a time range [0,7] and $B \rightarrow C$ within a time range [3,5], information about the transition between A and C is unavailable, thus violating Property 5. Such a sequence is viewed as an incomplete chronicle (as per Definition 6). Therefore, methods for mining temporal sequence patterns can also be used to mine incomplete chronicles.

The data extracted from complete chronicles is more comprehensive than that of incomplete chronicles, which is vital for descriptive tasks. So far, specific algorithms have been proposed that extract frequent chronicles in two phases [24]: 1) mining frequent sequences with the CloSpan algorithm [31]; and 2) adjusting the temporal constraints. However, a significant challenge is that the extraction of frequent sequences can result in different sequences with identical events due to variations in order. For instance,

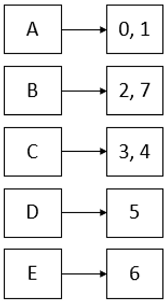


FIGURE 7. Event-index data representation.

$< (A)(B)(C) >$ and $< (B)(C)(A) >$. In sequential pattern mining, these sequences could correspond to different data transactions, leading to distinct final chronicles. Furthermore, existing methods often overlook Property 6.

To illustrate, let's revisit the sample chronicle displayed in Figure 6, obtained from [5]. Avoiding such inconsistencies is crucial for accurately describing data behaviour, a key goal of descriptive tasks. Given the aforementioned considerations, this research aims to develop efficient algorithms to mine complete chronicles, free of inconsistencies and containing distinct events. As this is a formidable task, especially in adjusting the temporal constraints, the proposed algorithms in this study should be efficient in terms of runtime and memory consumption.

IV. PROPOSED APPROACHES

Two different approaches are proposed (MEGMRTC and MEGBIA) and described in this section. The data representation for both approaches is the same, which is also explained in this section. MEGMRTC is an exhaustive search approach that extracts all the existing chronicles above a predefined support threshold (quality measure, see Definition 10). MEGBIA is an evolutionary search approach that extracts chronicles without needing any frequency threshold, which is a daunting process when the user has no knowledge about the data distribution or the application field.

A. DATA REPRESENTATION

A significant feature of the proposed approaches is that they do not engage with sequential patterns in the first phase. This strategy allows us to prevent extracting different sequential patterns that represent the same event graph, for instance, $< (A)(B)(C) >$ and $< (B)(C)(A) >$. Furthermore, if the same event appears more than once in the pattern, it should be considered as two independent events in the event graphs. In this respect, we convert the events into indices and extract sets of indices without considering the order. Consequently, the set $\{1, 2, 3\}$ is equivalent to $\{2, 3, 1\}$, considering A is denoted as 1, B as 2, and C as 3.

The original database is therefore transformed to distinguish the occurrences of the same event type in a sequence. An event e that appears n_e times on a sequence, being n_e the highest number of occurrences of e in any sequence from the whole database, is encoded by n_e items: $I_e^1 \ldots I_e^{n_e}$. The order of the I_e^i remains the same. To simplify operations, the $\leq_{\mathbb{E}}$ order of the chronicle is replaced by the $\leq_{\mathbb{I}}$ order, where $\forall e \in \mathbb{E}, \forall i \in [1, n_e]$ then $I_e^i \in \mathbb{I}$. Taking the sequence $\langle 1, \langle (A, 1), (B, 3), (A, 4), (E, 4), (C, 5), (C, 6), (D, 7) \rangle \rangle$ as an example from Table 1, the following ordered events are extracted: A, A, B, C, C, D, E. These events are transformed into the following indices: 0, 1, 2, 3, 4, 5, 6. The transformation information is maintained through a hashing function that determines the associations between events and indices (see Figure 7). For a given key k based on the event, the function maps k to the corresponding bucket that includes information on associated indices. Simultaneously, the input sequential database (see Table 1) is modified according to these event-index associations, resulting in a new database (see Table 2). Since each event is associated with one or more indices (see Figure 7), the indices in the transformed database are obtained in ascending order from the hashing function. For instance, take the sequence $\langle 5, \langle (A, 2), (B, 4), (C, 7) \rangle \rangle$ from Table 1. Here, event A is transformed into index 0, B into index 2, and C into index 3. These indices are the first for each event in Figure 7. In cases where an item appears more than once, the indices are taken in ascending order from its bucket. Thus, the sequence $\langle 6, \langle (C, 4), (B, 5), (A, 6), (C, 7), (D, 10) \rangle \rangle$ is encoded as $\langle 6, \langle (3, 4), (2, 5), (0, 6), (4, 7), (5, 10) \rangle \rangle$. The first event C is encoded as 3, whereas the second event C is encoded as 4 (see Figure 7). Finally, the transformed database is stored in memory as a list of hashing functions. Given a key k based on the index of an event in the data, it maps k to the corresponding bucket including information on the temporal constraint. This allows for faster access to the sequence items, thereby accelerating the evaluation process and the generation of temporal constraints. It is important to note that the time associated with an event can be represented as a timestamp in any format, for example, *Tues 21-02-2017 21:35*. However, for the sake of simplicity in this article, we denote it as an integer.

TABLE 2. Transformed database from the input database shown in Table 1.

SID	Sequence
1	(0, 1), (2, 3), (1, 4), (6, 4), (3, 5), (4, 6), (5, 7)
2	(0, 1), (1, 3), (2, 5), (3, 8)
3	(0, 1), (2, 4), (3, 5), (7, 6), (4, 8), (5, 9)
4	(2, 2), (0, 4), (3, 5), (7, 6)
5	(0, 2), (2, 4), (3, 7)
6	(3, 4), (2, 5), (0, 6), (4, 7), (5, 10)

B. MEGMRTC

This first approach (see Algorithm 1), known as MEGMRTC, extracts all frequent chronicles (considering a minimum frequency threshold) that exist in the data under analysis. This algorithm works as a two-step procedure: *a*) extracting the frequent patterns or itemsets; *b*) adding temporal constraints

Algorithm 1 MEGMRTC Algorithm

```
Require: $\Omega$, $sup\_threshold$, $minLen$, $max$, $min$   ▷ Original dataset $\Omega$ (including timestamps), minimum support $sup\_threshold$, minimum number of events $minLen$, minimum $min$ and maximum $max$ temporal distance between events
Ensure: $\mathcal{C}$
 1: $\Omega \leftarrow$ remove timestamps from $\Omega$
 2: $\omega \leftarrow$ data transformation $\Omega$
 3: $\mathcal{P} \leftarrow$ methodology based on $LCM(\Omega, \omega, sup\_threshold, minLen)$   ▷ Frequent event sets
 4: $\mathcal{C} \leftarrow \emptyset$
 5: for all $p \in \mathcal{P}$ do
 6:     $supp \leftarrow$ support of $p$
 7:     $S \leftarrow$ subset of sequences from $\Omega$ satisfying $p$
 8:     $J \leftarrow \emptyset$   ▷ Set of subsequences satisfied
 9:     for all $s \in S$ do
10:         $I \leftarrow$ gets subsequences from $s$ satisfying $p$, $max$ and $min$
11:         if $I == \emptyset$ then
12:             $supp = supp - 1$
13:             if $supp < sup\_threshold$ then
14:                 breaks and continues with next pattern
15:             end if
16:         end if
17:         $I \leftarrow$ remove repeated subsequences in $I$
18:         $J \leftarrow J \cup I$
19:     end for
20:     $J \leftarrow$ remove repeated sets and supersets, and sort by length
21:     $J_1 \leftarrow J[1]$ and remove $J[1]$ from $J$
22:     $\mathcal{T}_{best} \leftarrow \mathcal{T}_{\infty}$
23:     if $J$ is empty then
24:         $\mathcal{T} \leftarrow$ update $\mathcal{T}_{best}$ using $J_1[1]$
25:         $\mathcal{C} \leftarrow \mathcal{C} \cup (p, \mathcal{T})$
26:         break and continue with next pattern
27:     end if
28:     for all $j \in J_1$ do
29:         $\mathcal{T} \leftarrow$ updates $\mathcal{T}_{\infty}$ using $j$
30:         $recursiveDraws(J, 1, \mathcal{T})$   ▷ See Algorithm 2
31:     end for
32:     $\mathcal{C} \leftarrow \mathcal{C} \cup (p, \mathcal{T}_{best})$
33: end for
34: return $\mathcal{C}$
```

to the mined patterns. The first procedure aims to extract frequent events from the encoded data (just indices of the events are considered) by following a similar methodology to the one of the LCM algorithm [25] in its version for mining frequent itemsets. LCM has been proven to be one of the fastest and most efficient algorithms for mining frequent itemsets [21], and this is why this methodology was taken as a baseline. The proposed methodology includes a procedure to form new candidate patterns that consider the order in the bucket of the event-index data representation described in the previous section. As a matter of clarification, the itemset $\{0, 5\}$ cannot be extended as $\{0, 5, 7\}$ because of the index order in the bucket B : $[2, 7]$ (see Figure 7). However, the same itemset could be extended as $\{0, 5, 2\}$ as it follows the index order. This proposed methodology is key to returning valid sets of events that can be used to form chronicles in the next procedure.

The second procedure (see lines 9 to 34, Algorithm 1) analyzes every frequent pattern saved in $\mathcal{P}$ (see line 3, Algorithm 1). The algorithm iterates to analyze every sequence and subsequence that satisfies the pattern (see lines 9 to 19, Algorithm 1), and to save not repeated subsequences which satisfy the minimum and maximum temporal distances between events given by the user (parameters *min* and *max*). At this point, it is important to remark that those sequences with no subsequence in the feasible range of distances are removed, so the final frequency is reduced, and the pattern p is discarded from the resulting set if the frequency threshold does not meet (see lines 13 to 15, Algorithm 1). As an example, let us consider the pattern $p = \{0, 2, 3\}$ which is present in all the transactions so its support or frequency is 6. Let us now consider the minimum and maximum distance values of 0 and 7, respectively. In the analysis of the subsequences (see lines 9 to 19, Algorithm 1), SID #1 is satisfied because there is at least one subsequence within these ranges, e.g. $\langle(0, 1), (2, 3), (3, 5)\rangle$. SID #2 is satisfied because there is at least one subsequence within these ranges, e.g. $\langle(0, 3), (2, 5), (3, 8)\rangle$. SID #3 is also satisfied because there is at least one subsequence within the valid ranges, e.g. $\langle(0, 1), (2, 4), (3, 5)\rangle$. SID #4 is not satisfied because there is no subsequence within the valid ranges. This transaction produces two invalid subsequences: $\langle(2, 2), (0, 4), (3, 5)\rangle$ (it requires -2 time units to move from 0 to 2, which is out of the range), and $\langle(0, 4), (3, 5), (2, 6)\rangle$ (it requires -1 time unit to move from 2 to 3, which is out of the range). SID #5 is satisfied because there is at least one subsequence within the valid ranges, e.g. $\langle(0, 2), (2, 4), (3, 7)\rangle$. Finally, SID #6 is not satisfied because there is no subsequence within the valid ranges. As a result, the frequency of this pattern is reduced, and it is 4 now (transactions with SIDs #1, #2, #3, and #5).

Every valid sequence (including at least one valid subsequence) is analyzed to reduce the final set of sequences (see line 20, Algorithm 1), removing those sequences that are the same or supersets of others in terms of the distances between events. In the example under analysis, SID #1 provides two valid subsequences (it should be noted that index 3 and 4 belong to the same bucket so they are the same): $\langle(0, 1), (2, 3), (3, 5)\rangle$ and $\langle(0, 1), (2, 3), (4, 6)\rangle$. It means that the time constraints are 2 (to move from 0 to 2), 2 (to move from 2 to 3), and 4 (to move from 0 to 3) for the first subsequence. As for the second subsequence, the time constraints are 2 (to move from 0 to 2), 3 (to move from 2 to 4), and 5 (to move from 0 to 4). Similarly, SID #2 provides two valid subsequences: $\langle(0,1),(2,5),(3,8)\rangle$ and $\langle(1,3),(2,5),(3,8)\rangle$. Additionally, SID #3 provides the following valid subsequences: $\langle(0,1),(2,4),(3,5)\rangle$ and $\langle(0,1),(2,4),(4,8)\rangle$. Analysing SID #5, it provides a valid subsequence: $\langle(0,2),(2,4),(3,7)\rangle$. Here, the time constraints are 2 (to move from 0 to 2), 3 (to move from 2 to 3), and 5 (to move from 0 to 3). As a result, SID #5 is a subset of SID #1 because of the time constraints obtained, that is, $\{[2,2,4],[2,3,5]\} \supset \{[2,3,5]\}$, and SID #1 is removed. The time constraints for SID #2 are $\{[4,3,7],[2,3,5]\} \supset \{[2,3,5]\}$, which denote a superset of SID #5. Again, SID #2 is removed. Finally, the algorithm sorts the resulting sequences based on the number of subsequences they include (ascending order). The most restrictive transactions/sequences are considered first. Following the same example, the two resulting sequences (SIDs #3 and #5) are sorted. SID #5 is placed in the first position as it includes just one subsequence $\langle(0,2),(2,4),(3,7)\rangle$. SID #3 is placed in second position as it includes three subsequenes: $\langle(0,1),(7,6),(4,8)\rangle$, $\langle(0,1),(2,4),(3,5)\rangle$, and $\langle(0,1),(2,4),(4,8)\rangle$.

Once the set of valid subsequences is reduced, a chronicle is formed by considering the pattern p as the set of events, and updating the time constraints which are first fixed to the universal constraint $(-\infty,\infty)$. These bounds are updated (reduced or extended) iteratively until all the sequences (and subsequences) are analyzed in order. The sum of ranges is analyzed, and the procedure expands as a tree graph through a recursive procedure (see line 30, Algorithm 1) when it finds two or more subsequences that, if selected to update the temporal constraints, the sum of ranges for each subsequence is the same and it is also minimum (there is a draw). Then, the algorithm calculates the path that minimizes the ranges of constraints (see Algorithm 2). Back to the example, the transaction SID #5 is first analyzed and the chronicle is initialized to the tuple $(\mathcal{E},\mathcal{T})$, where $\mathcal{E}=\{0,2,3\}$, and $\mathcal{T}=\{0[2,2]2,2[3,3]3,0[5,5]3\}$. Then, the recursive procedure updates this chronicle by analysing the three subsequences given by SID #3, and keeping the one that produces a smaller sum of ranges. Following with the example, $\langle(0,1),(7,6),(4,8)\rangle$ updates the time constraints to $\mathcal{T}=\{0[2,5]2,2[2,3]3,0[5,7]3\}$; $\langle(0,1),(2,4),(3,5)\rangle$ updates the time constraints to $\mathcal{T}=\{0[2,3]2,2[1,3]3,0[4,5]3\}$; $\langle(0,1),(2,4),(4,8)\rangle$ updates the time constraints to $\mathcal{T}=\{0[2,3]2,2[3,4]3,0[5,7]3\}$. The sum of ranges for the first one is 3+1+2=6. The second one is 1+2+1=4. The third one is 1+1+2=4. Thus, two different time constraints optimize the ranges: $\mathcal{T}=\{0[2,3]2,2[1,3]3,0[4,5]3\}$ and $\mathcal{T}=\{0[2,3]2,2[3,4]3,0[5,7]3\}$. Due to there being a draw, both options should be taken if there are any additional sequences to be analyzed. Taking the first one as a result, the final chronicle is $(\mathcal{E},\mathcal{T})$, where $\mathcal{E}=\{A,B,C\}$, and $\mathcal{T}=\{A[2,3]B,B[1,3]C,A[4,5]C\}$.

Algorithm 2 Recursive Procedure for the Temporal Constraints

```
 1: procedure recursiveDraws($J, x, \mathcal{T}$)  ▷ Set of sets of subsequences $J$, index $x$ of $J$ for searching temporal constraints, set of temporal constraint $\mathcal{T}$
 2:     $\mathcal{T}_{best} \leftarrow \emptyset$
 3:     for $y$ in range [$x$, size of $J$] do
 4:         $I \leftarrow J[y]$
 5:         $\mathcal{T}_{min} \leftarrow \mathcal{T}_\infty$
 6:         $K \leftarrow \emptyset$
 7:         for all $i \in I$ do
 8:             $\mathcal{T}_{aux} \leftarrow$ updates $\mathcal{T}$ using $i$
 9:             if $\mathcal{T}_{aux} == \mathcal{T}$ then
10:                 break and continue with next set $I$ of subsequences
11:             end if
12:             if $sum(\mathcal{T}_{aux}) < sum(\mathcal{T}_{min})$ then
13:                 $K \leftarrow \emptyset$; insert $\mathcal{T}_{min}$ in $K$
14:                 $\mathcal{T}_{min} \leftarrow \mathcal{T}_{aux}$
15:             end if
16:             if $sum(\mathcal{T}_{aux}) == sum(\mathcal{T}_{min})$ then
17:                 insert $T_{min}$ in $K$
18:             end if
19:         end for
20:         if size($K$) > 1 then
21:             for all $k \in K$ do
22:                 $recursiveDraws(J, y+1, k)$  ▷ Recursive procedure
23:             end for
24:             return
25:         end if
26:         $\mathcal{T} \leftarrow K[1]$
27:     end for
28:     if $sum(\mathcal{T}) < sum(\mathcal{T}_{best})$ then
29:         $\mathcal{T}_{best} \leftarrow \mathcal{T}$
30:     end if
31: end procedure
```

C. MEGBIA ALGORITHM

The methodology we are presenting here is a bio-inspired algorithm, and to our knowledge, it is the first of its kind applied in the field of chronicle mining. The goal of this algorithm is to significantly reduce the substantial computational resources and memory needed when analyzing extensive datasets. When dealing with such datasets, where large sets of frequent patterns are extracted, exhaustive search algorithms may prove inordinately time-consuming, or may not even be complete.

An additional challenge in chronicle mining is accurately determining the threshold value. This task is not straightforward and usually requires deep domain-specific knowledge. Both novice and expert users find themselves

Algorithm 3 MEGBIA

```
Require: $\Omega$, $minLen$, $maxLen$, $minTemp$, $maxTemp$, $p_c$, $p_m$, $pSize$, $t$, $k$    ▷ Original dataset $\Omega$ (including timestamps), minimum $minLen$ and maximum $maxLen$ number of events, minimum $minTemp$ and maximum $maxTemp$ temporal distance between events, crossover probability $p_c$, mutation probabilty $p_m$, $pSize$ is the population size, $t$ is the tournament size, $k$ is the auxiliary population size
Ensure: $\mathcal{L}$
1: $\mathcal{L} \leftarrow \emptyset$    ▷ Best solution set
2: $\mathcal{P} \leftarrow \emptyset$    ▷ Population set
3: $I \leftarrow$ available events in $\Omega$
4: $exit \leftarrow False$
5: $\mathcal{P} \leftarrow$ creates $pSize$ initial solutions considering $I$, $minLen$, $maxLen$
6: $\mathcal{P} \leftarrow$ evaluates $\mathcal{P}$ and adjusts the temporal constraints using $\Omega$, and considering $minTemp$ and $maxTemp$
7: $\mathcal{L} \leftarrow$ gets the best $k$ solutions from $\mathcal{P}$
8: while $exit \neq True$ do
9:     $\mathcal{P}' \leftarrow$ applies tournament selector of size $t$ to $\mathcal{P}$
10:    $\mathcal{P}' \leftarrow$ applies crossover operator to $\mathcal{P}'$ using $p_c$ and $Items$
11:    $\mathcal{P}' \leftarrow$ applies mutation operator to $\mathcal{P}'$ using $p_m$ and $Items$
12:    $\mathcal{P}' \leftarrow$ evaluates $\mathcal{P}'$ and adjusts the temporal constraints using $\Omega$, and considering $minTemp$ and $maxTemp$
13:    $\mathcal{P} \leftarrow$ updates with elitism using $\mathcal{P}$ and $\mathcal{P}'$
14:    $\mathcal{L} \leftarrow$ gets the best $k$ solutions from $\mathcal{L} \cup \mathcal{P}$
15:    $exit \leftarrow True$ if the stop condition is reached
16: end while
17: return $\mathcal{L}$
```

having to estimate different threshold values, and then run the algorithm repeatedly until they achieve satisfactory results. In the realm of frequent pattern mining at large, it has been demonstrated by some researchers [29] that a slight alteration in the threshold value can lead to an extremely small (sometimes zero) or overwhelmingly large set of solutions. The latter scenario often necessitates additional filtering, and can significantly increase execution times. Consequently, there is a need for algorithms capable of extracting frequent event graphs without requiring any frequency threshold.

The proposed algorithm (see Algorithm 3), known as MEGBIA, follows a traditional evolutionary schema with elitism. MEGBIA is based on the same data representation as MEGMRTC, which was described in Section IV-A. The main procedures of this algorithm are described below.

- **Enconding criterion.** In this approach, each solution s represents a chronicle $\mathcal{C}$ defined as a tuple $(\mathcal{E}, \mathcal{T})$, following the formal definition given in Definition 5. The initial set $\mathcal{E}$ is randomly chosen from the set of feasible events I in the data (see line 5, Algorithm 3). The number of events taken to form $\mathcal{E}$ is a value in the range $[minLen, maxLen]$. These bounds are given by the user. It is the evolutionary process responsible for changing this set by adding and removing events to adjust the chronicle to form better solutions. The set $\mathcal{T}$ is obtained to accurately adjust the time constraints for every link between events from $\mathcal{E}$. Given the set $\mathcal{E} = \{0, 2, 3, 5\}$ with four different events, the set $\mathcal{T}$ will include 6 different time constraints. The relation between $\mathcal{E}$ and $\mathcal{T}$ is the size of $\mathcal{T}$ is the number of combinations of the elements in $\mathcal{E}$ taken 2 by to. Formally, it could be expressed as $|\mathcal{T}| = C_{|\mathcal{E}|,2}$. Last but not least, it is important to remark that the number of solutions (individuals) to be initialized is $pSize$, which is a predefined value.
- **Evaluation procedure.** This procedure is responsible for assigning a fitness value to each individual or solution (see line 5 and line 12, Algorithm 3). The fitness function evaluates how good a given solution is, and whether one individual or solution is better than another. This procedure is given by the frequency of the chronicle F and the sum of differences of ranges of $\mathcal{T}$, $sum(\mathcal{T})$. A solution s is said to be better than another one s' if $F_s > F_{s'}$ or $F_s == F_{s'} \neq 0$ and $sum(\mathcal{T})_s < sum(\mathcal{T})_{s'}$. It is important to keep in mind that chronicles with a support value of 0 are useless so they are discarded and no analysis of the ranges is required. In this evaluation procedure, the ranges of the temporal constraints are optimized similarly to Algorithm 1.
- **Selection operation.** The selection operator is a tournament selector of size t fixed by the user. It takes (with replacement, i.e. the same individual can be selected several times) t random individuals from the population, and the best individual is introduced into the parent set. This step is repeated $pSize$ times, so the parent set is formed by a total of $pSize$ individuals.
- **Crossover operation.** After selecting the parents, each pair of parents is crossed with a probability value p_c (see line 10, Algorithm 3). This genetic operator chooses a random item from each $\mathcal{E}$ of the pairs of parents and crosses (swaps) them. If the items to be crossed are the same event types, or there is no other available event type value in the pattern, the crossover cannot be performed. In other cases, it gets the values to be used and performs the swap.
- **Mutation operation.** Right after the crossover operator, the mutation genetic operator takes place. The mutation of the pattern $\mathcal{E}$ is carried out with a probability p_m (see line 11, Algorithm 3) considering three different options (depending on the length of the pattern and its restrictions): 1) to add an item; 2) to remove an item; 3) to change the item by another. When adding an item, a random event is selected and the next available value of the item is chosen. As a matter of clarification, let us consider the event-index data representation shown in Figure 7, and the pattern $\{0, 2, 3\}$ in which another

TABLE 3. Datasets and their characteristics. The datasets are ordered by increasing number of sequences.

Dataset	#Sequences	#Events	Mean Length	Std. dev. Length
SIGN	730	267	51.9973	12.3029
Synthetic5	1,000	5	8.5710	4.7420
Synthetic10	1,000	10	4.5390	1.1075
LEVIATHAN	5,834	9,025	33.8106	18.6090
kosarak10	10,000	10,094	8.1407	22.1080
kosarak15	15,000	11,871	8.1231	21.2148
kosarak20	20,000	13,678	8.1256	21.3454
FIFA	20,450	2,990	36.2392	24.0832
kosarak25	25,000	14,804	8.0425	20.7255
BIBLE	36,369	13,905	21.6411	12.2551
BMS1	59,601	497	2.5107	4.8528
BMS2	77,512	3,340	4.6222	6.0711

event A should be added. Then, the value 1 is added since it is the next available value for event A. If this is not possible, another event is chosen randomly. As for the item removal, the procedure is quite similar: an item is randomly chosen to be removed from the pattern $\mathcal{E}$. In such a situation in which the chosen item (event) appears more than once in $\mathcal{E}$, then the one with the highest value is removed. For example, considering again the event-index data representation shown in Figure 7, and the item 0 is chosen from the pattern $\{0, 1, 3\}$, then it is the value 1 the one that is removed since both (0 and 1) represent the event A. Finally, the third option is to change the value of an item from pattern $\mathcal{E}$. Again, in such a situation in which the chosen item (event) appears more than once in $\mathcal{E}$, then the one with the highest value is changed. Additionally, if the new event type cannot be introduced due to there is no additional feasible event type, then the operation is cancelled. For instance, let us suppose the pattern $\{0, 2, 3, 4, 5, 6, 7\}$ and we want to change the value 0 (event A according to the event-index data representation shown in Figure 7), the only available value is 1, which is of the same event type so the change is not performed.

- **Elitism.** To ensure that good solutions are not lost during the mining process, the algorithm uses elitism to keep the best solution found along the evolutionary process (see line 13, Algorithm 3). In this procedure, if the best solution belonging to the new population is worse than the best solution of the previous population, then that best solution replaces the worst one.
- **List of best solutions.** The k best solutions found during the execution of the algorithm are kept in a list or auxiliary population (see line 14, Algorithm 3).

All these procedures are combined to produce the proposed genetic algorithm (see Algorithm 3). The algorithm generates an initial set of *pSize* solutions from Ω, and these solutions are evaluated and their temporal constraints are assigned based on some minimum and maximum wide ranges (see lines 1 to 6, Algorithm 3). An evolutionary process (iterative procedure) is carried out and the best k solutions found along this process are kept in an auxiliary population $\mathcal{L}$. The evolutionary process carried out is a generational schema (see lines 8 to 16, Algorithm 3) with elitism (the best solution is never lost) and including well-known genetic operators: tournament selector of size t; crossover genetic operator to swap values between two solutions; and a mutation genetic operator to introduce diversity into the population (adding, removing or changing values). The stopping criterion of this proposal is based on the improvements done by the algorithm. Thus, if the algorithm is not able to improve the auxiliary population after a predefined number of iterations, then it returns such a population (including the best solutions found so far). This algorithm also returns the auxiliary population after a predefined number of iterations, or when the runtime limit is reached.

V. EXPERIMENTS

This section presents the experimental study. First, it describes the datasets and the parameters used in this study. Then, it performs a detailed analysis on the performance of all the proposed algorithms.

A. EXPERIMENTAL SETUP

The experiments employed twelve datasets, mainly from the SPMF library [10], along with two synthetically generated ones, Synthetic5 and Synthetic10. The datasets vary significantly in size and characteristics, as shown in Table 3. The number of sequences spans from 730 to nearly 78,000, while unique events range from 5 to about 15,000. The average events per sequence vary from 2 to around 52, suggesting a high possibility of multiple subsequences satisfying a chronicle. For the parameters, the minimum chronicle length is set to 3, with temporal constraints predefined as the universal constraint $(-\infty, \infty)$. The genetic algorithm parameters include a tournament size of 3, an auxiliary population size of 30, a crossover probability of 0.3, and a mutation probability of 0.9, as high mutation values promote solution diversity in pattern mining [27]. The algorithm halts when 250 iterations pass without improvements or at a total of 2,500 iterations. Each experiment was repeated five times for more accurate results.

TABLE 4. Average of ranges achieved by the approaches under study. *Time out* denotes that the algorithm took more than 3,600 seconds.

Dataset	Freq.Threshold	CPM	MEGMRTC
SIGN	0.6000	331.7767	331.7767
Synthetic5	0.5000	*Time out*	121.5780
Synthetic10	0.0350	25.7529	25.7529
LEVIATHAN	0.2250	*Time out*	425.2063
kosarak10	0.0200	585.4400	585.4400
kosarak15	0.0200	721.6600	721.6600
kosarak20	0.0200	758.7451	758.7451
FIFA	0.3425	389.6622	389.6622
kosarak25	0.0200	771.4600	771.4600
BIBLE	0.1250	*Time out*	388.3115
BMS1	0.0025	189.1053	189.1053
BMS2	0.0055	12.3544	12.3544

B. ANALYSIS OF PROPOSED MEGMRTC ALGORITHM

Our goal here is to compare the MEGMRTC algorithm's performance with the latest chronicle mining algorithm, CPM [24]. CPM employs the CloSpan algorithm [31] to extract frequent patterns and time constraints between these patterns. However, this can lead to different chronicles, as explained in Section III-A and associated with Property 6. Therefore, we have made a minor adjustment to CPM [24] by incorporating the LCM algorithm [25] in the frequent pattern generation phase. The comparison involves three aspects: temporal constraint adjustment, runtime, and memory usage.

Table 4 illustrates that our proposed MEGMRTC algorithm accurately adjusts temporal constraints, as it returns results identical to those of CPM, an exhaustive search method. In cases where CPM could not return any chronicle within 3,600 seconds, MEGMRTC continued to perform. Furthermore, Table 5 shows that our MEGMRTC algorithm obtains chronicles in less time than CPM. The Wilcoxon signed-rank test revealed significant differences between the two algorithms (p-value=0.00374), confirming superior performance by our proposed algorithm at a 99% significance level. Our proposed method was compared to CPM [24] in terms of memory requirements, revealing that ours generally consumes less memory (see Table 6). While no statistically significant difference was found at a 99% level (p-value=0.09176), our method performed better at a 90% level according to the Wilcoxon signed-rank test.

At this point, it is important to analyze the runtime in situations where CPM did not return any value due to the time limit. The aim of this analysis is to stress the algorithms by considering different frequency threshold values till we come to the limit, particularly in cases where CPM exceeded the time limit (Synthetic5, LEVIATHAN, and BIBLE datasets). As shown in Tables 7, 8 and 9, our method returned results more rapidly, even as frequency thresholds decreased. Memory usage between the two algorithms displayed minor differences, with our method typically requiring less (see Tables 10, 11 and 12). For the BIBLE dataset, our method performed considerably faster and with less memory requirement than CPM, even as the frequency threshold decreased to 0.1 (see Tables 9 and 12). Interestingly, memory usage by our method showed a significant increase

TABLE 5. Average runtime (seconds) after running the algorithms 5 times. The best results are highlighted in bold type-face. *Time out* denotes that the algorithm took more than 3,600 seconds.

Dataset	Freq.Threshold	CPM	MEGMRTC
SIGN	0.6000	7.5152	**4.4967**
Synthetic5	0.5000	*Time out*	**171.3591**
Synthetic10	0.0350	0.1884	**0.1861**
LEVIATHAN	0.2250	*Time out*	**132.35713**
kosarak10	0.0200	2.1811	**1.1577**
kosarak15	0.0200	2.2314	**1.7965**
kosarak20	0.0200	**2.4336**	2.4455
FIFA	0.3425	133.9492	**103.0651**
kosarak25	0.0200	6.1018	**3.0686**
BIBLE	0.1250	*Time out*	**690.8685**
BMS1	0.0025	4.1377	**2.1312**
BMS2	0.0055	6.9869	**3.9656**

TABLE 6. Average memory requirements (MB) of 5 different runs. The best results are highlighted in bold type-face. *Time out* denotes that the algorithm took more than 3,600 seconds.

Dataset	Freq.Threshold	CPM	MEGMRTC
SIGN	0.6000	**19.3**	19.5
Synthetic5	0.5000	*Time out*	**175.8**
Synthetic10	0.0350	35.3	**30.1**
LEVIATHAN	0.2250	*Time out*	**147.7**
kosarak10	0.0200	**22.5**	40.9
kosarak15	0.0200	**56.9**	57.0
kosarak20	0.0200	**72.8**	72.3
FIFA	0.3425	251.1	**241.5**
kosarak25	0.0200	**86.8**	87.1
BIBLE	0.1250	*Time out*	**721.5**
BMS1	0.0025	89.3	**87.4**
BMS2	0.0055	166.3	**158.1**

TABLE 7. Average runtime (seconds) after running the algorithms 5 times on reduced versions of the Synthetic5 dataset (see Table 3) using different threshold values. Time out denotes that the algorithm took more than 3,600 seconds.

Freq. Threshold	CPM	MEGMRTC
0.9	0.1	0.1
0.8	1.2	2.1
0.7	27.8	25.6
0.6	1,397.4	130.2
0.5	*Time out*	171.4

TABLE 8. Average runtime (seconds) after running the algorithms 5 times on reduced versions of the LEVIATHAN dataset (see Table 3) using different threshold values. Time out denotes that the algorithm took more than 3,600 seconds.

Freq. Threshold	CPM	MEGMRTC
0.6	0.5	0.5
0.5	1.4	1.4
0.4	*Time out*	3.2
0.3	*Time out*	15.4
0.2	*Time out*	146.2

TABLE 9. Average runtime (seconds) after running the algorithms 5 times on reduced versions of the BIBLE dataset (see Table 3) using different threshold values. Time out denotes that the algorithm took more than 3,600 seconds.

Freq. Threshold	CPM	MEGMRTC
0.5	2.7	2.8
0.4	2.8	2.8
0.3	18.6	15.9
0.2	2,569.1	64.4
0.1	*Time out*	1,583.6

TABLE 10. Average memory requirements (MB) after running the algorithms 5 times on Synthetic5 dataset (see Table 3) using different threshold values. Time out denotes that the algorithm took more than 3,600 seconds.

Freq. Threshold	CPM	MEGMRTC
0.9	34.6	54.1
0.8	59.8	65.3
0.7	65.5	61.2
0.6	158.2	150.1
0.5	*Time out*	175.8

TABLE 11. Average memory requirements (MB) after running the algorithms 5 times on LEVIATHAN dataset (see Table 3) using different threshold values. Time out denotes that the algorithm took more than 3,600 seconds.

Freq. Threshold	CPM	MEGMRTC
0.6	71.4	65.5
0.5	71.4	65.7
0.4	*Time out*	76.6
0.3	*Time out*	90.2
0.2	*Time out*	148.1

TABLE 12. Average memory requirements (MB) after running the algorithms 5 times on BIBLE dataset (see Table 3) using different threshold values. Time out denotes that the algorithm took more than 3,600 seconds.

Freq. Threshold	CPM	MEGMRTC
0.5	223.3	237.1
0.4	252.2	238.4
0.3	286.8	271.1
0.2	397.1	363.2
0.1	*Time out*	1,431.1

when the frequency threshold was reduced from 0.125 to 0.1, highlighting the impact of minor frequency threshold changes on resource requirements. This phenomenon aligns with findings reported by *Wu* et al. [29] in their frequent itemset mining study.

C. ANALYSIS OF PROPOSED MEGBIA ALGORITHM

In our second study, we evaluate the performance of MEGBIA, an evolutionary approach that does not require a frequency threshold. Its key advantage is its ability to extract frequent chronicles without this threshold, despite its typically longer runtime compared to exhaustive search methods. Table 13 outlines results for each dataset, including the number of chronicles returned, the average temporal constraints, runtime, and memory requirements. Unlike exhaustive methods, MEGBIA only returns the top 30 highest-frequency chronicles, making a direct comparison based on temporal constraints inappropriate due to the difference in the number of chronicles.

MEGBIA's solutions were compared to the best solutions from exhaustive methods. Although MEGBIA's solutions do not always rank high compared to exhaustive methods, its main advantage is its ability to extract frequent event graphs without a support threshold and without significant runtime increase. To analyze how good the solutions returned by the genetic algorithm are, we compare them to the best solutions found by the exhaustive approaches (the optimal temporal constraints were obtained as they analyzed all the solutions in the data). Table 14 shows the percentage of solutions given by the genetic algorithm that is in the top-30 of best solutions of the exhaustive search approaches. In general, except for some datasets, this percentage is not so high. However, it is important to remark that the main advantage of this evolutionary approach is its ability to extract frequent event graphs without needing any support threshold value and not increasing the runtime so much.

VI. CASE STUDY: SLEEP DISORDER

Sleep has an essential restorative function for physical and mental health and any sleep disorder has a huge impact on health life quality. It has brought the necessity of performing sleep studies where a machine is monitoring and recording many body activities that occur during sleep, including brain activity, heart rate, breathing, leg movements, and oxygen level. The Sleep Medicine Centre of the Hospital of Coimbra University has recently gathered information [17] from a sleep study on 100 different patients. The study includes data on healthy subjects, subjects with sleep disorders, and subjects under the effect of sleep medication. For each subject under study, different events were obtained every 30 seconds (1 epoch) including the following:

- LOut. Lights turn off.
- LON. Lights turn on.
- MChg. A montage change occurs in the system.
- PLM. This is the acronym for Periodic Leg Movement. This represents episodes of simple, repetitive muscle movements.
- CA. Central Apnea, which are pauses in breathing due to a lack of respiratory effort during sleep.
- LM. A leg movement is recorded.
- MP. Any periodic movement of the body is recored-
- OH. Obstructive Hypopnea, denoting shallow breathing episodes, called hypopneas, while sleeping.
- OA. Obstructive Apnea, characterizing recurrent episodes of complete or partial obstruction of the upper airway leading to reduced or absent breathing during sleep.

TABLE 13. Summary of the results returned by the proposed MEGBIA algorithm after running it 5 times. Time out denotes that the algorithm took more than 3,600 seconds.

Dataset	#Chronicles	Avg. ranges	Runtime	Memory
SIGN	30.0	291.6200	66.7687	22.3
Synthetic5	30.0	42.6600	51.7324	190.7
Synthetic10	30.0	22.8667	1.1790	38.9
LEVIATHAN	30.0	287.1981	354.9890	151.8
kosarak10	30.0	420.0400	13.0303	45.2
kosarak15	30.0	498.4867	16.2278	63.8
kosarak20	30.0	599.7200	17.5198	80.2
FIFA	30.0	269.8291	231.8101	274.8
kosarak25	30.0	475.2733	28.0753	95.6
BIBLE	30.0	201.7762	2,187.9098	801.1
BMS1	30.0	78.0600	9.9105	95.3
BMS2	3.8	0.3200	3.9119	173.8

TABLE 14. Percentages of best solutions found by the MEGBIA algorithm.

Dataset	% Best solutions
SIGN	12.00%
Synthetic5	72.67%
Synthetic10	20.67%
LEVIATHAN	38.65%
kosarak10	42.67%
kosarak15	35.33%
kosarak20	36.67%
FIFA	11.33%
kosarak25	42.00%
BIBLE	22.13%
BMS1	00.00%
BMS2	00.00%

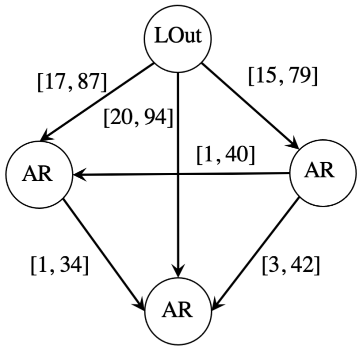


FIGURE 8. Frequent complete event graph satisfying 54% of the subjects under study.

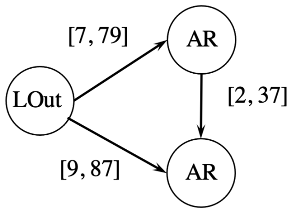


FIGURE 9. Frequent complete event graph satisfying 74% of the subjects under study.

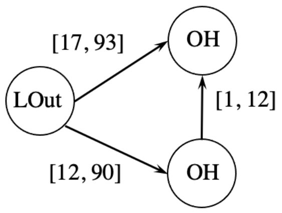


FIGURE 10. Complete event graph related to Obstructive Hypopnea and satisfying 20% of the subjects under study.

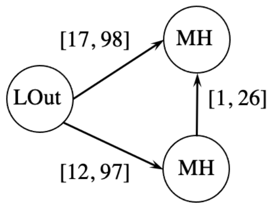


FIGURE 11. Complete event graph related to Mixed Hypopnea and satisfying 20% of the subjects under study.

- REMAw. This event represents the action of awakening in REM period.
- AR. Arousal is an abrupt change from sleep to wakefulness.
- MH. Mixed Hypopnea or shallow breathing in which the airflow in and out of the airway is less than half of normal. It is usually associated with oxygen desaturation.

This study aims to elucidate frequent sleep-related bio-physiological changes via descriptive event graphs. Figure 8 represents a prevalent event graph, observed in 54% of subjects, depicting frequent transitions from sleep to wakefulness. The temporal constraints are denoted in epochs (30 seconds). Starting from lights-off, three arousals occur within an hour: the first occurs between 15 and 79 epochs (between 7 and 40 minutes after lights turn off), followed by two more within 20 minutes of the preceding one. A simplified version, Figure 9, is found in 74% of patients. It details two arousals: the first within 3-40 minutes post-lights-off, and a second within 1-19 minutes, spanning no more than 43 minutes from lights-off to the second arousal. Event graphs also encompass Hypopnea-related events. Figure 10, found in 20.8% of subjects, showcases Obstructive Hypopnea occurring twice: first within 6-45 minutes post-lights-off, and subsequently within 6 minutes. Figure 11 describes Mixed Hypopnea, observed in 20% of subjects. Two events occur: initially within 6-48 minutes after lights-off, followed by a second within 13 minutes. This highlights that around a quarter of patients experience two Mixed Hypopnea events within the first 49 minutes post-lights-off.

VII. CONCLUSION

Chronicle mining appears as a promising research area, and it is our understanding that it should be studied more in-depth. As it has been described, very few articles can be found in the literature, and most of them were proposed for predictive maintenance purposes. None of such research studies proposes approaches for mining descriptive information from data (temporal constraint networks). This descriptive information is more restrictive and requires extra requirements from the algorithms such as correctness and completeness. This article has proposed some formal definitions and properties that were needed to formalize this research area. This article has also proposed two algorithms for chronicle mining from a descriptive point of view: an exhaustive search approach and a bio-inspired algorithm. The approaches have been compared to the best algorithm so far, which is the CPM algorithm. The results have shown good performance in terms of runtime, memory requirement and scalability. Additionally, this research work has presented the application of chronicle mining to real-world data related to sleep disorders. Alluring frequent event graphs were obtained on patients who were under the effect of sleep medication.

REFERENCES

[1] M. T. Alam, A. Roy, C. F. Ahmed, M. A. Islam, and C. K. Leung, "UGMINE: Utility-based graph mining," *Int. J. Speech Technol.*, vol. 53, no. 1, pp. 49–68, Jan. 2023.

[2] M. R. Álvarez, P. Félix, and P. Cariñena, "Discovering metric temporal constraint networks on temporal databases," *Artif. Intell. Med.*, vol. 58, no. 3, pp. 139–154, Jul. 2013.

[3] R. Bellazzi, C. Larizza, P. Magni, and R. Bellazzi, "Temporal data mining for the quality assessment of hemodialysis services," *Artif. Intell. Med.*, vol. 34, no. 1, pp. 25–39, May 2005.

[4] Y. Chen, "Discovering time-interval sequential patterns in sequence databases," *Expert Syst. Appl.*, vol. 25, no. 3, pp. 343–354, Oct. 2003.

[5] D. Cram, B. Mathern, and A. Mille, "A complete chronicle discovery approach: Application to activity analysis," *Expert Syst.*, vol. 29, no. 4, pp. 321–346, Sep. 2012.

[6] Y. Dauxais, T. Guyet, D. Gross-Amblard, and A. Happe, "Discriminant chronicles mining," in *Proc. Conf. Artif. Intell. Med. Eur.* Cham, Switzerland: Springer, 2017, pp. 234–244.

[7] Y. Dauxais, T. Guyet, D. Gross-Amblard, and A. Happe, "Discriminant chronicles mining: Application to care pathways analytics," in *Proc. 16th Conf. Artif. Intell. Med.*, in Lecture Notes in Computer Science, vol. 10259, A. T. Teije, C. Popow, J. H. Holmes, and L. Sacchi, Eds. Cham, Switzerland: Springer, 2017, pp. 234–244.

[8] C. Dousson and T. V. Duong, "Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems," in *Proc. IJCAI*, vol. 99, 1999, pp. 620–626.

[9] M. Dresler, V. I. Spoormaker, P. Beitinger, M. Czisch, M. Kimura, A. Steiger, and F. Holsboer, "Neuroscience-driven discovery and development of sleep therapeutics," *Pharmacol. Therapeutics*, vol. 141, no. 3, pp. 300–334, Mar. 2014.

[10] P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, and H. T. Lam, "The SPMF open-source data mining library version 2," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, 2016, pp. 36–40.

[11] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Sci. Pattern Recognit.*, vol. 1, no. 1, pp. 54–77, 2017.

[12] P. Fournier-Viger, Y. Yang, P. Yang, J. C.-W. Lin, and U. Yun, "TKE: Mining top-k frequent episodes," in *Proc. Int. Conf. Ind., Eng. Appl. Appl. Intell. Syst.*, in Lecture Notes in Computer Science, vol. 12144, H. Fujita, P. Fournier-Viger, M. Ali, and J. Sasaki, Eds., 2020, pp. 832–845.

[13] T. Guyet and P. Besnard, *Mining Temporal Data With Chronicles*. Berlin, Germany: Springer, 2023.

[14] Y.-H. Hu, T. C.-K. Huang, H.-R. Yang, and Y.-L. Chen, "On mining multi-time-interval sequential patterns," *Data Knowl. Eng.*, vol. 68, no. 10, pp. 1112–1127, Oct. 2009.

[15] K.-Y. Huang and C.-H. Chang, "Efficient mining of frequent episodes from complex sequences," *Inf. Syst.*, vol. 33, no. 1, pp. 96–114, Mar. 2008.

[16] M. S. Islam, P. C. Kar, M. Samiullah, C. F. Ahmed, and C. K.-S. Leung, "Discovering probabilistically weighted sequential patterns in uncertain databases," *Appl. Intell.*, vol. 53, no. 6, pp. 6525–6553, Mar. 2023.

[17] S. Khalighi, T. Sousa, J. M. Santos, and U. Nunes, "ISRUC-sleep: A comprehensive public dataset for sleep researchers," *Comput. Methods Programs Biomed.*, vol. 124, pp. 180–192, Feb. 2016.

[18] N. Li, X. Yao, and D. Tian, "Mining temporal sequential patterns based on multi-granularities," *Int. J. Comput. Commun. Control*, vol. 7, no. 3, p. 494, Sep. 2014.

[19] Y. Li, G. Wang, Y. Yuan, X. Cao, L. Yuan, and X. Lin, "*PrivTS*: Differentially private frequent time-constrained sequential pattern mining," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, in Lecture Notes in Computer Science, J. Pei, Y. Manolopoulos, S. W. Sadiq, and J. Li, Eds., 2018, pp. 92–111.

[20] M.-Y. Lin, S.-C. Hsueh, and C.-W. Chang, "Fast discovery of time-constrained sequential patterns using time-indexes," in *Proc. Int. Conf. Adv. Data Mining Appl.*, in Lecture Notes in Computer Science, vol. 4093, X. Li, O. R. Zaïane, and Z. Li, Eds. Xi'an, China, 2006, pp. 693–701.

[21] J. M. Luna, P. Fournier-Viger, and S. Ventura, "Frequent itemset mining: A 25 years review," *WIREs Data Mining Knowl. Discovery*, vol. 9, no. 6, Nov. 2019.

[22] J. M. Luna, P. Fournier-Viger, and S. Ventura, "Extracting user-centric knowledge on two different spaces: Concepts and records," *IEEE Access*, vol. 8, pp. 134782–134799, 2020.

[23] J. M. Luna, F. Padillo, M. Pechenizkiy, and S. Ventura, "Apriori versions based on MapReduce for mining frequent patterns on big data," *IEEE Trans. Cybern.*, vol. 48, no. 10, pp. 2851–2865, Oct. 2018.

[24] C. Sellami, A. Samet, and M. A. Bach Tobji, "Frequent chronicle mining: Application on predictive maintenance," in *Proc. 17th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2018, pp. 1388–1393.

[25] T. Uno, M. Kiyomi, and H. Arimura, "LCM ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining," in *Proc. 1st Int. Workshop Open Source Data Mining, Frequent Pattern Mining Implement.*, Aug. 2005, pp. 77–86.

[26] W. M. P. van der Aalst and A. J. M. M. Weijters, "Process mining: A research agenda," *Comput. Ind.*, vol. 53, no. 3, pp. 231–244, Apr. 2004.

[27] S. Ventura and J. M. Luna, *Pattern Mining With Evolutionary Algorithms*. Berlin, Germany: Springer, 2016.

[28] J.-Z. Wang, Y.-C. Chen, W.-Y. Shih, L. Yang, Y.-S. Liu, and J.-L. Huang, "Mining high-utility temporal patterns on time interval–based data," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 4, May 2020.

[29] C. W. Wu, B.-E. Shie, V. S. Tseng, and P. S. Yu, "Mining top-k high utility itemsets," in *Proc. 18th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Beijing, China, Aug. 2012, pp. 78–86.

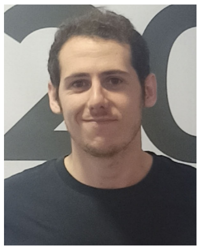
[30] S.-Y. Wu and Y.-L. Chen, "Discovering hybrid temporal patterns from sequences consisting of point- and interval-based events," *Data Knowl. Eng.*, vol. 68, no. 11, pp. 1309–1330, Nov. 2009.

[31] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining: Closed sequential patterns in large datasets," in *Proc. SIAM Int. Conf. Data Mining*, May 2003, pp. 166–177.

[32] M. Yoshida, T. Iizuka, H. Shiohara, and M. Ishiguro, "Mining sequential patterns including time intervals," *Proc. SPIE*, vol. 4057, pp. 213–220, Apr. 2000.

HARETH ZMEZM received the Master of Computer Science degree in network technology from University Kebangsaan Malaysia (UKM). He is currently an Academic Lecturer with the College of Science Computer Science and the Biology Department, Kerbala University, Iraq. He has published several journal articles and international scientific conferences. His research interests include network security, AI, data mining, and medical applications.

JOSÉ MARÍA LUNA received the Ph.D. degree in computer science from the University of Granada, Spain, in 2014. He is currently an Associate Professor with the Department of Computer Science and Numerical Analysis, University of Córdoba, Spain. He is the author of the two books related to pattern mining, published by Springer. He has published more than 35 papers in top ranked journals and international scientific conferences, and he is the author of two book chapters. He has also been engaged in four national and regional research projects. He has contributed to three international projects. His research interests include evolutionary computation and pattern mining.



EDUARDO ALMEDA is currently pursuing the Ph.D. degree with the KDIS Laboratory Research Group, University of Córdoba, Spain. He was involved on frequent pattern mining and chronicle mining projects. His research interests include healthcare applications, including radiomics and early detection of cancer.

SEBASTIÁN VENTURA (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees in sciences from the University of Córdoba, Spain, in 1989 and 1996, respectively. He is currently a Full Professor with the Department of Computer Science and Numerical Analysis, University of Córdoba, where he heads the Knowledge Discovery and Intelligent Systems Research Laboratory. He has published three books and about 300 papers in journals and scientific conferences, and edited three books and several special issues in international journals. He has also been engaged in 15 research projects (being the coordinator of seven of them) supported by the Spanish and Andalusian governments and the European Union. His research interests include data science, computational intelligence, and their applications. He is a Senior Member of the IEEE Computer, the IEEE Computational Intelligence, the IEEE Systems, Man, and Cybernetics societies, and the Association of Computing Machinery (ACM).