Efficient Generalized Temporal Pattern Mining in Big Time Series Using Mutual Information

Ho Long, Van; Ho, Nguyen Thi Thao; Pedersen, Torben Bach; Papapetrou, Panagiotis

Publication date:
2023

Efficient Generalized Temporal Pattern Mining in Big Time Series Using Mutual Information

Van Long Ho, Nguyen Ho, Torben Bach Pedersen, *CS Distinguished Contributor, IEEE,* and Panagiotis Papapetrou

Abstract—Big time series are increasingly available from an ever wider range of IoT-enabled sensors deployed in various environments. Significant insights can be gained by mining temporal patterns from these time series. Temporal pattern mining (TPM) extends traditional pattern mining by adding event time intervals into extracted patterns, making them more expressive at the expense of increased time and space complexities. Besides frequent temporal patterns (FTPs), which occur frequently in the entire dataset, another useful type of temporal patterns are so-called *rare temporal patterns (RTPs)*, which appear rarely but with high confidence. Mining rare temporal patterns yields additional challenges. For FTP mining, the temporal information and complex relations between events already create an exponential search space. For RTP mining, the support measure is set very low, leading to a further combinatorial explosion and potentially producing too many uninteresting patterns. Thus, there is a need for a generalized approach which can mine both frequent and rare temporal patterns. This paper presents our *Generalized Temporal Pattern Mining from Time Series (GTPMfTS)* approach with the following specific contributions: (1) The end-to-end GTPMfTS process taking time series as input and producing frequent/rare temporal patterns as output. (2) The efficient *Generalized Temporal Pattern Mining (GTPM)* algorithm mines frequent and rare temporal patterns using efficient data structures for fast retrieval of events and patterns during the mining process, and employs effective pruning techniques for significantly faster mining. (3) An approximate version of GTPM that uses mutual information, a measure of data correlation, to prune unpromising time series from the search space. (4) An extensive experimental evaluation of GTPM for rare temporal pattern mining (RTPM) and frequent temporal pattern mining (FTPM), showing that RTPM and FTPM signficantly outperform the baselines on runtime and memory consumption, and can scale to big datasets. The approximate RTPM is up to one order of magnitude, and the approximate FTPM up to two orders of magnitude, faster than the baselines, while retaining high accuracy.

Index Terms—Temporal Pattern Mining, Rare Temporal Patterns, Time Series, Mutual Information.

1 Introduction

IoT-enabled sensors have enabled the collection of many big time series, e.g., from smart-meters, -plugs, and -appliances in households, weather stations, and GPS-enabled mobile devices. Extracting patterns from these time series can offer new domain insights for evidence-based decision making and optimization. As an example, consider Fig. 1 that shows the electricity usage of a water boiler with a hot water tank collected by a 20 euro Wifi-enabled smart-plug, and accurate CO2 intensity (g/kWh) forecasts of local electricity, e.g., as supplied by the Danish Transmission System Operator [1]. From Fig. 1, we can identify several useful patterns. First, the water boiler switches *On* once a day, for one hour between 6 and 7AM. This indicates that the resident takes only one hot shower per day which starts between 5.30 and 6.30AM. Second, all water boiler *On* events are contained in CO2 *High* events, i.e., the periods when CO2 intensity is high. Third, between two consecutive *On* events of the boiler, there is a CO2 *Low* event lasting for one or more hours which occurs at most 4 hours before the hot shower (so water heated during that event will still be hot at 6AM). Pattern mining can be used to extract the relations between CO2 intensity and water boiler events. However, traditional sequential patterns only capture the sequential occurrence of events, e.g., that one boiler *On* event follows after another, but not that there is at least 23 hours between them; or that there is a CO2 *Low* event between the two boiler *On* events, but not when or for how long it lasts. In contrast, *temporal pattern mining* (TPM) adds temporal information into patterns, providing details on when certain relations between events happen, and for how long. For example, TPM expresses the above relations as: ([7:00 - 8:00, Day X] BoilerOn $\rightarrow$ [6:00 - 7:00, Day X+1] BoilerOn) (meaning BoilerOn is followed by BoilerOn the next day), ([6:00 - 10:00, Day X] HighCO2 $\succcurlyeq$ [7:00 - 8:00, Day X] BoilerOn) (meaning HighCO2 contains BoilerOn), and ([7:00 - 8:00, Day X] BoilerOn $\rightarrow$ [0:00 - 2:00, Day X+1]

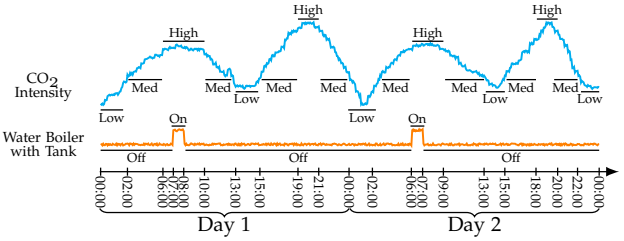


Fig. 1: CO2 intensity and water boiler electricity usage

- *Van Long Ho, Nguyen Ho, and Torben Bach Pedersen are with the Department of Computer Science, Aalborg University, 9100 Aalborg, Denmark. E-mail: {vlh,ntth,tbp}@cs.aau.dk.*
- *Panagiotis Papapetrou is with the Department of Computer and Systems Sciences, Stockholm University, 7003 Stockholm, Sweden. E-mail: panagiotis@dsv.su.se.*

LowCO2 $\rightarrow$ [6:00 - 7:00, Day X+1] BoilerOn) (meaning there is a LowCO2 event between two BoilerOn events). As the resident is very keen on reducing her CO2 footprint, we can rely on the above temporal patterns to automatically (using the smart-plug) delay turning on the boiler until the CO2 intensity is low again, saving CO2 without any loss of comfort for the resident. In the smart city domain, temporal patterns extracted from vehicle GPS data [2] can reveal spatio-temporal correlations between traffic jams, advising drivers to take another route for their morning commute.

Finding frequent temporal patterns (FTPs) is useful; however, in many applications, some patterns appear rarely but are still very interesting and useful due to high confidence. We call such patterns *rare temporal patterns (RTPs)*. For example, considering smart city applications, a rare pattern could be: ([20:00, 22:00] Snow $\succcurlyeq$ [20:15, 21:15] HighWind $\rightarrow$ [21:20, 21:50] HighInjuryMotorist), which means that the coincidence of snow and strong winds leads to traffic accidents within an hour. This pattern occurs rarely but supports transportation coordinators in warning citizens about traffic accidents. In health care, identifying symptoms and relations among them supports health experts in diagnosing diseases in the early phases.

Challenges of mining frequent temporal patterns. Mining temporal patterns is much more expensive than mining sequential patterns. Not only does the temporal information add extra computation to the mining process, the complex relations between events also add an additional exponential factor O(3^{h^2}) to the O(m^h) search space complexity (m is the number of events and h is the length of temporal patterns), yielding an overall complexity of O($m^h 3^{h^2}$) (see Lemma 2 in Section 4.4). Existing TPM methods [3], [4], [5] do not scale to big datasets, i.e., many time series and many sequences, and/or do not work directly on time series but only on preprocessed temporal events.

Challenges of mining rare temporal patterns. The support measure represents the frequency of a temporal pattern across the entire dataset. However, to find rare temporal patterns, the support has to be set very low, which causes a combinatorial explosion, potentially producing too many patterns that are uninteresting to the user. Existing work proposes solutions to mine rare itemsets [6], [7], [8], [9] and rare sequential patterns [10], [11], [12]. However, they do not consider the temporal aspect of items/events. Thus, addressing the explosion of rare temporal patterns with high confidence is still an open problem.

Generalized temporal pattern mining. Since there are many joint challenges in mining frequent and rare temporal patterns, there is a need for a *generalized* approach that can mine both types of patterns efficiently.

Contributions. In this paper, we present our comprehensive *Generalized Temporal Pattern Mining from Time Series (GTPMfTS) approach* which solves the above challenges. The paper significantly extends a previous conference paper [13]. Our key contributions are: (1) We present *end-to-end GTPMfTS process* that receives time series as input, and produces frequent/rare temporal patterns as output. Within this process, a splitting strategy is proposed to convert time series into event sequences while ensuring the preservation of temporal patterns. (2) We propose the *efficient Generalized Temporal Pattern Mining (GTPM) algorithm* to mine both frequent and rare temporal patterns. The novelties of GTPM are: a) the use of an efficient data structure, Hierarchical Hash Tables, to enable fast retrieval of events and patterns during the mining process; and b) pruning techniques based on the Apriori principle and the transitivity property of temporal relations to enable faster mining. (3) Based on the information theory concept of mutual information, which measures the correlation among time series, we propose a novel *approximate version of GTPM* that prunes unpromising time series to significantly reduce the search space and can scale on big datasets, i.e., many time series and many sequences. (4) We perform extensive experiments on synthetic and real-world datasets for both rare temporal pattern mining (RTPM) and frequent temporal pattern mining (FTPM), showing that our RTPM and FTPM significantly outperform the baselines on both runtime and memory usage. Compared to the baselines, the approximate RTPM has up to one order of magnitude speedup, and the approximate FTPM up to two orders of magnitude speedup, while retaining high accuracy compared to the exact algorithms.

Compared to the the conference version [13], this paper generalizes the TPM problem, to mine both frequent and (the novel proposal of) rare temporal patterns. For FTPM, this paper uses Hierarchical Hash Tables to retrieve events and patterns quickly, a significant improvement over the Hierarchical Pattern Graph in the conference version [13]. Moreover, we now combine the lower bound of support and the lower bound of confidence from the conference version [13] for the approximate FTPM to further accelerate the mining. For RTPM, we introduce the first exact and approximate algorithms to mine rare temporal patterns. In the present paper, we further provide a set of new experiments to compare our algorithms with the baselines.

Paper Outline. The paper is structured as follows. Section 2 discusses the related work. Section 3 formulates the generalized temporal pattern mining problem. Section 4 describes the exact GTPM algorithm. Section 5 presents the approximate GTPM algorithm. Section 6 presents the experimental evaluation. Finally, Section 7 concludes and points to future work.

2 Related Work

Temporal pattern mining: Compared to sequential pattern mining, TPM is rather a new research topic. One of the first papers in this area is of Kam et al. that uses a hierarchical representation to manage temporal relations [14], and based on that mines temporal patterns. However, the approach in [14] suffers from *ambiguity* when presenting temporal relations. For example, using the representation in [14], it is possible to have two temporal patterns that involve the same set of temporal events, for example, (((a overlaps b) before c) overlaps d), and ((a overlaps b) before (c contains d)). Thus, the same set of events can be mapped to different temporal patterns that are semantically different. Our GTPM avoids this ambiguity by defining a temporal pattern as a set of pairwise temporal relations between two events. In [15], Wu et al. develop TPrefix to mine temporal patterns from non-ambiguous temporal relations. However, TPrefix has several inherent limitations: it scans the database repeatedly, and the algorithm does not employ any pruning strategies to

reduce the search space. In [16], Moskovitch et al. design a TPM algorithm using the transitivity property of temporal relations. They use this property to generate candidates by inferring new relations between events. In comparison, our GTPM uses the transitivity property for effective pruning. In [17], Iyad et al. propose a TPM framework to detect events in time series. However, their focus is to find irregularities in the data. In [18], Wang et al. propose a temporal pattern mining algorithm HUTPMiner to mine high-utility patterns. Different from our GTPM which uses *support* and *confidence* to measure the frequency of patterns, HUTPMiner uses *utility* to measure the importance or profit of an event/ pattern, thereby addresses an orthogonal problem. In [19], Amit et al. propose STIPA which uses a Hoeppner matrix representation to compress temporal patterns for memory savings. However, STIPA does not use any pruning/ optimization strategies and thus, despite the efficient use of memory, it cannot scale to large datasets, unlike our GTPM. Other work [20], [21] proposes TPM algorithms to classify health record data. However, these methods are very domain-specific, thus cannot generalize to other domains.

The state-of-the-art TPM methods that currently achieve the best performance are our baselines: H-DFS [5], TPMiner [3], IEMiner [4], and Z-Miner [22]. H-DFS is a hybrid algorithm that uses breadth-first and depth-first search strategies to mine frequent arrangements of temporal intervals. H-DFS uses a data structure called ID-List to transform event sequences into vertical representations, and temporal patterns are generated by merging the ID-Lists of different events. This means that H-DFS does not scale well when the number of time series increases. In [4], Patel et al. design a hierarchical lossless representation to model event relations, and propose IEMiner that uses Apriori-based optimizations to efficiently mine patterns from this new representation. In [3], Chen et al. propose TPMiner that uses endpoint and endtime representations to simplify the complex relations among events. Similar to [5], IEMiner and TPMiner do not scale to datasets with many time series. Z-Miner [22], proposed by Lee et al., is the most recent work addressing TPM. Z-Miner improves the mining efficiency over existing methods by employing two data structures: a hierarchical hash-based structure called Z-Table for time-efficient candidate generation and support count, and Z-Arrangement, a structure to efficiently store event intervals in temporal patterns for efficient memory consumption. Although using efficient data structures, Z-Miner neither employs the transitivity property of temporal relations nor mutual information for pruning. Thus, Z-Miner is less efficient than our exact and approximate GTPM in both runtimes and memory usage, and does not scale to large datasets with many sequences and many time series (see Section 6). Our GTPM algorithm improves on these methods by: (1) using efficient data structures and applying pruning techniques based on the Apriori principle and the transitivity property of temporal relations to enable fast mining, (2) the approximate GTPM can handle datasets with many time series and sequences, and (3), providing an end-to-end GTPMfTS process to mine temporal patterns directly from time series, a feature that is not supported by the baselines.

Rare pattern mining: Finding rare patterns that occur infrequently in a given database has received some attention in recent years. Techniques to find rare patterns in time series, often called rare motifs, are proposed in [15], [23], [24]. However, since time series motifs are the repeated subsequences of the time series, rare motif discovery techniques cannot deal with temporal events, and thus, are insufficient for rare temporal pattern mining. A related approach concerns rare association rules [6], [7], [8], [9], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34] that find rare associations between items in the database. However, all the mentioned work can only discover rare association rules built among itemsets, and cannot deal with temporal events and the complex temporal relations between them. Another research direction studies rare sequential patterns [10], [11], [12], [35], [36], [37]. However, rare sequential patterns only consider sequential occurrence between events, and therefore, cannot model other complex relations such as overlapping or containing between temporal events. To the best of our knowledge, there is currently no existing work that studies rare temporal pattern mining which mines rare occurrences of temporal patterns in a time series database.

Using correlations in TPM: Different correlation measures such as expected support [38], all-confidence [39], and mutual information (MI) [40], [41], [42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52] have been used to optimize the pattern mining process. However, these only support sequential patterns. To the best of our knowledge, our proposed approximate GTPM is the first that uses MI to optimize TPM.

3 Preliminaries

In this section, we introduce the notations and the main concepts that will be used throughout the paper.

3.1 Temporal Event of Time Series

Definition 3.1 (Time series) A *time series* $X = x_1, x_2, ..., x_n$ is a sequence of data values that measure the same phenomenon during an observation time period, and are chronologically ordered.

Definition 3.2 (Symbolic time series) A *symbolic time series* X_S of a time series X encodes the raw values of X into a sequence of symbols. The finite set of permitted symbols used to encode X is called the *symbol alphabet* Σ_X of X.

The symbolic time series X_S is obtained using a mapping function f: $X \rightarrow \Sigma_X$ that maps each value $x_i \in X$ to a symbol $\omega \in \Sigma_X$. For example, let X = 1.61, 1.21, 0.41, 0.0 be a time series representing the energy usage of an electrical device. Using the symbol alphabet Σ_X = {On, Off}, where On represents that the device is on and operating (e.g., $x_i \geq 0.5$), and Off that the device is off ($x_i < 0.5$), the symbolic representation of X is: X_S = On, On, Off, Off. The mapping function f can be defined using existing time series representation techniques such as SAX [53].

Definition 3.3 (Symbolic database) Given a set of time series $\mathcal{X} = \{X_1, ..., X_n\}$, the set of symbolic representations of the time series in $\mathcal{X}$ forms a *symbolic database* $\mathcal{D}_{\text{SYB}}$.

An example of the symbolic database $\mathcal{D}_{\text{SYB}}$ is shown in Table 1. There are 4 time series representing the energy usage of 4 electrical appliances: {Stove, Toaster, Clothes Washer, Iron}. For brevity, we name the appliances respectively as {S, T, W, I}. All appliances have the same alphabet Σ = {On, Off}.

TABLE 1: A Symbolic Database $\mathcal{D}_{\text{SYB}}$

Time	10:00	10:05	10:10	10:15	10:20	10:25	10:30	10:35	10:40	10:45	10:50	10:55	11:00	11:05	11:10	11:15	11:20	11:25	11:30	11:35	11:40	11:45	11:50	11:55	12:00	12:05	12:10	12:15	12:20	12:25	12:30	12:35	12:40	12:45	12:50	12:55
S	On	On	On	On	Off	Off	Off	On	On	Off	Off	Off	Off	Off	Off	On	On	On	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On	On	On	On	On	On
T	Off	Off	Off	Off	Off	Off	Off	On	On	Off	Off	On	On	Off	Off	On	On	On	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On	On	On	On	On	On
W	On	On	On	On	On	On	On	On	On	Off	Off	Off	Off	On	On	On	On	On	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	On	On	On	On	On	On	On	On
I	Off	Off	Off	Off	Off	Off	On	On	On	Off	Off	Off	On	On	Off	Off	On	On	Off	Off	Off	Off	Off	Off	Off	Off	Off	On	On	Off	Off	Off	Off	Off	On	On

TABLE 2: Temporal Relations between Events

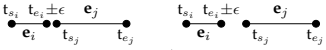
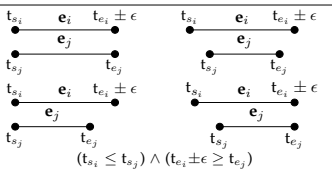
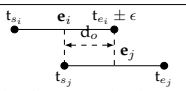
Follows: $E_{i_{\triangleright e_i}} \rightarrow E_{j_{\triangleright e_j}}$	 $t_{e_i} \pm \epsilon \leq t_{s_j}$
Contains: $E_{i_{\triangleright e_i}} \succcurlyeq E_{j_{\triangleright e_j}}$	 $(t_{s_i} \leq t_{s_j}) \wedge (t_{e_i} \pm \epsilon \geq t_{e_j})$
Overlaps: $E_{i_{\triangleright e_i}} \between E_{j_{\triangleright e_j}}$	 $(t_{s_i} < t_{s_j}) \wedge (t_{e_i} \pm \epsilon < t_{e_j}) \wedge (t_{e_i} - t_{s_j} \geq d_o \pm \epsilon)$

TABLE 3: A Temporal Sequence Database $\mathcal{D}_{\text{SEQ}}$

ID	Temporal sequences
1	(SOn,[10:00,10:15]), (TOff,[10:00,10:35]), (WOn,[10:00,10:40]), (IOff,[10:00,10:30]), (SOff,[10:15,10:35]), (IOn,[10:30,10:40]), (SOn, [10:35,10:40]), (TOn,[10:35,10:40])
2	(SOff,[10:45,11:15]), (TOff,[10:45,10:55]), (WOff,[10:45,11:05]), (IOff,[10:45,11:00]), (TOn,[10:55,11:00]), (TOff,[11:00,11:15]), (IOn,[11:00,11:05]), (WOn,[11:05,11:25]), (IOff,[11:05,11:20]), (SOn,[11:15,11:25]), (TOn,[11:15,11:25]), (IOn,[11:20,11:25])
3	(SOff,[11:30,12:10]), (TOff,[11:30,12:10]), (WOff,[11:30,12:10]), (IOff,[11:30,12:10])
4	(SOn,[12:15,12:55]), (TOn,[12:15,12:55]), (WOn,[12:15,12:55]), (IOn,[12:15,12:20]), IOff,[12:20,12:50]), (IOn,[12:50,12:55])

Definition 3.4 (Temporal event in a symbolic time series) A *temporal event* E in a symbolic time series X_S is a tuple $E = (\omega, T)$ where $\omega \in \Sigma_X$ is a symbol, and $T = \{[t_{s_i}, t_{e_i}]\}$ is the set of time intervals during which X_S is associated with the symbol ω.

Given a time series X, a temporal event is created by first converting X into symbolic time series X_S, and then combining identical consecutive symbols in X_S into one single time interval. For example, consider the symbolic representation of S in Table 1. By combining its consecutive On symbols, we form the temporal event *"Stove is On"* as: (SOn, {[10:00, 10:15], [10:35, 10:40], [11:15, 11:25], [12:15, 12:55]}).

Definition 3.5 (Instance of a temporal event) Let $E = (\omega, T)$ be a temporal event, and $[t_{s_i}, t_{e_i}] \in T$ be a time interval. The tuple $e = (\omega, [t_{s_i}, t_{e_i}])$ is called an *instance* of the event E, representing a single occurrence of E during $[t_{s_i}, t_{e_i}]$. We use the notation $E_{\triangleright e}$ to say that event E has an instance e.

3.2 Relations between Temporal Events

We adopt the popular Allen's relations model [54] and define three basic temporal relations between events. Furthermore, to avoid the exact time mapping problem in Allen's relations, we adopt the *buffer* idea from [5], adding a tolerance *buffer* ϵ to the relation's endpoints. However, we change the way ϵ is used in [5] to ensure the relations are *mutually exclusive* (proof is in the electronic appendix).

Consider two temporal events E_i and E_j, and their corresponding instances, $e_i = (\omega_i, [t_{s_i}, t_{e_i}])$ and $e_j = (\omega_j, [t_{s_j}, t_{e_j}])$. Let ϵ be a non-negative number ($\epsilon \geq 0$) representing the buffer size. The following relations can be defined between E_i and E_j through e_i and e_j.

Definition 3.6 (Follows) E_i and E_j form a *Follows* relation through e_i and e_j, denoted as Follows($E_{i_{\triangleright e_i}}$,$E_{j_{\triangleright e_j}}$) or $E_{i_{\triangleright e_j}} \rightarrow E_{j_{\triangleright e_j}}$, iff $t_{e_i} \pm \epsilon \leq t_{s_j}$.

Definition 3.7 (Contains) E_i and E_j form a *Contains* relation through e_i and e_j, denoted as Contains($E_{i_{\triangleright e_i}}$, $E_{j_{\triangleright e_j}}$) or $E_{i_{\triangleright e_j}} \succcurlyeq E_{j_{\triangleright e_j}}$, iff $(t_{s_i} \leq t_{s_j}) \wedge (t_{e_i} \pm \epsilon \geq t_{e_j})$.

Definition 3.8 (Overlaps) E_i and E_j form an *Overlaps* relation through e_i and e_j, denoted as Overlaps($E_{i_{\triangleright e_i}}$, $E_{j_{\triangleright e_j}}$) or $E_{i_{\triangleright e_i}} \between E_{j_{\triangleright e_j}}$, iff $(t_{s_i} < t_{s_j}) \wedge (t_{e_i} \pm \epsilon < t_{e_j}) \wedge (t_{e_i} - t_{s_j} \geq d_o \pm \epsilon)$, where d_o is the minimal overlapping duration between two event instances, and $0 \leq \epsilon \ll d_o$.

The *Follows* relation represents sequential occurrences of one event after another. For example, $E_{i_{\triangleright e_i}}$ is followed by $E_{j_{\triangleright e_j}}$ if the end time t_{e_i} of e_i occurs before the start time t_{s_j} of e_j. Here, the buffer ϵ is used as a tolerance, i.e., the *Follows* relation between $E_{i_{\triangleright e_i}}$ and $E_{j_{\triangleright e_j}}$ holds if $(t_{e_i} + \epsilon)$ or $(t_{e_i} - \epsilon)$ occurs before t_{s_j}. On the other hand, in a *Contains* relation, one event occurs entirely within the timespan of another event. Finally, in an *Overlaps* relation, the timespans of the two occurrences overlap each other. Table 2 illustrates the three temporal relations and their conditions.

3.3 Temporal Pattern

Definition 3.9 (Temporal sequence) A list of n event instances S=<e_1, ..., e_i, ..., e_n> forms a *temporal sequence* if the instances are chronologically ordered by their start times. Moreover, S has size n, denoted as $|S| = n$.

Definition 3.10 (Temporal sequence database) A set of temporal sequences forms a *temporal sequence database* $\mathcal{D}_{\text{SEQ}}$ where each row i contains a temporal sequence S_i.

Table 3 shows the temporal sequence database $\mathcal{D}_{\text{SEQ}}$, created from the symbolic database $\mathcal{D}_{\text{SYB}}$ in Table 1.

Definition 3.11 (Temporal pattern) Let $\Re$={Follows, Contains, Overlaps} be the set of temporal relations. A *temporal pattern* P=<(r_{12}, E_1, E_2),...,$(r_{(n-1)(n)}, E_{n-1}, E_n)$> is a list of triples ($r_{ij}$,$E_i$,$E_j$), each representing a relation $r_{ij} \in \Re$ between two events E_i and E_j.

Note that the relation r_{ij} in each triple is formed using the specific instances of E_i and E_j. A temporal pattern that has n events is called an n-event pattern. We use $E_i \in P$ to denote that the event E_i occurs in P, and $P_1 \subseteq P$ to say that a pattern P_1 is a sub-pattern of P.

Definition 3.12 (Temporal sequence supports a pattern) Let S=<e_1,...,e_i,...,e_n> be a temporal sequence. We say that S *supports* a temporal pattern P, denoted as $P \in S$, iff $|S| \geq 2 \wedge \forall (r_{ij}, E_i, E_j) \in P$, $\exists (e_l, e_m) \in S$ such that r_{ij} holds between $E_{i_{\triangleright e_l}}$ and $E_{j_{\triangleright e_m}}$.

If P is supported by S, P can be written as P=<$(r_{12}$, $E_{1_{\triangleright e_1}}$, $E_{2_{\triangleright e_2}}$), ..., ($r_{(n-1)(n)}$,$E_{n-1_{\triangleright e_{n-1}}}$, $E_{n_{\triangleright e_n}}$)>, where the

relation between two events in each triple is expressed using the event instances.

In Fig. 1, consider the sequence S =<e_1=(HighCO2, [6:00, 10:00]), e_2=(BoilerOn, [7:00, 8:00]), e_3=(LowCO2, [13:00, 15:00])> representing the order of CO2 intensity and boiler events. Here, S supports a 3-event pattern P=<(Contains, HighCO2$_{\triangleright e_1}$, BoilerOn$_{\triangleright e_2}$), (Follows, HighCO2$_{\triangleright e_1}$, LowCO2$_{\triangleright e_3}$), (Follows, BoilerOn$_{\triangleright e_2}$, LowCO2$_{\triangleright e_3}$)>.

Maximal duration constraint: Let $P \in S$ be a temporal pattern supported by the sequence S. The duration between the start time of the instance e_1, and the end time of the instance e_n in S must not exceed the predefined maximal time duration $t_{\max}$: $t_{e_n} - t_{s_1} \leq t_{\max}$.

The maximal duration constraint guarantees that the relation between any two events is temporally valid. This enables the pruning of invalid patterns. For example, under this constraint, a *Follows* relation between a *"Washer On"* event and a *"Dryer On"* event in Table 3 happening one year apart should be considered invalid.

3.4 Frequency and Likelihood Measures

Given a temporal sequence database $\mathcal{D}_{\text{SEQ}}$, we want to find patterns that occur at certain frequency in $\mathcal{D}_{\text{SEQ}}$. We use *support* and *confidence* [55] to measure the frequency and likelihood of a pattern.

Definition 3.13 (Support of a temporal event) The *support* of a temporal event E in $\mathcal{D}_{\text{SEQ}}$ is the number of sequences $S \in \mathcal{D}_{\text{SEQ}}$ containing at least one instance e of E.

$$supp(E) = |\{S \in \mathcal{D}_{\text{SEQ}} \text{ s.t. } \exists e \in S : E_{\triangleright e}\}| \quad (1)$$

The *relative support* of E is the fraction between $supp(E)$ and the size of $\mathcal{D}_{\text{SEQ}}$:

$$rel\text{-}supp(E) = supp(E)/|\mathcal{D}_{\text{SEQ}}| \quad (2)$$

Similarly, the support of a group of events $(E_1, ..., E_n)$, denoted as $supp(E_1, ..., E_n)$, is the number of sequences $S \in \mathcal{D}_{\text{SEQ}}$ which contain at least one instance $(e_1, ..., e_n)$ of the event group.

Definition 3.14 (Support of a temporal pattern) The *support* of a pattern P is the number of sequences $S \in \mathcal{D}_{\text{SEQ}}$ that support P.

$$supp(P) = |\{S \in \mathcal{D}_{\text{SEQ}} \text{ s.t. } P \in S\}| \quad (3)$$

The *relative support* of P in $\mathcal{D}_{\text{SEQ}}$ is the fraction

$$rel\text{-}supp(P) = supp(P)/|\mathcal{D}_{\text{SEQ}}| \quad (4)$$

Definition 3.15 (Confidence of an event pair) The *confidence* of an event pair (E_i, E_j) in $\mathcal{D}_{\text{SEQ}}$ is the fraction between $supp(E_i, E_j)$ and the support of its most frequent event:

$$conf(E_i, E_j) = \frac{supp(E_i, E_j)}{\max\{supp(E_i), supp(E_j)\}} \quad (5)$$

Definition 3.16 (Confidence of a temporal pattern) The *confidence* of a temporal pattern P in $\mathcal{D}_{\text{SEQ}}$ is the fraction between $supp(P)$ and the support of its most frequent event:

$$conf(P) = \frac{supp(P)}{\max_{1 \leq k \leq |P|}\{supp(E_k)\}} \quad (6)$$

where $E_k \in P$ is a temporal event. Since the denominator in Eq. (6) is the maximum support of the events in P, the confidence computed in Eq. (6) is the *minimum confidence* of a pattern P in $\mathcal{D}_{\text{SEQ}}$, which is also called the *all-confidence* as in [55]. Note that unlike association rules, temporal patterns do not have antecedents and consequents. Instead, they represent pair-wise temporal relations between events based on their temporal occurrences. Thus, while the *support* and *relative support* of event(s)/ pattern(s) defined in Eqs. (1) − (4) follow the same intuition as the traditional support concept, indicating how frequently an event/ pattern occurs in a given database, the *confidence* computed in Eqs. (5) − (6) instead represents the minimum likelihood of an event pair/ pattern, knowing the likelihood of its most frequent event.

Frequent temporal patterns vs. Rare temporal patterns: Consider a temporal pattern P in a temporal sequence database $\mathcal{D}_{\text{SEQ}}$ with the support $\sigma = supp(P)$ and the confidence $\delta = conf(P)$. Pattern P is considered to be *frequent* in $\mathcal{D}_{\text{SEQ}}$ if both support σ and confidence δ are high, representing the presence of pattern P in a large fraction of sequences in the database. In contrast, pattern P is considered to be *rare* in $\mathcal{D}_{\text{SEQ}}$ if the support σ is low and the confidence δ is high, indicating a type of pattern that occurs only in a small fraction of sequences but with high likelihood, given the occurrence evidence of the involved events.

Problem Definition: Generalized Temporal Pattern Mining. Given a set of univariate time series $\mathcal{X} = \{X_1, ..., X_n\}$, let $\mathcal{D}_{\text{SEQ}}$ be the temporal sequence database obtained from $\mathcal{X}$, and $\sigma_{\min}$, $\sigma_{\max}$ and δ be the minimum support, maximum support and minimum confidence thresholds, respectively. The Generalized Temporal Pattern Mining from Time Series (GTPMfTS) problem aims to find all temporal patterns P in $\mathcal{D}_{\text{SEQ}}$ such that P satisfies the support and confidence constraints, i.e., $\sigma_{\min} \leq supp(P) \leq \sigma_{\max} \wedge conf(P) \geq \delta$.

Using the three constraints $\sigma_{\min}$, $\sigma_{\max}$ and δ, GTPMfTS can mine frequent temporal patterns in $\mathcal{D}_{\text{SEQ}}$ by setting $\sigma_{\max} = \infty$, and assigning $\sigma_{\min}$ and δ to high threshold values. In contrast, to mine rare temporal patterns, GTPMfTS will assign low threshold values to $\sigma_{\min}$ and $\sigma_{\max}$, constraining on a low occurrence frequency, and a high value to δ, constraining on a high likelihood of the patterns.

4 Generalized Temporal Pattern Mining

In this section, we present the Generalized Temporal Pattern Mining (GTPM) algorithm to mine both frequent and rare temporal patterns from time series. Fig. 2 gives an overview of the GTPMfTS process which consists of two phases. The first phase, *Data Transformation*, converts a set of time series $\mathcal{X}$ into a symbolic database $\mathcal{D}_{\text{SYB}}$, and then converts $\mathcal{D}_{\text{SYB}}$ into a temporal sequence database $\mathcal{D}_{\text{SEQ}}$. The second phase, *Generalized Temporal Pattern Mining (GTPM)*, mines both frequent and rare temporal patterns, and consists of three steps: (1) *Mining Single Events*, (2) *Mining 2-Event Patterns*, and (3) *Mining k-Event Patterns* (k>2). The final output is a set of all temporal patterns in $\mathcal{D}_{\text{SEQ}}$ that satisfy the minimum support, maximum support and minimum confidence constraints.

4.1 Data Transformation

4.1.1 Symbolic Time Series Representation

Given a set of time series $\mathcal{X}$, the symbolic representation of each time series $X \in \mathcal{X}$ is obtained by using a mapping function as in Def. 3.2.

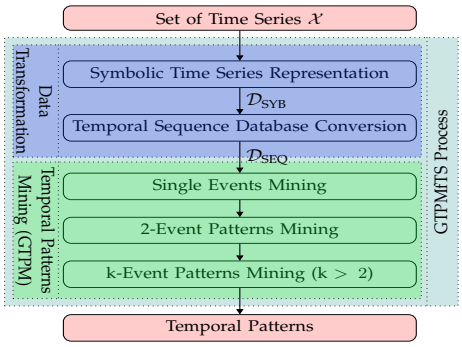
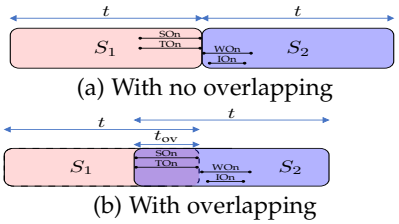


Fig. 2: The GTPMfTS process



(a) With no overlapping

(b) With overlapping

Fig. 3: Splitting strategy

4.1.2 Temporal Sequence Database Conversion

To convert $\mathcal{D}_{SYB}$ to $\mathcal{D}_{SEQ}$, a straightforward approach is to split the symbolic series in $\mathcal{D}_{SYB}$ into equal-length sequences, each belongs to a row in $\mathcal{D}_{SEQ}$. For example, if each symbolic series in Table 1 is split into 4 sequences, then each sequence will last for 40 minutes. The first sequence S_1 of $\mathcal{D}_{SEQ}$ therefore contains temporal events of S, T, W, and I from 10:00 to 10:40. The second sequence S_2 contains events from 10:45 to 11:25, and similarly for S_3 and S_4.

However, the splitting can lead to a potential loss of temporal patterns. The loss happens when a *splitting point* accidentally divides a temporal pattern into different sub-patterns, and places these into separate sequences. We explain this situation in Fig. 3a. Consider 2 sequences S_1 and S_2, each of length t. Here, the splitting point divides a pattern of 4 events, {SOn, TOn, WOn, IOn}, into two sub-patterns, in which SOn and TOn are placed in S_1, and WOn and IOn in S_2. This results in the loss of this 4-event pattern which can be identified only when all 4 events are in the same sequence.

To prevent such a loss, we propose a *splitting strategy* using overlapping sequences. Specifically, two consecutive sequences are overlapped by a duration t_{ov}: $0 \leq t_{ov} \leq t_{max}$, where t_{max} is the *maximal duration* of a temporal pattern. The value of t_{ov} decides how large the overlap between S_i and S_{i+1} is: $t_{ov} = 0$ results in no overlap, i.e., no redundancy, but with a potential loss of patterns, while $t_{ov} = t_{max}$ creates large overlaps between sequences, i.e., high redundancy, but all patterns are preserved. As illustrated in Fig. 3b, the overlapping between S_1 and S_2 keeps the 4 events together in the same sequence S_2, and thus helps preserve the pattern.

4.2 Generalized Temporal Pattern Mining

We now present the GTPM algorithm to mine temporal patterns, both frequent and rare, from $\mathcal{D}_{SEQ}$. We note that for frequent patterns, only two constraints σ_{min} and δ are used, whereas with rare patterns, all three constraints σ_{min}, σ_{max}, and δ are used. In the following when presenting the GTPM algorithm, the discussion applies to both frequent and rare patterns, with the implication that σ_{max} is set to ∞ when mining frequent patterns.

The main novelties of GTPM are: a) the use of efficient data structures, i.e., the *Hierarchical Lookup Hash (HLH)* structure [56], and b) the proposal of two groups of pruning techniques based on the Apriori principle and the temporal transitivity property of temporal events. Particularly, instead of using the Hierarchical Pattern Graph as in [13], we use the Hierarchical Lookup Hash data structure to enable faster retrieval of events and patterns during the mining process. Algorithm 1 provides the pseudo-code of our GTPM algorithm.

Algorithm 1: Generalized Temporal Pattern Mining

Input: Temporal sequence database $\mathcal{D}_{SEQ}$, minimum support threshold σ_{min}, maximum support threshold σ_{max}, confidence threshold δ
Output: The set of temporal patterns P satisfying σ_{min}, σ_{max}, δ

```
//Mining single events
foreach event E_i ∈ D_SEQ do
    Compute supp(E_i);
    if supp(E_i) ≥ σ_min then
        Insert E_i to 1Freq;
//Mining 2-event patterns
EventPairs ← Cartesian(1Freq,1Freq);
FrequentPairs ← ∅;
foreach (E_i, E_j) in EventPairs do
    Compute supp(E_i, E_j);
    if supp(E_i, E_j) ≥ σ_min then
        FrequentPairs ← Apply_Lemma4(E_i, E_j);
foreach (E_i, E_j) in FrequentPairs do
    Retrieve event instances;
    Check temporal relations against σ_min, σ_max, δ;
//Mining k-event patterns
Filtered1Freq ← Transitivity_Filtering(1Freq);
kEvents ← Cartesian(Filtered1Freq,(k-1)Freq);
FrequentkEvents ← Apriori_Filtering(kEvents);
foreach kEvents in FrequentkEvents do
    Retrieve relations;
    Iteratively check relations against σ_min, σ_max, δ;
```

4.3 Mining Single Events

Hierarchical lookup hash structure HLH_1: We use the hierarchical lookup hash structure HLH_1, illustrated in Fig. 4 to store single events. HLH_1 is a hierarchical data structure that consists of two hash tables: the *single event hash table* EH, and the *event sequence hash table* SH. Each hash table has a list of <key, value> pairs. In EH, the key is the event symbol $\omega \in \Sigma_X$ representing the event E_i, and the value is the set of sequences $< S_i, ..., S_k >$ (arranged in an increasing order) that contain E_i. In SH, the key is taken from the value component of EH, i.e., the set of sequences, while the value stores event instances of E_i that occur in the corresponding sequence in $\mathcal{D}_{SEQ}$. The HLH_1 structure

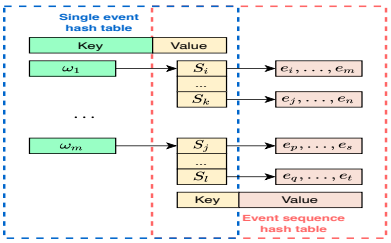


Fig. 4: The HLH_1 structure

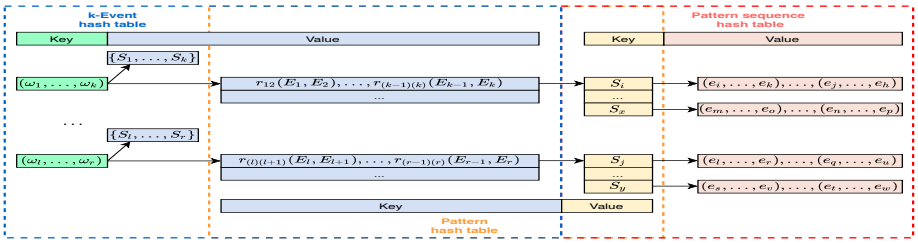


Fig. 5: The $HLH_k (k \geq 2)$ structure

enables faster retrieval of event sequences and instances when mining k-event patterns.

Mining Single Events: The first step in GTPM is to find single events that satisfy the minimum support constraint $\sigma_{\min}$ (Alg. 1, lines 1-4). To do that, GTPM scans $\mathcal{D}_{\text{SEQ}}$ to compute the support of each event E_i, and checks whether $supp(E_i) \geq \sigma_{\min}$. Note that for single events, we do not consider the constraints on the confidence δ, since confidence of single events is always 1, and on maximum support $\sigma_{\max}$ because of the following lemma.

Lemma 1. *Let P be a temporal pattern and E_i be a single event such that $E_i \in P$. Then $supp(P) \leq supp(E_i)$.*

Proof. *Detailed proofs of all lemmas, theorems, and complexities in this article can be found in the electronic appendix.*

From Lemma 1, a single event E_i whose support $supp(E_i) > \sigma_{\max}$ can form a pattern P that has $supp(P) \leq \sigma_{\max}$. Thus, the constraint on $\sigma_{\max}$ is not considered for single events to avoid the loss of potential temporal patterns.

We provide a running example using data in Table 3, with $\sigma_{\min} = 0.7$, $\sigma_{\max} = 0.9$, and $\delta = 0.7$. The data structure HLH_1, shown in Fig. 6, stores 7 single events satisfying $\sigma_{\min}$ constraint. The event WOff does not satisfy $\sigma_{\min}$ (only appears in sequences 2 and 4), and is thus omitted.

Complexity: The complexity of finding single events is $O(m \cdot |\mathcal{D}_{\text{SEQ}}|)$, where m is the number of distinct events.

4.4 Mining 2-event Patterns

Search space of GTPM: The next step in GTPM is to mine 2-event patterns. A straightforward approach would be to enumerate all possible event pairs, and check whether each pair can form patterns that satisfy the support and confidence constraints. However, this *naive* approach is very expensive. Not only does it need to repeatedly scan $\mathcal{D}_{\text{SEQ}}$ to check each combination of events, the complex relations between events also add an extra exponential factor 3^{h^2} to the m^h number of possible candidates, creating a very large search space that makes the approach infeasible.

Lemma 2. *Let m be the number of distinct events in $\mathcal{D}_{SEQ}$, and h be the longest length of a temporal pattern. The total number of temporal patterns is $O(m^h 3^{h^2})$.*

Lemma 2 shows the driving factors of GTPM's exponential search space (proof in the electronic appendix): the number of events (m), the max pattern length (h), and the number of temporal relations (3). A dataset of just a few hundred events can create a very large search space with billions of candidate patterns. The optimizations and approximation proposed in the following sections will help mitigate this problem.

Hierarchical lookup hash structure HLH_k: We maintain k-event groups and patterns found by GTPM using the

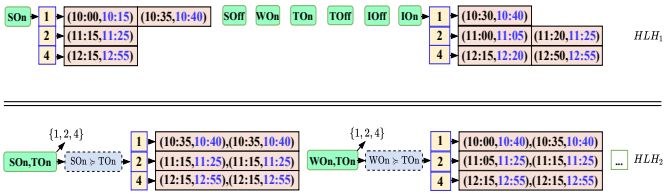


Fig. 6: A hierarchical lookup hash tables for the running example

HLH_k $(k \geq 2)$ data structure, illustrated in Fig. 5. HLH_k contains three hash tables, each has a list of <key, value> pairs: the *k-event hash table* EH_k, the *pattern hash table* PH_k, and the *pattern sequence hash table* SH_k. For each *<key, value>* pair of EH_k, *key* is the list of symbols $(\omega_1 ..., \omega_k)$ representing the k-event group $(E_1, ..., E_k)$, and *value* is an *object* structure which consists of two components: (1) the list of sequences $< S_i, ..., S_k >$ (arranged in increasing order) where $(E_1, ..., E_k)$ occurs, and (2), a list of k-event temporal patterns $P = \{(r_{12}, E_1, E_2), ..., (r_{(k-1)(k)}, E_{k-1}, E_k)\}$ created from the k-event group $(E_1, ..., E_k)$. In PH_k, the *key* takes the *value* component of EH_k, i.e. the k-event pattern P, while the *value* is the list of sequences that support P. In SH_k, the *key* takes the *value* component of PH_k, i.e., the list of sequences that support P, while the *value* is the list of event instances from which the temporal relations in P are formed. The HLH_k hash structure helps speed up the mining of k-event groups through the use of sequences in EH_k, and enables faster search for temporal relations between k events using the information in PH_k and SH_k.

Two-steps filtering approach to mine 2-event patterns: Given the huge set of pattern candidates stated in Lemma 2, it is expensive to check their support and confidence. We propose a *filtering approach* to reduce the unnecessary candidate checking. Specifically, the mining process is divided into two steps: (1) it first finds k-event groups that satisfy the minimum support and confidence constraints using $\sigma_{\min}$ and δ, (2) it then generates temporal patterns only from those k-event groups. The correctness of this filtering approach is based on the Apriori-inspired lemmas below.

Lemma 3. *Let P be a 2-event pattern formed by an event pair (E_i, E_j). Then, $supp(P) \leq supp(E_i, E_j)$.*

From Lemma 3, the support of a pattern is at most the support of its events. Thus, infrequent event pairs (those do not satisfy minimum support) cannot form frequent patterns and thereby, can be safely pruned.

Lemma 4. *Let (E_i, E_j) be a pair of events forming a 2-event pattern P. Then $conf(P) \leq conf(E_i, E_j)$.*

From Lemma 4, the confidence of a pattern P is always

at most the confidence of its events. Thus, a low-confidence event pair cannot form any high-confidence patterns and therefore, can be safely pruned. We note that the Apriori principle has already been used in other work, e.g., [3], [5], for mining optimization. However, they only apply this principle to the support (Lemma 3), while we further extend it to the confidence (Lemma 4). Applying Lemmas 3 and 4 to the event filtering step will remove infrequent or low-confidence event pairs, reducing the candidate patterns of GTPM. Furthermore, we do not consider the constraint on $\sigma_{\max}$ in this filtering step to avoid the loss of 2-event patterns, as event pairs that do not satisfy the $\sigma_{\max}$ constraint can still form 2-event patterns satisfying $\sigma_{\max}$ (Lemma 3).

Step 2.1. Mining event pairs considering $\sigma_{\min}$ and δ: This step finds event pairs in $\mathcal{D}_{\text{SEQ}}$ satisfying $\sigma_{\min}$ and δ, using the set *1Freq* found in HLH_1 (Alg. 1, lines 5-10). First, GTPM generates all possible event pairs by calculating the Cartesian product *1Freq* $\times$ *1Freq*. Next, for each pair (E_i, E_j), the set $\mathcal{S}_{ij}$ (representing the set of sequences where both events occur) is computed by taking the *intersection* between the set of sequences $\mathcal{S}_i$ of E_i and the set of sequences $\mathcal{S}_j$ of E_j in HLH_1. Finally, we compute the support $supp(E_i, E_j)$ using $\mathcal{S}_{ij}$, and compare against $\sigma_{\min}$. If $supp(E_i, E_j) \geq \sigma_{\min}$, (E_i, E_j) has high enough support. Next, (E_i, E_j) is further filtered using Lemma 4: (E_i, E_j) is selected only if its confidence is at least δ. After this step, only event pairs satisfying $\sigma_{\min}$ and δ are kept in EH_2 of HLH_2.

Step 2.2. Mining 2-event patterns: This step mines 2-event patterns from the event pairs found in step 2.1 (Alg. 1, lines 11-13), considering three constraints $\sigma_{\min}$, $\sigma_{\max}$, and δ. For each event pair (E_i, E_j), we use the set of sequences $\mathcal{S}_{ij}$ to check the temporal relations between E_i and E_j. Specifically, for each sequence $S \in \mathcal{S}_{ij}$, the pairs of event instances (e_i, e_j) are extracted, and the relations between them are verified. The support and confidence of each relation $r(E_{i_{\triangleright e_i}}, E_{j_{\triangleright e_j}})$ are computed and compared against $\sigma_{\min}$, and δ thresholds, after which only relations satisfying the two constraints are selected and stored in PH_2, while their event instances are stored in SH_2. Examples of the relations in HLH_2 can be seen in Fig. 6, e.g., event pair (SOn, TOn). We also emphasize that HLH_2 only stores patterns that satisfy the two constraints $\sigma_{\min}$, and δ, thus, patterns in PH_2 are frequent temporal patterns. To mine rare temporal patterns from HLH_2, we take a further step by iterating through every 2-event pattern P in PH_2, and checking the satisfaction of P against the constraint $\sigma_{\max}$.

Complexity: Let m be the number of single events in HLH_1, and i be the average number of event instances of each event. The complexity of 2-event pattern mining is $O(m^2 i^2 \ |\mathcal{D}_{\text{SEQ}}|^2)$.

4.5 Mining k-event Patterns

Mining k-event patterns ($k \geq 3$) follows a similar process as 2-event patterns, with additional prunings based on the transitivity property of temporal relations.

Step 3.1. Mining k-event combinations considering $\sigma_{\min}$ and δ: This step finds k-event combinations that satisfy the minimum support and confidence constraints (Alg. 1, lines 14-16).

Let *(k-1)Freq* be the set of (k-1)-event combinations found in HLH_{k-1}, and *1Freq* be the set of single events in HLH_1. To generate all k-event combinations, the typical process is to compute the Cartesian product: *(k-1)Freq* $\times$ *1Freq*. However, we observe that using *1Freq* to generate k-event combinations at HLH_k can create redundancy, since *1Freq* might contain events that when combined with *(k-1)Freq*, result in combinations that clearly cannot form any patterns satisfying the minimum support constraint. To illustrate this observation, consider the event IOn in HLH_1 in Fig. 6. Here, IOn is a frequent event, and thus, can be combined with frequent event pairs in HLH_2 such as (SOn, TOn) to create a 3-event combination (SOn, TOn, IOn). However, (SOn, TOn, IOn) cannot form any 3-event patterns whose support is greater than $\sigma_{\min}$, since IOn is not present in any frequent 2-event patterns in HLH_2. To reduce the redundancy, the combination (SOn, TOn, IOn) should not be created in the first place. We rely on the *transitivity property* of temporal relations to identify such event combinations.

Lemma 5. *Let $S =< e_1,..., e_{n-1} >$ be a temporal sequence that supports an (n-1)-event pattern $P =< (r_{12}, E_{1_{\triangleright e_1}}, E_{2_{\triangleright e_2}}),..., (r_{(n-2)(n-1)}, E_{n-2_{\triangleright e_{n-2}}}, E_{n-1_{\triangleright e_{n-1}}}) >$. Let e_n be a new event instance added to S to create the temporal sequence $S' =< e_1, ..., e_n >$.*

The set of temporal relations $\Re$ is transitive on S': $\forall e_i \in S'$, $i < n$, $\exists r \in \Re$ s.t. $r(E_{i_{\triangleright e_i}}, E_{n_{\triangleright e_n}})$ holds.

Lemma 5 says that given a temporal sequence S, a new event instance added to S will always form at least one temporal relation with existing instances in S. This is due to the temporal transitivity property, which can be used to prove the following lemma.

Lemma 6. *Let $N_{k-1} = (E_1, ..., E_{k-1})$ be a (k-1)-event combination and E_k be a single event, both satisfying the $\sigma_{\min}$ constraint. The combination $N_k = N_{k-1} \cup E_k$ can form k-event temporal patterns whose support is at least $\sigma_{\min}$ if $\forall E_i \in N_{k-1}$, $\exists r \in \Re$ s.t. $r(E_i, E_k)$ is a frequent temporal relation.*

From Lemma 6, only single events in HLH_1 that appear in HLH_{k-1} should be used to create k-event combinations. Using this result, a filtering on *1Freq* is performed before calculating the Cartesian product. Specifically, from the events in HLH_{k-1}, we extract distinct single events D_{k-1}, and *intersect* D_{k-1} with *1Freq* to remove redundant single events: *Filtered1Freq* = $D_{k-1} \cap$ *1Freq*. Next, the Cartesian product *(k-1)Freq* $\times$ *Filtered1Freq* is calculated to generate k-event combinations. Finally, we apply Lemmas 3 and 4 to select k-event combinations *kFreq* which upheld the $\sigma_{\min}$ and δ constraints. Similar to step 2.1, we do not consider $\sigma_{\max}$ when generating the k-event combination.

Step 3.2. Mining k-event patterns: This step mines k-event patterns that satisfy the three constraints of $\sigma_{\min}$, $\sigma_{\max}$, and δ (Alg. 1, lines 17-19). Unlike 2-event patterns, verifying the relations in a k-event combination ($k \geq 3$) is much more expensive, as it requires to compute the frequency of $\frac{1}{2}k(k-1)$ triples of temporal relations. To reduce the cost of relation checking, we propose an iterative verification method that relies on the *transitivity property* and the Apriori principle.

Lemma 7. *Let P and P' be two temporal patterns. If $P' \subseteq P$, then conf(P') $\geq$ conf(P).*

Lemma 8. *Let P and P' be two temporal patterns. If $P' \subseteq P$ and $\frac{supp(P')}{\max_{1\leq k\leq |P|}\{supp(E_k)\}}{}_{E_k\in P} \leq \delta$, then conf(P) $\leq \delta$.*

Lemma 7 says that, the confidence of a pattern P is always at most the confidence of its sub-patterns. Consequently, from Lemma 8, a temporal pattern P cannot be high-confidence if any of its sub-patterns are low-confidence.

Let $N_{k-1} = (E_1, ..., E_{k-1})$ be a (k-1)-event combination in HLH_{k-1}, $N_1 = (E_k)$ be an event in HLH_1, and $N_k = N_{k-1} \cup N_1 = (E_1, ..., E_k)$ be a k-event combination in HLH_k. To find k-event patterns for N_k, we first retrieve the set P_{k-1} containing (k-1)-event patterns of N_{k-1} by accessing the EH_{k-1} table. Each $p_{k-1} \in P_{k-1}$ is a list of $\frac{1}{2}(k-1)(k-2)$ triples: $\{(r_{12}, E_{1_{\triangleright e_1}}, E_{2_{\triangleright e_2}})$,...,$(r_{(k-2)(k-1)}, E_{k-2_{\triangleright e_{k-2}}}, E_{k-1_{\triangleright e_{k-1}}})\}$. We iteratively verify the possibility of p_{k-1} forming a k-event pattern with E_k that can satisfy the $\sigma_{\min}$ constraint as follows. We first check whether the triple $(r_{(k-1)k}, E_{k-1_{\triangleright e_{k-1}}}, E_{k_{\triangleright e_k}})$ satisfies the constraints of $\sigma_{\min}$, $\sigma_{\max}$, and δ by accessing the HLH_2 table. If the triple does not satisfy the minimum and maximum support constraints (using Lemmas 5 and 6), or the confidence constraint (using Lemmas 5, 7, and 8), the verifying process stops immediately for p_{k-1}. Otherwise, it continues on the triple $(r_{(k-2)k}, E_{k-2_{\triangleright e_{k-2}}}, E_{k_{\triangleright e_k}})$, until it reaches $(r_{1k}, E_{1_{\triangleright e_1}}, E_{k_{\triangleright e_k}})$.

We note that the transitivity property of temporal relations has been exploited in [16] to generate new relations. Instead, we use this property to prune unpromising candidates (Lemmas 5, 6, 7, 8).

Complexity: Let r be the average number of (k-1)-event patterns in HLH_{k-1}. The complexity of k-event pattern mining is $O(|1Freq| \cdot |(k\text{-}1)Freq| \cdot r \cdot k^2 \cdot |\mathcal{D}_{\text{SEQ}}|)$.

GTPM overall complexity: Throughout this section, we have seen that GTPM complexity depends on the size of the search space $(O(m^h 3^{h^2}))$ and the complexity of the mining process itself, i.e., $O(m \cdot |\mathcal{D}_{\text{SEQ}}|) + O(m^2 i^2 \ |\mathcal{D}_{\text{SEQ}}|^2) + O(|1Freq| \cdot |(k\text{-}1)Freq| \cdot r \cdot k^2 \cdot |\mathcal{D}_{\text{SEQ}}|)$. While the parameters m, h, i, r and k depend on the number of time series, others such as $|1Freq|$, $|(k\text{-}1)Freq|$ and $|\mathcal{D}_{\text{SEQ}}|$ also depend on the number of temporal sequences. Thus, given a dataset, GTPM complexity is driven by two main factors: the number of time series and the number of temporal sequences.

5 Approximate GTPM

5.1 Mutual Information of Symbolic Time Series

Let X_S and Y_S be the symbolic series representing the time series X and Y, respectively, and Σ_X, Σ_Y be their alphabets.

Definition 5.1 (Entropy) The *entropy* of X_S, denoted as $H(X_S)$, is defined as

$$H(X_S) = -\sum_{x\in\Sigma_X} p(x) \cdot \log p(x) \tag{7}$$

Intuitively, the entropy measures the amount of information or the inherent uncertainty in the possible outcomes of a random variable. The higher the $H(X_S)$, the more uncertain the outcome of X_S.

The conditional entropy $H(X_S|Y_S)$ quantifies the amount of information needed to describe the outcome of X_S, given the value of Y_S, and is defined as

$$H(X_S|Y_S) = -\sum_{x\in\Sigma_X}\sum_{y\in\Sigma_Y} p(x,y) \cdot \log \frac{p(x,y)}{p(y)} \tag{8}$$

Definition 5.2 (Mutual information) The *mutual information* (MI) of two symbolic series X_S and Y_S, denoted as $I(X_S; Y_S)$, is defined as

$$I(X_S; Y_S) = \sum_{x\in\Sigma_X}\sum_{y\in\Sigma_Y} p(x,y) \cdot \log \frac{p(x,y)}{p(x)\cdot p(y)} \tag{9}$$

The MI represents the reduction of uncertainty of one variable (e.g., X_S), given the knowledge of another variable (e.g., Y_S). The larger $I(X_S; Y_S)$, the more information is shared between X_S and Y_S, and thus, the less uncertainty about one variable given the other.

Since $0 \leq I(X_S; Y_S) \leq \min(H(X_S), H(Y_S))$ [57], MI has no upper bound. To scale the MI into the range $[0 - 1]$, we use normalized mutual information as defined below.

Definition 5.3 (Normalized mutual information) The *normalized mutual information* (NMI) of two symbolic time series X_S and Y_S, denoted as $\widetilde{I}(X_S; Y_S)$, is defined as

$$\widetilde{I}(X_S; Y_S) = \frac{I(X_S; Y_S)}{H(X_S)} = 1 - \frac{H(X_S|Y_S)}{H(X_S)} \tag{10}$$

$\widetilde{I}(X_S; Y_S)$ represents the reduction (in percentage) of the uncertainty of X_S due to knowing Y_S. Based on Eq. (10), a pair of variables (X_S, Y_S) holds a mutual dependency if $\widetilde{I}(X_S; Y_S) > 0$. Eq. (10) also shows that NMI is not symmetric, i.e., $\widetilde{I}(X_S; Y_S) \neq \widetilde{I}(Y_S; X_S)$.

5.2 Lower Bound of the Support of an Event Pair

Consider two symbolic series X_S and Y_S. Let X_1 be an event in X_S, Y_1 be an event in Y_S, and $\mathcal{D}_{\text{SYB}}$ and $\mathcal{D}_{\text{SEQ}}$ be the symbolic and the sequence databases created from X_S and Y_S, respectively. We first study the relationship between the support of (X_1, Y_1) in $\mathcal{D}_{\text{SYB}}$ and $\mathcal{D}_{\text{SEQ}}$.

Lemma 9. *Let $supp(X_1, Y_1)_{\mathcal{D}_{SYB}}$ and $supp(X_1, Y_1)_{\mathcal{D}_{SEQ}}$ be the support of (X_1, Y_1) in $\mathcal{D}_{SYB}$ and $\mathcal{D}_{SEQ}$, respectively. Then $supp(X_1, Y_1)_{\mathcal{D}_{SYB}} \leq supp(X_1, Y_1)_{\mathcal{D}_{SEQ}}$ holds.*

Proof. (Sketch - Detailed proof in the electronic appendix). Let n be the length of each symbolic time series in $\mathcal{D}_{\text{SYB}}$, and m be the length of each temporal sequence. The number of temporal sequences obtained in $\mathcal{D}_{\text{SEQ}}$ is: $\lceil \frac{n}{m} \rceil$.

The support of (X_1, Y_1) in $\mathcal{D}_{\text{SYB}}$ is computed as:

$$supp(X_1, Y_1)_{\mathcal{D}_{\text{SYB}}} = \frac{\sum_{i=1}^{\lceil \frac{n}{m} \rceil} \sum_{j=1}^{m} s_{ij}}{n} \tag{11}$$

where

$$s_{ij} = \begin{cases} 1, & \text{if } (X_1, Y_1) \text{ occurs in row j of the sequence } s_i \text{ in } \mathcal{D}_{\text{SYB}} \\ 0, & \text{otherwise} \end{cases}$$

Moreover, we have:

$$supp(X_1, Y_1)_{\mathcal{D}_{\text{SEQ}}} = \frac{\sum_{i=1}^{\lceil \frac{n}{m} \rceil} g_i}{n/m} = \frac{m \cdot \sum_{i=1}^{\lceil \frac{n}{m} \rceil} g_i}{n} \tag{12}$$

where

$$g_i = \begin{cases} 1, & \text{if } (X_1, Y_1) \text{ occurs in the sequence } g_i \text{ in } \mathcal{D}_{\text{SEQ}} \\ 0, & \text{otherwise} \end{cases}$$

We also get:

$$supp(X_1,Y_1)_{\mathcal{D}_{\text{SEQ}}} = \frac{m \cdot \sum_{i=1}^{\lceil \frac{n}{m} \rceil} g_i}{n} = \frac{\sum_{i=1}^{\lceil \frac{n}{m} \rceil} m \cdot g_i}{n} = \frac{\sum_{i=1}^{\lceil \frac{n}{m} \rceil} \left(\sum_{j=1}^{m} s_{ij} + \vartheta_i \right)}{n} \quad (13)$$

where s_{ij} is defined as in Eq. (11), and

$$\vartheta_i = \begin{cases} m - \sum_{j=1}^{m} s_{ij}, & \text{if } \sum_{j=1}^{m} s_{ij} \neq 0 \\ 0, & \text{otherwise} \end{cases}$$

From Eq. (13), we have:

$$supp(X_1,Y_1)_{\mathcal{D}_{\text{SEQ}}} = \frac{\sum_{i=1}^{\lceil \frac{n}{m} \rceil} \sum_{j=1}^{m} s_{ij}}{n} + \frac{\sum_{i=1}^{\lceil \frac{n}{m} \rceil} \vartheta_i}{n} = supp(X_1,Y_1)_{\mathcal{D}_{\text{SYB}}} + \vartheta \quad (14)$$

where $\vartheta = \frac{\sum_{i=1}^{\lceil \frac{n}{m} \rceil} \vartheta_i}{n}$ is the difference between the probabilities of (X_1, Y_1) in $\mathcal{D}_{\text{SEQ}}$ and $\mathcal{D}_{\text{SYB}}$.
From Eq. (14), we have:

$$supp(X_1,Y_1)_{\mathcal{D}_{\text{SYB}}} \leq supp(X_1,Y_1)_{\mathcal{D}_{\text{SEQ}}} \quad (15)$$

□

From Lemma 9, a frequent event pair in $\mathcal{D}_{\text{SYB}}$ is also frequent in $\mathcal{D}_{\text{SEQ}}$. We now investigate the relation between $\widetilde{I}(X_S;Y_S)$ in $\mathcal{D}_{\text{SYB}}$ and the support of (X_1, Y_1) in $\mathcal{D}_{\text{SEQ}}$.

Theorem 1. *(Lower bound of the support) Let $\mu_{\min}$ be the minimum mutual information threshold. If the NMI $\widetilde{I}(X_S;Y_S) \geq \mu_{\min}$, then the lower bound of the support of (X_1, Y_1) in $\mathcal{D}_{SEQ}$ is:*

$$supp(X_1,Y_1)_{\mathcal{D}_{SEQ}} \geq \lambda_2 \cdot e^{W\left(\frac{\log \lambda_1^{1-\mu_{\min}} \cdot ln2}{\lambda_2}\right)} \quad (16)$$

where λ_1 is the minimum support of $X_i \in X_S$, λ_2 is the support of $Y_1 \in Y_S$, and W is the Lambert function [58].

Proof. (Sketch - Detailed proof in the electronic appendix). From Eq. (10), we have:

$$\widetilde{I}(X_S;Y_S) = 1 - \frac{H(X_S|Y_S)}{H(X_S)} \geq \mu_{\min} \quad (17)$$

$$\Rightarrow \frac{H(X_S|Y_S)}{H(X_S)} = \frac{p(X_1,Y_1) \cdot \log p(X_1|Y_1)}{\sum_i p(X_i) \cdot \log p(X_i)} + \frac{\sum_{i \neq 1 \wedge j \neq 1} p(X_i,Y_j) \cdot \log \frac{p(X_i,Y_j)}{p(Y_j)}}{\sum_i p(X_i) \cdot \log p(X_i)} \leq 1 - \mu_{\min} \quad (18)$$

Let $\lambda_1 = p(X_k)$ such that $p(X_k) = \min\{p(X_i)\}, \forall i$, and $\lambda_2 = p(Y_1)$. We obtain:

$$\frac{H(X_S|Y_S)}{H(X_S)} \geq \frac{p(X_1,Y_1) \cdot \log \frac{p(X_1,Y_1)}{\lambda_2}}{\log \lambda_1} \quad (19)$$

From Eqs. (18), (19), the support lower bound of (X_1, Y_1) in $\mathcal{D}_{\text{SYB}}$ is derived as:

$$supp(X_1,Y_1)_{\mathcal{D}_{\text{SYB}}} \geq \lambda_2 \cdot e^{W\left(\frac{\log \lambda_1^{1-\mu_{\min}} \cdot \ln 2}{\lambda_2}\right)} \quad (20)$$

Since:

$$supp(X_1,Y_1)_{\mathcal{D}_{\text{SEQ}}} \geq supp(X_1,Y_1)_{\mathcal{D}_{\text{SYB}}} \quad (21)$$

It follows that:

$$supp(X_1,Y_1)_{\mathcal{D}_{\text{SEQ}}} \geq \lambda_2 \cdot e^{W\left(\frac{\log \lambda_1^{1-\mu_{\min}} \cdot \ln 2}{\lambda_2}\right)} \quad (22)$$

□

From Theorem 1, we can derive the minimum MI threshold $\mu_{\min}$ such that the support of (X_1, Y_1) is at least $\sigma_{\min}$.

Corollary 1.1. *The support of an event pair $(X_1, Y_1) \in (X_S, Y_S)$ in $\mathcal{D}_{SEQ}$ is at least $\sigma_{\min}$ if $\widetilde{I}(X_S;Y_S)$ is at least $\mu_{\min}$, where:*

$$\mu_{\min} \geq \begin{cases} 1 - \frac{\lambda_2}{e \cdot \ln 2 \cdot \log \frac{1}{\lambda_1}}, & \text{if} \quad 0 \leq \frac{\sigma_{\min}}{\lambda_2} \leq \frac{1}{e} \\ 1 - \frac{\sigma_{\min} \cdot \log \frac{\sigma_{\min}}{\lambda_2}}{\ln 2 \cdot \log \lambda_1}, & \text{otherwise} \end{cases} \quad (23)$$

Interpretation of the support lower bound: Given two symbolic series X_S and Y_S, and a minimum mutual information threshold $\mu_{\min}$. Theorem 1 says that, if X_S and Y_S are mutually dependent with the minimum MI value $\mu_{\min}$, then the support of an event pair in (X_S, Y_S) is at least the lower bound in Eq. (16). Combining Theorem 1 and Lemma 3, we can conclude that if an event pair of (X_S,Y_S) has a support less than the lower bound in Eq. (16), then any pattern P formed by that event pair also has support less than that lower bound. This allows us to construct an approximate version of GTPM (discussed in Section 5.5).

5.3 Lower bound of the Confidence of an Event Pair

Consider two events X_1, Y_1 of two symbolic series X_S and Y_S. We derive the confidence lower bound of (X_1, Y_1) in the sequence database $\mathcal{D}_{\text{SEQ}}$ as follows.

Theorem 2. *(Lower bound of the confidence) Let $\sigma_{\min}$ and $\mu_{\min}$ be the minimum support and minimum mutual information thresholds, respectively. Assume that $supp(X_1,Y_1)_{\mathcal{D}_{SEQ}} \geq \sigma_{\min}$. If the NMI $\widetilde{I}(X_S;Y_S) \geq \mu_{\min}$, then the lower bound of the confidence of (X_1, Y_1) in $\mathcal{D}_{SEQ}$ is:*

$$conf(X_1,Y_1)_{\mathcal{D}_{SEQ}} \geq \sigma_{\min} \cdot \lambda_1^{\frac{1-\mu_{\min}}{\sigma_{\min}}} \cdot \left(\frac{n_x - 1}{1 - \sigma_{\min}}\right)^{\frac{\lambda_3}{\sigma_{\min}}} \quad (24)$$

where n_x is the number of symbols in Σ_X, λ_1 is the minimum support of $X_i \in X_S$, and λ_3 is the support of $(X_i, Y_j) \in (X_S, Y_S)$ such that $p(X_i|Y_j)$ is minimal, $\forall (i \neq 1 \wedge j \neq 1)$.

Proof. (Sketch - Detailed proof in the electronic appendix). Let $\lambda_1 = p(X_k)$ such that $p(X_k) = \min\{p(X_i)\}, \forall i$, and $\lambda_3 = p(X_m, Y_n)$ such that $p(X_m|Y_n) = \min\{p(X_i|Y_j)\}, \forall (i \neq 1 \wedge j \neq 1)$. Then, by applying the min-max inequality theorem for the sum of ratio [59] to the numerator of Eq. (18), we obtain:

$$\frac{H(X_S|Y_S)}{H(X_S)} \geq \frac{p(X_1,Y_1) \cdot \log p(X_1|Y_1) + \lambda_3 \cdot \log \frac{1 - p(X_1,Y_1)}{n_x - p(Y_1)}}{\log \lambda_1} \geq \frac{\sigma_{\min} \cdot \log \frac{p(X_1,Y_1)}{p(Y_1)} + \lambda_3 \cdot \log \frac{1-\sigma_{\min}}{n_x - 1}}{\log \lambda_1} \quad (25)$$

Next, assume that $supp(Y_1)_{\mathcal{D}_{\text{SYB}}} \geq supp(X_1)_{\mathcal{D}_{\text{SYB}}}$. From Eqs. (18), (25), the confidence lower bound of (X_1, Y_1) in $\mathcal{D}_{\text{SYB}}$ is derived as:

$$conf(X_1,Y_1)_{\mathcal{D}_{\text{SYB}}} = \frac{supp(X_1,Y_1)_{\mathcal{D}_{\text{SYB}}}}{supp(Y_1)_{\mathcal{D}_{\text{SYB}}}} \geq \lambda_1^{\frac{1-\mu_{\min}}{\sigma_{\min}}} \cdot \left(\frac{n_x - 1}{1 - \sigma_{\min}}\right)^{\frac{\lambda_3}{\sigma_{\min}}} \quad (26)$$

Since: $$conf(X_1, Y_1)_{\mathcal{D}_{\text{SEQ}}} \geq \sigma_{\min} \cdot conf(X_1, Y_1)_{\mathcal{D}_{\text{SYB}}} \quad (27)$$

It follows that:

$$conf(X_1, Y_1)_{\mathcal{D}_{\text{SEQ}}} \geq \sigma_{\min} \cdot \lambda_1^{\frac{1-\mu_{\min}}{\sigma_{\min}}} \cdot \left(\frac{n_x - 1}{1 - \sigma_{\min}}\right)^{\frac{\lambda_3}{\sigma_{\min}}} \quad (28)$$

□

From Theorem 2, we can derive the minimum MI threshold $\mu_{\min}$ such that the confidence of (X_1, Y_1) is at least δ.

Corollary 2.1. *The confidence of an event pair $(X_1, Y_1) \in (X_S, Y_S)$ in $\mathcal{D}_{SEQ}$ is at least δ if $\widetilde{I}(X_S; Y_S)$ is at least $\mu_{\min}$, where:*

$$\mu_{\min} \geq 1 - \sigma_{\min} \cdot \log_{\lambda_1}\left(\frac{\delta}{\sigma_{\min}} \cdot \left(\frac{1 - \sigma_{\min}}{n_x - 1}\right)^{\frac{\lambda_3}{\sigma_{\min}}}\right) \quad (29)$$

Interpretation of the confidence lower bound: Given two symbolic series X_S and Y_S, and a minimum mutual information threshold $\mu_{\min}$. Theorem 2 says that, if X_S and Y_S are mutually dependent with the minimum MI value $\mu_{\min}$, then the confidence of an event pair in (X_S, Y_S) is at least the lower bound in Eq. (24). Combining Theorem 2 and Lemma 4, if an event pair of (X_S,Y_S) has a confidence less than the lower bound in Eq. (24), then any pattern P formed by that event pair also has a confidence less than that lower bound. This allows us to construct an approximate version of GTPM (discussed in Section 5.5).

5.4 Upper Bound of the Support of an Event Pair

We derive the support upper bound of the event pair (X_1, Y_1) of X_S and Y_S in $\mathcal{D}_{\text{SEQ}}$ as follows.

Theorem 3. *(Upper bound of the support) Let $\sigma_{\min}$ be the minimum support threshold, and $\mu_{\max}$ be the maximum mutual information threshold, respectively. Assume that $supp(X_1, Y_1)_{\mathcal{D}_{SEQ}} \geq \sigma_{\min}$. If the NMI $\widetilde{I}(X_S;Y_S) \leq \mu_{\max}$, then the upper bound of the support of (X_1, Y_1) in $\mathcal{D}_{SEQ}$ is:*

$$supp(X_1, Y_1)_{\mathcal{D}_{SEQ}} \leq \lambda_2 \cdot e^{W\left(\frac{\log \frac{\lambda_5^{1-\mu_{\max}}}{\lambda_4^{1-\sigma_{\min}}} \cdot \ln 2}{\lambda_2}\right)} + \vartheta \quad (30)$$

where: λ_2 is the support of $Y_1 \in Y_S$, λ_4 is the fraction between the support of $(X_i, Y_j) \in (X_S, Y_S)$ and the support of $Y_j \in Y_S$ such that $p(X_i|Y_j)$ is minimal, $\forall i \neq 1 \wedge j \neq 1$, λ_5 is the maximum support of $X_i \in X_S$, and ϑ is the difference between the probabilities of (X_1, Y_1) in $\mathcal{D}_{SEQ}$ and $\mathcal{D}_{SYB}$.

Proof. (Sketch - Detailed proof in the electronic appendix). Let $\lambda_2 = p(Y_1)$, $\lambda_4 = \min\{p(X_i|Y_j)\}\ \forall (i \neq 1 \wedge j \neq 1)$, and $\lambda_5 = \max\{p(X_i)\}\ \forall i$. We obtain:

$$\frac{H(X_S|Y_S)}{H(X_S)} \leq \frac{p(X_1, Y_1) \cdot \log \frac{p(X_1,Y_1)}{\lambda_2} + (1 - \sigma_{\min}) \cdot \log \lambda_4}{\log \lambda_5} \quad (31)$$

From Eqs. (10), we have:

$$\widetilde{I}(X_S; Y_S) = 1 - \frac{H(X_S|Y_S)}{H(X_S)} \leq \mu_{\max} \Rightarrow \frac{H(X_S|Y_S)}{H(X_S)} \geq 1 - \mu_{\max} \quad (32)$$

From Eqs. (31) and (32), we have:

$$\frac{p(X_1, Y_1) \cdot \log \frac{p(X_1,Y_1)}{\lambda_2} + (1 - \sigma_{\min}) \cdot \log \lambda_4}{\log \lambda_5} \geq 1 - \mu_{\max} \quad (33)$$

$$\Leftrightarrow p(X_1, Y_1) \leq \lambda_2 \cdot e^{W\left(\frac{\log \frac{\lambda_5^{1-\mu_{\max}}}{\lambda_4^{1-\sigma_{\min}}} \cdot \ln 2}{\lambda_2}\right)} \quad (34)$$

From Eq. (14), we have:

$$p(X_1, Y_1) = supp(X_1, Y_1)_{\mathcal{D}_{\text{SYB}}} = supp(X_1, Y_1)_{\mathcal{D}_{\text{SEQ}}} - \vartheta \quad (35)$$

From Eqs. (34) and (35), we have:

$$supp(X_1, Y_1)_{\mathcal{D}_{\text{SEQ}}} \leq \lambda_2 \cdot e^{W\left(\frac{\log \frac{\lambda_5^{1-\mu_{\max}}}{\lambda_4^{1-\sigma_{\min}}} \cdot \ln 2}{\lambda_2}\right)} + \vartheta \quad (36)$$

□

From Theorem 3, we can derive the maximum MI threshold $\mu_{\max}$ such that the support of (X_1, Y_1) is at most $\sigma_{\max}$.

Corollary 3.1. *The support of an event pair $(X_1, Y_1) \in (X_S, Y_S)$ in $\mathcal{D}_{SEQ}$ is at most $\sigma_{\max}$ if $\widetilde{I}(X_S; Y_S)$ is at most $\mu_{\max}$, where:*

$$\mu_{\max} \leq 1 - \frac{\frac{\sigma_{\max} - \vartheta}{\lambda_2} \cdot \log \frac{\sigma_{\max} - \vartheta}{\lambda_2} + \log \lambda_4^{1-\sigma_{\min}}}{\log \lambda_5} \quad (37)$$

Interpretation of the support upper bound: Given a maximum MI threshold $\mu_{\max}$, let X_S and Y_S be two symbolic series. Theorem 3 says that, if the NMI of X_S and Y_S is at most $\mu_{\max}$, then the support of an event pair in (X_S, Y_S) is at most the upper bound in Eq. (30). Combining Theorem 3 and Lemma 3, we can conclude that if an event pair in (X_S,Y_S) has a support less than the upper bound, then any pattern P formed by that event pair also has support less than that upper bound.

Setting the values of $\mu_{\min}$ and $\mu_{\max}$: GTPM uses three user-defined parameters, the minimum support $\sigma_{\min}$, the maximum support $\sigma_{\max}$, and the minimum confidence δ to mine both frequent and rare temporal patterns (with $\sigma_{\max}$ is set to ∞ in case of frequent patterns). To mine frequent patterns that satisfy both $\sigma_{\min}$ and δ constraints, we select $\mu_{\min}$ such that both Eqs. (23) and (29) hold, i.e., the maximum value of $\mu_{\min}$ provided by the two equations. On the other hand, to mine rare patterns that also have to satisfy $\sigma_{\max}$ constraint, $\mu_{\max}$ is chosen using Eq. (37).

Algorithm 2: Approximate GTPM using MI

Input: A set of time series $\mathcal{X}$, a minimum support threshold $\sigma_{\min}$, a maximum support threshold $\sigma_{\max}$, a minimum confidence threshold δ
Output: The set of temporal patterns P

1: Convert $\mathcal{X}$ to $\mathcal{D}_{\text{SYB}}$ and $\mathcal{D}_{\text{SEQ}}$;
2: Scan $\mathcal{D}_{\text{SYB}}$ to compute the probability of each event, event pair, and plus ϑ value;
3: **foreach** *pair of symbolic time series* $(X_S, Y_S) \in \mathcal{D}_{SYB}$ **do**
4: Compute $\widetilde{I}(X_S;Y_S)$ and $\widetilde{I}(Y_S;X_S)$;
5: Compute $\mu_{\min}$ using Eqs. (23) and (29);
6: Compute $\mu_{\max}$ using Eqs. (37);
7: **if** $\min\{\widetilde{I}(X_S;Y_S), \widetilde{I}(Y_S;X_S)\} \geq \mu_{\min}$ **then**
8: **if** $\min\{\widetilde{I}(X_S;Y_S), \widetilde{I}(Y_S;X_S)\} \leq \mu_{\max}$ **then**
9: Insert X_S and Y_S into X_C;
10: **foreach** $X_S \in \mathsf{X}_C$ **do**
11: Mine single events from X_S as in Section 4.3;
12: **foreach** $(X_S, Y_S) \in \mathsf{X}_C$ **do**
13: Mine 2-event patterns from (X_S, Y_S) as in Section 4.4;
14: **if** $k \geq 3$ **then**
15: Mine k-event patterns similar to the exact GTPM in Section 4.5;

5.5 Using the Bounds for Approximate GTPM

Approximate GTPM: Approximate GTPM is based on the exact GTPM and performs the mining only on *the set of mutually dependent symbolic series* $\mathsf{X}_C \in \mathcal{X}$ *with minimum and maximum MI thresholds* $\mu_{\min}$ *and* $\mu_{\max}$. Algorithm 2 describes the approximate GTPM. First, $\mathcal{D}_{\text{SYB}}$ is scanned once to compute the probability of each single event, pair of events, and plus ϑ value (line 2). Next, NMI, $\mu_{\min}$, and $\mu_{\max}$ are computed for each symbolic series pair (lines 4-6). The pairs of symbolic series whose $\min\{\widetilde{I}(X_S;Y_S), \widetilde{I}(Y_S;X_S)\}$ is at least $\mu_{\min}$, and $\min\{\widetilde{I}(X_S;Y_S), \widetilde{I}(Y_S;X_S)\}$ is at most $\mu_{\max}$ are inserted into X_C (lines 7-9). Then, we traverse each series in X_C to mine the single events (lines 10-11). Next, each event pair in corresponding series in X_C is employed to mine the 2-event patterns (lines 12-13). For k-event pattern ($k \geq 3$), the mining process is similar to GTPM (lines 14-15).

Complexity analysis of Approximate GTPM: To compute NMI, $\mu_{\min}$, and $\mu_{\max}$, we only have to scan $\mathcal{D}_{\text{SYB}}$ once to calculate the probability for each single event, pair of events, and plus ϑ value. Thus, the cost of NMI, $\mu_{\min}$, and $\mu_{\max}$ computations is $|\mathcal{D}_{\text{SYB}}|$. On the other hand, the complexity of the exact GTPM at HLH_1 and HLH_2 are $O(m^2 i^2\ |\mathcal{D}_{\text{SEQ}}|^2) + O(m \cdot |\mathcal{D}_{\text{SEQ}}|)$ (Sections 4.3 and 4.4). Thus, the approximate GTPM is significantly faster than the exact GTPM.

6 Experimental Evaluation

We evaluate GTPM in two different settings: to mine rare temporal patterns, named as RTPM, and to mine frequent temporal patterns, named as FTPM. Note that for RTPM, all three constraints $\sigma_{\min}$, $\sigma_{\max}$ and δ are used, whereas for FTPM, only $\sigma_{\min}$ and δ are used. In each setting, the performance of both exact and approximate versions are assessed. We use real-world datasets from four application domains: smart energy, smart city, sign language, and health. Due to space limitations, we only present here the most important results, and discuss other findings in the electronic appendix.

TABLE 4: Characteristics of the Datasets

	NIST	UKDALE	DataPort	SC	ASL	INF
# sequences	1460	1520	1460	1216	1908	608
# variables	49	24	21	26	25	25
# distinct events	98	48	42	130	173	124
# instances/seq.	55	190	49	162	20	48

TABLE 5: Parameters and values

Params	Values
Minimum support $\sigma_{\min}$	User-defined: $\sigma_{\min}$ = 0.2%, 0.4%, 0.6% 1%, 3% ...
Maximum support $\sigma_{\max}$	User-defined: $\sigma_{\max}$ = 2%, 6%, 10%, 15%, 20%, ...
Minimum confidence δ	User-defined: δ = 40%, 50%, 60%, 70%, 80%, ...
Overlapping duration t_{ov}	User-defined: t_{ov} (hours) = 0, 1, 2, 3 (NIST, UKDALE, DataPort, and SC) t_{ov} (frames) = 0, 150, 300, 450 (ASL) t_{ov} (days) = 0, 7, 10, 14 (INF)
Tolerance buffer ϵ	User-defined: ϵ (mins) = 0, 1, 2, 3 (NIST, UKDALE, and DataPort) ϵ (mins) = 0, 5, 10, 15 (SC) ϵ (frames) = 0, 30, 45, 60 (ASL) ϵ (days) = 0, 1, 2, 3 (INF)

6.1 Experimental Setup

Datasets: We use three *smart energy* (SE) datasets, NIST [60], UKDALE [61], and DataPort [62] that measure the energy consumption of electrical appliances in residential households. For the *smart city* (SC), we use weather and vehicle collision data obtained from NYC Open Data Portal [63]. For *sign language*, we use the American Sign Language (ASL) datasets [64] containing annotated video sequences of different ASL signs and gestures. For *health*, we combine the *influenza* (INF) dataset [65] and weather data [66] from Kawasaki, Japan. Table 4 summarizes their characteristics.

Baseline methods: Our exact RTPM version is referred to as E-RTPM, and the approximate one as A-RTPM. Since our work is the first that studies rare temporal pattern mining, there is not an exact baseline to compare against RTPM. However, we adapt the state-of-the-art method for frequent temporal pattern mining Z-Miner [22] to find rare temporal patterns. The Adapted Rare Z-Miner is referred to as ARZ-Miner. Similarly, we denote the exact FTPM version as E-FTPM, and the approximate one as A-FTPM. We use 4 baselines (detailed in Section 2) to compare with our FTPM: Z-Miner [22], TPMiner [3], IEMiner [4], and H-DFS [5]. Since the exact versions (E-RTPM and E-FTPM) and the baselines provide the same exact solutions, we use the baselines only for quantitative evaluation.

Infrastructure: We use a VM with 32 AMD EPYC cores (2GHz), 512 GB RAM, and 1 TB storage.

Parameters: Table 5 lists the parameters and their values used in our experiments.

6.2 Qualitative Evaluation

Rare temporal patterns: Table 6 shows several interesting rare temporal patterns extracted by RTPM. Patterns P1-P5 are from SC and P6-P8 are from INF. Analyzing these patterns can reveal some rare but interesting relations between temporal events. For example, P1-P5 show there exists an

association between extreme weather conditions and high accident numbers, such as high pedestrian injury during a heavy snowing day, which is very important to act on even though it occurs rarely.

Frequent temporal patterns: Table 7 lists some interesting frequent temporal patterns extracted by FTPM. Patterns P9-P15 are from SEs and P16-P18 are from ASL. Analyzing these patterns will reveal useful information about the domains. For example, P9-P15 show how the residents interact with electrical appliances in their houses. Specifically, P9 shows that a resident turns on the light upstairs in the early morning, and goes to the bathroom. Then, within a minute later, the microwave in the kitchen is turned on. This pattern occurs with minimum support of 20%, reflecting a living habit of the residents. Moreover, P9 also implies that there might be more than one person living in the house, in which one resident is in the bathroom while the other is downstairs preparing breakfast.

6.3 Quantitative Evaluation of RTPM

6.3.1 RTPM: Baseline comparison on real world datasets

We compare E-RTPM and A-RTPM with the adapted baseline ARZ-Miner in terms of runtime and memory usage. Figs. 7, 8, 9, and 10 show the comparison results on NIST and SC. Note that Figs. 7-14 use the same legend and log-scale y axes.

As shown in Figs. 7 and 8, A-RTPM achieves the best runtime among all methods, and E-RTPM has better runtime than the baseline. The range and average speedups of A-RTPM compared to other methods are: [1.9-7.2] and 3.4 (E-RTPM), [5.4-48.9] and 16.5 (ARZ-Miner). The speedup of E-RTPM compared to the baseline is [2.9-24.7] and 7.4 on average. Note that the time to compute MI, $\mu_{\min}$, and $\mu_{\max}$ for NIST and SC in Figs. 7 and 8 are 35.4 and 28.7 seconds, respectively, i.e., negligible compared to the total runtime.

In terms of memory consumption, as shown in Figs. 9 and 10, A-RTPM uses the least memory, while E-RTPM uses less memory than the baseline. A-RTPM consumes [1.6-3.9] (on average 2.1) times less memory than E-RTPM, and [7.2-120.6] (on average 24.1) times less than ARZ-Miner. E-RTPM uses [4.6-61.8] (on average 14.7) times less memory than ARZ-Miner.

6.3.2 RTPM: Scalability evaluation on synthetic datasets

As discussed in Section 4, the complexity of GTPM in general (and RTPM in particular) is driven by two main factors: (1) the number of temporal sequences, and (2) the number of time series. The evaluation on real-world datasets has shown that E-RTPM and A-RTPM outperform the baseline significantly in both runtimes and memory usage. However, to further assess the scalability of RTPM, we scale these two factors using synthetic datasets. Specifically, starting from the real-world datasets, we generate 10 times more sequences, and create up to 1000 synthetic time series. We then evaluate the scalability of RTPM in two scenarios: varying the number of sequences, and varying the number of time series.

Figs. 11 and 12 show the runtimes of A-RTPM, E-RTPM and the baseline when the number of sequences changes. We can see that A-RTPM and E-RTPM outperform and scale better than the baseline in this configuration. The range and average speedups of A-RTPM w.r.t. other methods are: [2.3-5.7] and 3.2 (E-RTPM), [5.1-19.8] and 12.5 (ARZ-Miner). Similarly, the range and average speedups of E-RTPM compared to ARZ-Miner are [2.7-7.6] and 5.3.

Figs. 13 and 14 compare the runtimes of A-RTPM with other methods when changing the number of time series. It is seen that, A-RTPM achieves highest speedup in this configuration. The range and average speedups of A-RTPM are [3.5-7.4] and 4.6 (E-RTPM), [7.2-24.8] and 15.2 (ARZ-Miner), and of E-RTPM is [3.6-9.5] and 6.4 (ARZ-Miner).

On average, E-RTPM consumes 17.2 times less memory than the baseline, while A-RTPM uses 20.6 times less memory than E-RTPM and the baseline in the scalability study. Furthermore, Fig. 13a shows that A-RTPM and E-RTPM can scale well on big datasets while the baseline cannot. Specifically, the baseline fails for large configurations as it runs out of memory, e.g., when # Time Series ≥ 1000 on the synthetic NIST. We add an additional bar chart for A-RTPM, including the time to compute MI, $\mu_{\min}$, and $\mu_{\max}$ (top red) and the mining time (bottom blue) for comparison, showing that this time is negligible.

Finally, the percentage of time series and events pruned by A-RTPM in the scalability test are provided in Table 8. Note that for the NIST dataset, every time series has two events, On and Off. Thus, the percentage of pruned time series and the percentage of pruned events are the same in NIST. We can see that the higher $\sigma_{\min}$, δ, and $\sigma_{\max}$, the more time series (events) are pruned. This is because higher $\sigma_{\min}$ and δ result in higher $\mu_{\min}$, and higher $\sigma_{\max}$ results in lower $\mu_{\max}$, and thus, more pruned time series.

6.3.3 E-RTPM: Evaluation of different pruning techniques

We evaluate the following combinations of E-RTPM pruning techniques: (1) NoPrune: E-RTPM with no pruning, (2) Apriori: E-RTPM with Apriori-based pruning (Lemmas 3, 4), (3) Trans: E-RTPM with transitivity-based pruning (Lemmas 5, 6, 7, 8), and (4) All: E-RTPM applied both pruning techniques.

We use 3 different scenarios that vary: the minimum support, the minimum confidence, and the maximum support. Figs. 15, 16 show the results. We see that (All)-E-RTPM has the best performance of all versions, with a speedup over (NoPrune)-E-RTPM ranging from 15 up to 74, depending on the configurations. Thus, the proposed prunings are very effective in improving E-RTPM performance. Furthermore, (Trans)-E-RTPM delivers a larger speedup than (Apriori)-E-RTPM, with the average speedup between 12 and 28 for (Trans)-E-RTPM, and between 7 and 19 for (Apriori)-E-RTPM, but applying both yields the best speedup.

6.3.4 A-RTPM: Evaluation of accuracy

To evaluate A-RTPM accuracy, we compare the patterns extracted by A-RTPM and E-RTPM. Table 9 shows the accuracies of A-RTPM for different $\sigma_{\min}$, δ, and $\sigma_{\max}$ on the real world datasets. It is seen that A-RTPM obtains high accuracy ($\geq 83\%$) with lowest $\sigma_{\min}$ and δ, and highest $\sigma_{\max}$, e.g., $\sigma_{\min} = 1\%$,$\delta = 60\%$, $\sigma_{\max} = 20\%$, and very high accuracy ($\geq 93\%$) with higher $\sigma_{\min}$ and δ, and lower $\sigma_{\max}$, e.g., $\sigma_{\min} = 3\%$, $\delta = 70\%$, $\sigma_{\max} = 10\%$.

TABLE 6: Summary of Interesting Rare Patterns

Patterns	$\sigma_{\min}$ (%)	δ (%)	$\sigma_{\max}$ (%)
(P1) Heavy Rain ≽ Unclear Visibility ≽ Overcast Cloudiness → High Motorist Injury	5	30	9
(P2) Heavy Rain ⟨⟩ Strong Wind → High Motorist Injury	2	40	6
(P3) Very Strong Wind → High Motorist Injury	5	40	9
(P4) Strong Wind ⟨⟩ High Pedestrian Injury	4	30	8
(P5) Extremely Unclear Visibility ≽ High Snow ≽ High Pedestrian Injury	3	45	7
(P6) Frost Temperature ⟨⟩ High Snow ≽ High Influenza	1	42	6
(P7) Low Temperature ≽High Influenza	1	42	6
(P8) Heavy Rain ≽ High Influenza	3	35	8

TABLE 7: Summary of Interesting Frequent Patterns

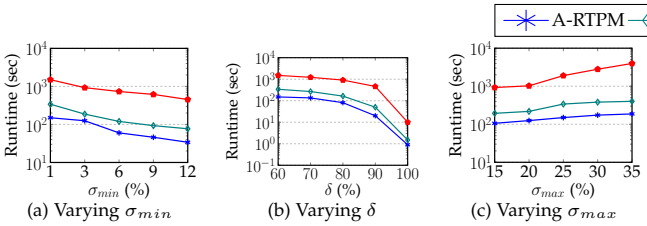
Patterns	$\sigma_{\min}$ (%)	δ (%)
(P9) ([05:58, 08:24] First Floor Lights) ≽ ([05:58, 06:59] Upstairs Bathroom Lights) ≽ ([05:59, 06:06] Microwave)	20	30
(P10) ([18:00, 18:30] Lights Dining Room) → ([18:31, 20:16] Children Room Plugs) ⟨⟩ ([19:00, 22:31] Lights Living Room)	20	20
(P11) ([15:59, 16:05] Hallway Lights) → ([17:58, 18:29] Kitchen Lights ≽ ([18:00, 18:18] Plug In Kitchen) ≽ ([18:08, 18:15] Microwave)	20	25
(P12) ([06:02, 06:19] Kitchen Lights) → ([06:05, 06:12] Microwave) ⟨⟩ ([06:09, 06:11] Kettle)	20	35
(P13) ([16:45, 17:30] Washer) → ([17:40,18:55] Dryer) → ([19:05, 20:10] Dining Room Lights) ≽ ([19:10, 19:30] Cooktop)	10	30
(P14) ([06:10, 07:00] Kitchen Lights) ≽ ([06:10, 06:15] Kettle) → ([06:30, 06:40] Toaster) → ([06:45, 06:48] Microwave)	25	40
(P15) ([18:00, 18:25] Kitchen Lights) ≽ ([18:00, 18:05] Kettle) → ([18:05, 18:10] Microwave) → ([19:35, 20:50] Washer)	20	40
(P16) [2.12 seconds] Negation ≽ [0.27 seconds] Lowered Eye-brows	10	10
(P17) [2.04 seconds] Negation ≽ [0.52 seconds] Rapid Shake-head	10	10
(P18) [1.53 seconds] Wh-question ≽ [0.36 seconds] Lowered Eye-brows → [0.05 seconds] Blinking Eye-aperture	10	15

TABLE 8: Pruned Time Series and Events from A-RTPM

# Attr.	$\sigma_{\min}$ (%) - δ (%) - $\sigma_{\max}$ (%)								
	NIST			SC					
	Pruned Time Series / Events (%)			Pruned Time Series (%)			Pruned Events (%)		
	6-80-20	3-70-15	1-60-10	6-80-20	3-70-15	1-60-10	6-80-20	3-70-15	1-60-10
200	59.50	39.50	22.50	48.50	30.50	15.50	39.10	25.10	11.90
400	58.50	38.25	21.25	45.75	29.75	14.75	37.55	24.30	11.45
600	56.50	36.17	19.83	43.17	27.17	14.33	36.43	23.03	10.57
800	51.63	35.88	19.63	42.38	23.88	14.25	33.55	21.28	10.30
1000	49.70	34.10	19.40	41.30	22.70	13.80	32.94	20.14	9.96

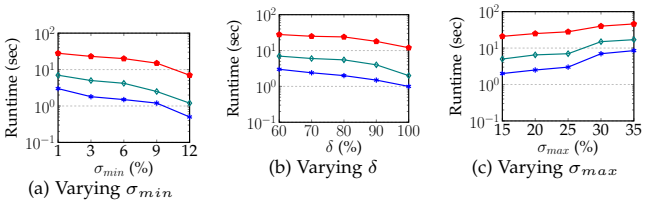
TABLE 9: RTPM Accuracy (%)

$\sigma_{\max}$ (%)	$\sigma_{\min}$ (%) - δ (%)					
	NIST			SC		
	1-60	3-70	6-80	1-60	3-70	6-80
10	93	96	100	91	93	100
15	86	92	95	86	91	100
20	84	92	92	83	87	90



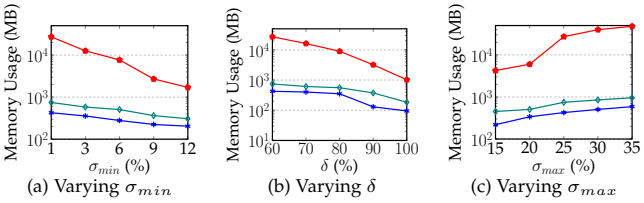
(a) Varying σ_{min} (b) Varying δ (c) Varying σ_{max}

Fig. 7: RTPM-Runtime Comparison on NIST (real-world)



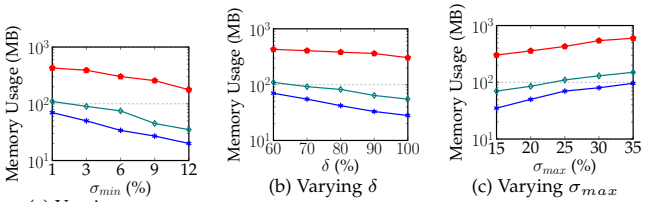
(a) Varying σ_{min} (b) Varying δ (c) Varying σ_{max}

Fig. 8: RTPM-Runtime Comparison on SC (real-world)



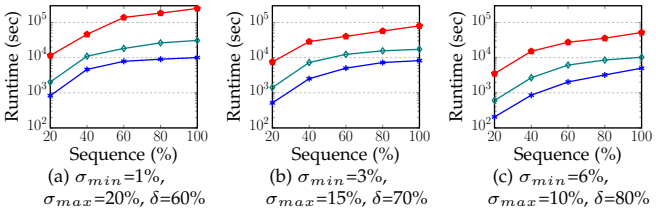
(a) Varying σ_{min} (b) Varying δ (c) Varying σ_{max}

Fig. 9: RTPM-Memory Usage Comparison on NIST (real-world)



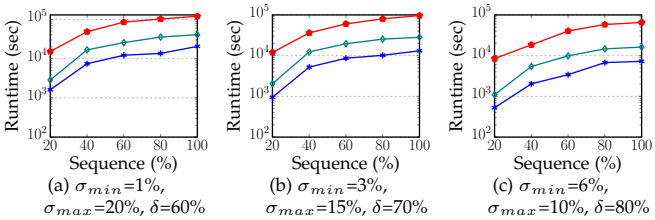
(a) Varying σ_{min} (b) Varying δ (c) Varying σ_{max}

Fig. 10: RTPM-Memory Usage Comparison on SC (real-world)



(a) σ_{min}=1%, σ_{max}=20%, δ=60% (b) σ_{min}=3%, σ_{max}=15%, δ=70% (c) σ_{min}=6%, σ_{max}=10%, δ=80%

Fig. 11: RTPM-Varying % of sequences on NIST (synthetic)



(a) σ_{min}=1%, σ_{max}=20%, δ=60% (b) σ_{min}=3%, σ_{max}=15%, δ=70% (c) σ_{min}=6%, σ_{max}=10%, δ=80%

Fig. 12: RTPM-Varying % of sequences on SC (synthetic)

6.4 Quantitative Evaluation of FTPM

6.4.1 FTPM: Baselines comparison on real world datasets

We compare E-FTPM and A-FTPM against the baselines in terms of runtime and memory usage. Further, we also compare E-FTPM and A-FTPM against E-HTPGM and A-HTPGM from the conference version [13] to assess the performance improvement obtained by using the new data structure. Figs. 17, 18, 19, and 20 show the experimental

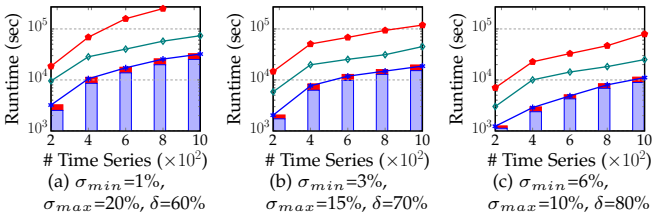


Fig. 13: RTPM-Varying # of time series on NIST (synthetic)

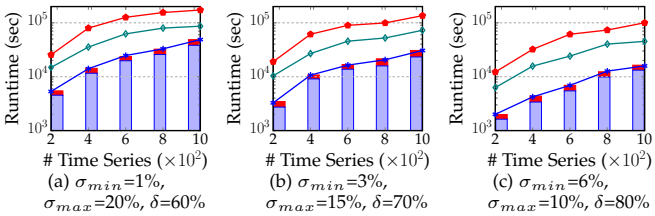


Fig. 14: RTPM-Varying # of time series on SC (synthetic)

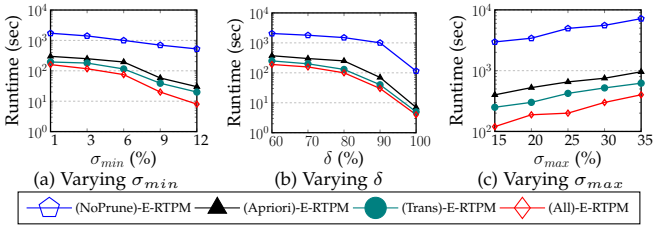


Fig. 15: Runtimes of E-RTPM on NIST (real-world)

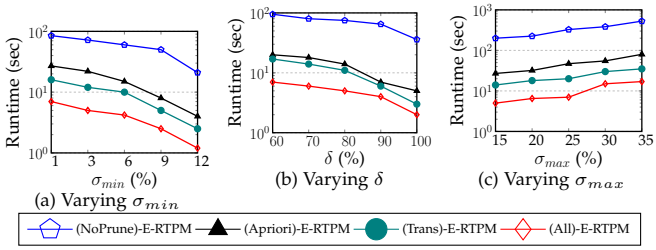


Fig. 16: Runtimes of E-RTPM on SC (real-world)

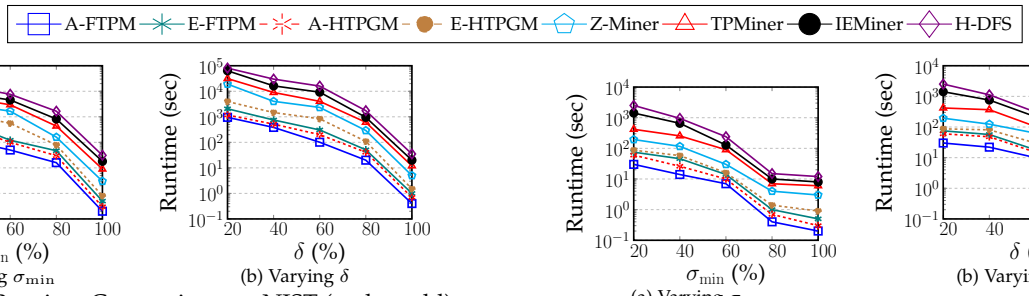


Fig. 17: FTPM-Runtime Comparison on NIST (real-world)

Fig. 18: FTPM-Runtime Comparison on SC (real-world)

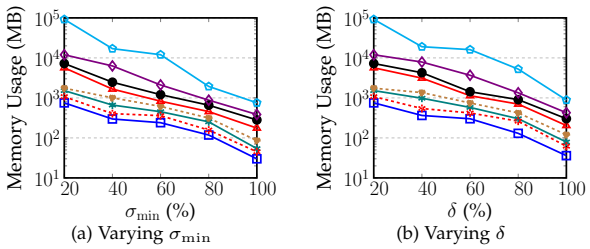


Fig. 19: FTPM-Memory Usage Comparison on NIST (real-world)

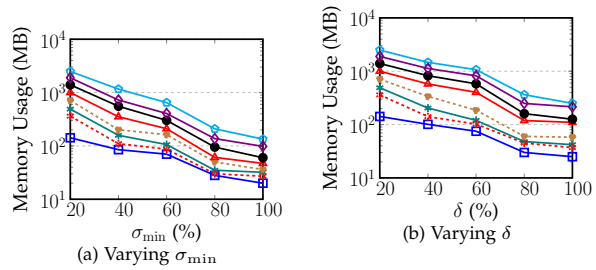


Fig. 20: FTPM-Memory Usage Comparison on SC (real-world)

TABLE 10: The Accuracy of A-FTPM (%)

σ_{min} (%)	δ (%)							
	NIST				SC			
	10	**20**	**50**	**80**	**10**	**20**	**50**	**80**
10	87	89	91	94	78	83	98	100
20	96	89	91	94	83	83	98	100
50	100	100	96	94	99	99	98	100
80	100	100	100	100	100	100	100	100

results on NIST and SC.

We can see from Figs. 17 and 18 that A-FTPM achieves the fastest runtime among all methods, and E-FTPM has faster runtime than the baselines. On the tested datasets, the range and average speedups of A-FTPM compared to E-FTPM is [1.5-6.1] and 2.7, and compared to the baselines is [4.2-356.1] and 45.8. The range and average speedup of E-FTPM compared to the baselines is [2.6-130.4] and 24.7.

Note that the time to compute MI and $\mu_{\min}$ for NIST and SC datasets in Figs. 17 and 18 are 32.6 and 26.4 seconds, respectively, making it negligible in the total runtime. Moreover, by using the improved hierarchical hash table instead of the hierarchical pattern tree in [13], both E-FTPM and A-FTPM are more efficient than E-HTPGM and A-HTPGM. The speedup of E-FTPM over E-HTPGM is in the range [1.1-4.7], and A-FTPM over A-HTPGM is in the range [1.3-5.6].

Finally, A-FTPM is most efficient, i.e., achieves highest speedup and memory saving, when the support threshold is low, e.g., $\sigma_{\min} = 20\%$. This is because typical datasets often contain many patterns with very low support and confidence. Thus, using A-FTPM to prune uncorrelated series early helps save computational time and resources. However, the speedup comes at the cost of a small loss in accuracy (discussed in Section 6.4.2).

In terms of memory consumption, as shown in Figs. 19 and 20, A-FTPM uses the least memory, while E-FTPM uses less memory than the baselines. A-FTPM consumes [1.4-3.6] (on average 1.9) times less memory than E-FTPM, and [6.8-112.6] (on average 15.4) times less than the baselines. E-FTPM uses [4.1-58.2] (on average 5.8) times less memory than the baselines. Compared to E-HTPGM and A-HTPGM [13], E-FTPM and A-FTPM are both more memory efficient. E-FTPM consumes [1.1-2.8] times less memory than E-

HTPGM, while A-FTPM uses [1.2-3.1] times less memory than A-HTPGM.

We also perform other experiments on FTPM, including scalability evaluation on synthetic datasets, and evaluation of different pruning techniques on real-world datasets as in RTPM. These experiments are reported in the electronic appendix.

6.4.2 A-FTPM: Evaluation of the accuracy

We proceed to evaluate the accuracy of A-FTPM by comparing the patterns extracted by A-FTPM and E-FTPM. Table 10 shows the accuracies of A-FTPM for different support and confidence thresholds on the real-world datasets. It is seen that A-FTPM obtains high accuracy ($\geq 78\%$) when $\sigma_{\min}$ and δ are low, e.g., $\sigma_{\min} = \delta = 10\%$, and very high accuracy ($\geq 95\%$) when $\sigma_{\min}$ and δ are high, e.g., $\sigma_{\min} = \delta = 50\%$.

Other experiments: We analyze the effects of the tolerance buffer ϵ, and the overlapping duration t_{ov} to the quality of extracted patterns. The analysis can be found in the electronic appendix.

7 Conclusion and Future Work

This paper presents our comprehensive Generalized Frequent Temporal Pattern Mining from Time Series (GTPMfTS) solution that offers: (1) an end-to-end GTPMfTS process to mine both rare and frequent temporal patterns from time series, (2) an efficient and exact Generalized Temporal Pattern Mining (GTPM) algorithm that employs efficient data structures and multiple pruning techniques to achieve fast mining, and (3) an approximate GTPM that uses mutual information to prune unpromising time series, allows GTPM to scale on big datasets. Extensive experiments conducted on real world and synthetic datasets for rare temporal pattern mining (RTPM) and frequent temporal pattern mining (FTPM) show that both exact and approximate algorithms for RTPM and FTPM outperform the baselines, consume less memory, and scale well on big datasets. Compared to the baselines, the approximate A-RTPM is up to an order of magnitude speedup and the approximate A-FTPM delivers two orders of magnitude speedup. In future work, we plan to extend GTPM to prune at the event level to further improve their performance.

References

[1] Energinet. (2021) Energi data portal. [Online]. Available: https://www.energidataservice.dk/tso-electricity/co2emis/

[2] K. Torp, O. Andersen, and C. Thomsen, "Travel-time computation based on gps data," in *Big Data Management and Analytics: 9th European Summer School*. Springer, 2020.

[3] Y. Chen, W. Peng, and S. Lee, "Mining temporal patterns in time interval-based data," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 27, 2015.

[4] D. Patel, W. Hsu, and M. L. Lee, "Mining relationships among interval-based events for classification," in *Proceedings of the ACM SIGMOD international conference on Management of data*, 2008.

[5] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos, "Mining frequent arrangements of temporal intervals," *Knowledge and Information Systems (KAIS)*, vol. 21, 2009.

[6] Y. Li and S. Cai, "Detecting outliers in data streams based on minimum rare pattern mining and pattern matching," *Information Technology and Control*, vol. 51, no. 2, 2022.

[7] Y. Cui, W. Gan, H. Lin, and W. Zheng, "Fri-miner: fuzzy rare itemset mining," *Applied Intelligence*, 2022.

[8] Y. Ji and Y. Ohsawa, "Mining frequent and rare itemsets with weighted supports using additive neural itemset embedding," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[9] S. Cai, J. Chen, H. Chen, C. Zhang, Q. Li, R. N. A. Sosu, and S. Yin, "An efficient anomaly detection method for uncertain data based on minimal rare patterns with the consideration of anti-monotonic constraints," *Information Sciences*, vol. 580, 2021.

[10] A. Rahman, "Rare sequential pattern mining of critical infrastructure control logs for anomaly detection," Ph.D. dissertation, Queensland University of Technology, 2019.

[11] M. Iqbal, C. P. Wulandari, W. Yunanto, and G. I. P. Sari, "Mining non-zero-rare sequential patterns on activity recognition," *Jurnal Matematika MANTIK*, vol. 5, no. 1, 2019.

[12] A. Samet, T. Guyet, and B. Negrevergne, "Mining rare sequential patterns with asp," in *International Conference on Inductive Logic Programming*, 2017.

[13] V. L. Ho, N. Ho, and T. B. Pedersen, "Efficient temporal pattern mining in big time series using mutual information," vol. 15, no. 3. VLDB Endowment, 2022.

[14] P.-s. Kam and A. W.-C. Fu, "Discovering temporal patterns for interval-based events," in *Data Warehousing and Knowledge Discovery (DaWak)*, 2000.

[15] S.-Y. Wu and Y.-L. Chen, "Mining nonambiguous temporal patterns for interval-based events," *EEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 19, 2007.

[16] R. Moskovitch and Y. Shahar, "Fast time intervals mining using the transitivity of temporal relations," *Knowledge and Information Systems*, vol. 42, 2015.

[17] I. Batal, D. Fradkin, J. Harrison, F. Moerchen, and M. Hauskrecht, "Mining recent temporal patterns for event detection in multivariate time series data," in *Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining*, 2012.

[18] 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 Transactions on Intelligent Systems and Technology (TIST)*, vol. 11, no. 4, 2020.

[19] A. K. Sharma and D. Patel, "Stipa: A memory efficient technique for interval pattern discovery," in *IEEE International Conference on Big Data (Big Data)*. IEEE, 2018.

[20] I. Batal, H. Valizadegan, G. F. Cooper, and M. Hauskrecht, "A temporal pattern mining approach for classifying electronic health record data," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, 2013.

[21] E. A. Campbell, E. J. Bass, and A. J. Masino, "Temporal condition pattern mining in large, sparse electronic health record data: A case study in characterizing pediatric asthma," *Journal of the American Medical Informatics Association*, vol. 27, 2020.

[22] Z. Lee, T. Lindgren, and P. Papapetrou, "Z-miner: an efficient method for mining frequent arrangements of event intervals," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[23] Y. Gao and J. Lin, "Efficient discovery of time series motifs with large length range in million scale time series," in *IEEE International Conference on Data Mining (ICDM)*. IEEE, 2017.

[24] N. Begum and E. Keogh, "Rare time series motif discovery from unbounded streams," *Proceedings of the VLDB Endowment*, vol. 8, no. 2, 2014.

[25] E. Alipourchavary, S. M. Erfani, and C. Leckie, "Mining rare recurring events in network traffic using second order contrast patterns," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[26] S. Bouasker, W. Inoubli, S. B. Yahia, and G. Diallo, "Pregnancy associated breast cancer gene expressions: new insights on their regulation based on rare correlated patterns," *Transactions on Computational Biology and Bioinformatics*, vol. 18, no. 3, 2020.

[27] A. Borah and B. Nath, "Rare association rule mining from incremental databases," *Pattern Analysis and Applications*, vol. 23, 2020.

[28] P. Fournier-Viger, P. Yang, Z. Li, J. C.-W. Lin, and R. U. Kiran, "Discovering rare correlated periodic patterns in multiple sequences," *Data & Knowledge Engineering*, vol. 126, 2020.

[29] S. Biswas and K. C. Mondal, "Dynamic fp tree based rare pattern mining using multiple item supports constraints," in *Computational Intelligence, Communications, and Business Analytics (CICBA)*. Springer, 2019.

[30] S. Piri, D. Delen, T. Liu, and W. Paiva, "Development of a new metric to identify rare patterns in association analysis: The case

of analyzing diabetes complications," *Expert Systems with Applications*, vol. 94, 2018.
[31] C. Cappiello, N. T. T. Ho, B. Pernici, P. Plebani, and M. Vitali, "Co 2-aware adaptation strategies for cloud applications," *IEEE Transactions on Cloud Computing*, vol. 4, no. 2, pp. 152–165, 2015.
[32] A. Barkat, A. D. dos Santos, and T. T. N. Ho, "Open stack and cloud stack: Open source solutions for building public and private clouds," in *2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. IEEE, 2014, pp. 429–436.
[33] T. T. N. Ho, "Towards sustainable solutions for applications in cloud computing and big data," in *Doctoral dissertation*. Politecnico di Milano, Italy, 2017, http://hdl.handle.net/10589/131740.
[34] ——, "Activity recognition using smartphone based sensors," in *Master thesis*. Politecnico di Milano, Italy, 2013, http://hdl.handle.net/10589/85064.
[35] A. Rahman, Y. Xu, K. Radke, and E. Foo, "Finding anomalies in scada logs using rare sequential pattern mining," in *Network and System Security*. Springer, 2016.
[36] J. Zhu, K. Wang, Y. Wu, Z. Hu, and H. Wang, "Mining user-aware rare sequential topic patterns in document streams," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 28, no. 7, 2016.
[37] W. Ouyang, "Mining rare sequential patterns in large transaction databases," in *International Conference on Computer Science and Electronic Technology*. Atlantis Press, 2016.
[38] A. U. Ahmed, C. F. Ahmed, M. Samiullah, N. Adnan, and C. K.-S. Leung, "Mining interesting patterns from uncertain databases," *Information Sciences*, vol. 354, 2016.
[39] Y.-K. Lee, W.-Y. Kim, Y. D. Cai, and J. Han, "Comine: Efficient mining of correlated patterns," in *IEEE International Conference on Data Mining (ICDM)*, 2003.
[40] Y. Ke, J. Cheng, and W. Ng, "Correlated pattern mining in quantitative databases," *ACM Transactions on Database Systems (TODS)*, vol. 33, 2008.
[41] N. Ho, T. B. Pedersen, M. Vu, and C. A. Biscio, "Efficient bottom-up discovery of multi-scale time series correlations using mutual information," in *International Conference on Data Engineering (ICDE)*. IEEE, 2019.
[42] N. Ho, H. Vo, M. Vu, and T. B. Pedersen, "Amic: An adaptive information theoretic method to identify multi-scale temporal correlations in big time series data," *IEEE Transactions on Big Data*, vol. 7, no. 1, 2019.
[43] J. Blanchard, F. Guillet, R. Gras, and H. Briand, "Using information-theoretic measures to assess association rule interestingness," in *IEEE International Conference on Data Mining (ICDM)*, 2005.
[44] X. Cunjin, S. Wanjiao, Q. Lijuan, D. Qing, and W. Xiaoyang, "A mutual-information-based mining method for marine abnormal association rules," *Computers & Geosciences*, vol. 76, 2015.
[45] N. Ho, V. L. Ho, T. B. Pedersen, and M. Vu, "Efficient and distributed temporal pattern mining," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 335–343.
[46] Y. Yao, "Information-theoretic measures for knowledge discovery and data mining," in *Entropy measures, maximum entropy principle and emerging applications*, 2003.
[47] N. T. T. Ho, T. B. Pedersen, L. Van Ho, and M. Vu, "Efficient search for multi-scale time delay correlations in big time series," in *International Conference on Extending Database Technology (EDBT)*. OpenProceedings. org, 2020.
[48] N. Ho, H. Vo, and M. Vu, "An adaptive information-theoretic approach for identifying temporal correlations in big data sets," in *IEEE International Conference on Big Data (Big Data)*. IEEE, 2016.
[49] T. T. N. Ho and B. Pernici, "A data-value-driven adaptation framework for energy efficiency for data intensive applications in clouds," in *IEEE conference on technologies for sustainability (SusTech)*. IEEE, 2015.
[50] M. Gribaudo, T. T. N. Ho, B. Pernici, and G. Serazzi, "Analysis of the influence of application deployment on energy consumption," in *Energy Efficient Data Centers (E2DC)*. Springer, 2015.
[51] T. T. N. Ho, M. Gribaudo, and B. Pernici, "Characterizing energy per job in cloud applications," *Electronics*, vol. 5, no. 4, 2016.
[52] N. Ho, M. Gribaudo, and B. Pernici, "Improving energy efficiency for transactional workloads in cloud environments," in *Proceedings of the Eighth International Conference on Future Energy Systems*, 2017.
[53] J. Lin, E. Keogh, S. Lonardi, and B. Chiu, "A symbolic representation of time series, with implications for streaming algorithms," in *Proceedings of the ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2003.
[54] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, 1983.
[55] E. R. Omiecinski, "Alternative interest measures for mining associations in databases," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 15, no. 1, 2003.
[56] V. L. Ho, N. Ho, and T. B. Pedersen, "Mining seasonal temporal patterns in time series," in *IEEE International Conference on Data Engineering (ICDE)*. IEEE, 2023.
[57] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.
[58] R. M. Corless, G. H. Gonnet, D. E. Hare, D. J. Jeffrey, and D. E. Knuth, "On the lambert w function," *Advances in Computational mathematics*, vol. 5, 1996.
[59] E. F. Beckenbach, R. Bellman, and R. E. Bellman, "An introduction to inequalities," Mathematical Association of America Washington, DC, Tech. Rep., 1961.
[60] W. Healy, F. Omar, L. Ng, T. Ullah, W. Payne, B. Dougherty, and A. H. Fanney. (2018) Net zero energy residential test facility instrumented data. [Online]. Available: https://pages.nist.gov/netzero/index.html/
[61] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," *Scientific Data*, 2015.
[62] P. S. Data. (2016) Pecan street dataport. [Online]. Available: https://www.pecanstreet.org/dataport/
[63] N. Y. City. (2019) Nyc opendata. [Online]. Available: https://opendata.cityofnewyork.us/
[64] C. Neidle, A. Opoku, G. Dimitriadis, and D. Metaxas, "New shared & interconnected asl resources: Signstream® 3 software; dai 2 for web access to linguistically annotated video corpora; and a sign bank," in *Workshop on the Representation and Processing of Sign Languages: Involving the Language Community, Miyazaki, Language Resources and Evaluation Conference*, 2018.
[65] K. city infectious disease surveillance system. (2021) Kidss. [Online]. Available: https://kidss.city.kawasaki.jp/
[66] O. Weather. (2021) Open weather. [Online]. Available: https://openweathermap.org/