Why Not Match: On Explanations of Event Pattern Queries

Shaoxu Song
Tsinghua University
Beijing, China
sxsong@tsinghua.edu.cn

Ruihong Huang
Tsinghua University
Beijing, China
hrh16@mails.tsinghua.edu.cn

Yu Gao
Tsinghua University
Beijing, China
gaoyu15@mails.tsinghua.edu.cn

Jianmin Wang
Tsinghua University
Beijing, China
jimwang@tsinghua.edu.cn

ABSTRACT

Queries over event data are posed in a form of event patterns, for example, to retrieve the flights from IAH to LGA without a stopover. If the expected answer is not returned, one may ask why not, also known as explanations of non-answers. Analogous to the relational data, the explanations over event data lie in two aspects. (1) The *pattern consistency explanation* indicates that the patterns specified in the query are wrong (inconsistent), that is, there exists no tuple of events that can match the query. (2) The *timestamp modification explanation* speculates that the instance of event tuple is incorrect, for example, the timestamps of some events are imprecise and need modification. To the best of our knowledge, this is the first study on explaining non-answers over event data. We prove that both explanation problems are NP-complete. By encoding event patterns as a novel notation, we identify the special cases that can be efficiently solved or approximated. General cases are addressed by utilizing the solutions of special cases. Extensive experiments over real and synthetic datasets demonstrate both effectiveness and efficiency of our proposal.

ACM Reference Format:
Shaoxu Song, Ruihong Huang, Yu Gao, and Jianmin Wang. 2021. Why Not Match: On Explanations of Event Pattern Queries. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD '21), June 20–25, 2021, Virtual Event, China.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3448016.3452818

1 INTRODUCTION

Event patterns [12] are often employed to specify the queries over event data in complex event processing [5, 6], where event data are the instances of events associated with the corresponding occurrence timestamps and event patterns specify the constraints on event occurrence. Let us consider a motivating example of event patterns on flights for tracing COVID-19 cases.

SIGMOD '21, June 20–25, 2021, Virtual Event, China

ACM ISBN 978-1-4503-8343-1/21/06…$15.00
https://doi.org/10.1145/3448016.3452818

Table 1: Tuple t_1 of flights on 07/01 matches the patterns of the query in Example 1. Another expected to match tuple t_2 does not and thus needs explanation.

Tuple	E_1 (UA104 arrival)	E_2 (AA514 departure)	E_3 (UA729 arrival)	E_4 (CO193 departure)
t_1 (07/01)	17:08	18:58	17:25	19:13
t_2 (07/02)	17:06	18:54	17:24	20:08 (19:08)

Example 1. Suppose that there are COVID-19 cases reported who transferred in LGA and stayed there for at least one hour. Let E_1 denote the arrival event of flight UA104 and E_2 be the departure event of flight AA514, taken by the reported passenger. The other passengers transferring in a similar period should be traced, by a query with pattern

$$p_0 : \texttt{SEQ}(\texttt{AND}(E_1, E_3)\ \texttt{WITHIN}\ 30\ \text{minutes},\ \texttt{AND}(E_2, E_4)\ \texttt{WITHIN}\ 30\ \text{minutes})\ \texttt{ATLEAST}\ 2\ \text{hours}$$

where E_3 is the arrival time and E_4 is the departure time of the traced passenger. The keyword SEQ denotes the sequential occurrence of events or patterns, while AND means concurrent occurrence. WITHIN and ATLEAST specify the maximum and minimum timestamp distances of events occurring in the corresponding pattern, respectively. The nested sub-pattern $\texttt{AND}(E_1, E_3)$ WITHIN 30 minutes states that two passengers can arrive in any order within 30 minutes. Likewise, $\texttt{AND}(E_2, E_4)$ WITHIN 30 minutes means that they depart again in any order with time difference no greater than 30 minutes. The SEQ pattern requires that the arrival time of both passengers should be earlier than both their departure time, in particular, E_1 of the reported passenger earlier than E_4 of the traced passenger, and similarly E_3 earlier than E_2. In other words, the transfer periods of two passengers overlap. The term ATLEAST 2 hours of SEQ implies that the overlap time is at least 1 hour, given the aforesaid WITHIN 30 minutes of both AND sub-patterns. Although the pattern cannot exactly find out all the contacts, it helps reducing the search space. □

1.1 Explanations of Non-Answers

Unfortunately, for various reasons as illustrated below, some expected answers of the user may be absent. Existing studies on non-answer explanations of database queries are mainly in two aspects, that is, (1) query explanation that focuses on the issues in query [10, 27], and (2) data explanation that addresses the issues in the

data [18, 19]. Analogously, in this paper, for the non-answers to event pattern queries, we also study (1) the pattern consistency explanation on queries and (2) the timestamp modification explanation over data. It is worth noting that only when given a set of consistent event patterns, the explanation on data modification is possible. Moreover, imprecise timestamps are prevalent in practice and often lead to non-answers [32]. In this sense, the studied explanations are the two top problems for explaining event pattern queries, analogous to the two modes of query and data explanations for database queries, respectively.

1.1.1 Pattern Consistency Explanation. The *pattern consistency explanation* is to determine whether there exists an assignment of timestamps that can satisfy the given query patterns, also known as the pattern consistency problem. If the query pattern is not consistent, it naturally explains why non-answers.

During the query development time, the pattern consistency explanation could be used to avoid an inconsistent pattern and help the developer pose the desired query. Suppose that the following event pattern query is developed to trace the passengers transferred in LGA, again in monitoring COVID-19 cases in Example 1.

$$\texttt{SEQ}(\texttt{AND}(E_1, E_3)\ \texttt{ATLEAST 30 minutes},\ \texttt{AND}(E_2, E_4)\ \texttt{ATLEAST 30 minutes})\ \texttt{WITHIN 45 minutes}$$

The pattern consistency explanation could directly tell the developer that no answer will be returned for this inconsistent pattern, without evaluating in the data. That is, no events could satisfy the condition specified in the pattern, since the events in two AND patterns need at least 30+30 minutes which is unlikely to occur in a total of 45 minutes in sequence (SEQ).

1.1.2 Timestamp Modification Explanation. The *timestamp modification explanation* is to minimally modify the timestamps of events, in order to make them match the given event patterns, also known as the timestamp modification problem. It explains that the timestamps are wrong and should be modified to make the tuple an answer to the query.

During debugging, if an expected result is not returned, the timestamp modification explanation could help the developer in finding out the problems in data. Consider again the scenario of monitoring potential COVID-19 cases by the event pattern query p_0 in Example 1. Suppose that the user already knew some suspected passengers transferred in LGA, but not returned in the aforesaid event pattern query. The timestamp modification explanation suggests that it is because the departure time $t_2[E_4]$ = 20:08 of flight CO193 is wrong (e.g., owing to semantics ambiguity, out-of-date data or pure errors [24]) and should be modified to 19:24 in order to appear as the expected result in the aforesaid event pattern query.

It is true that arbitrary changes of timestamps to satisfy the query patterns do not make sense. For example, an explanation excessively modifying the timestamps of all flights in a day (the whole t_2 in Table 1) to any other day (such as t_1 that matches the pattern) is meaningless. Therefore, rather than considering all the possible worlds with an arbitrary combination of all timestamps [32], the possible explanations (by modifying the data in databases [18]) need to be ranked [23]. In our scenario of events, we propose to obtain the most likely modification explanation of imprecise timestamps to satisfy the query pattern, following the minimum change idea in data cleaning [13]. The rationale is that people or systems always try to minimize their mistakes in practice. Without further knowledge (on the truth 19:08 in Table 1), the minimally modified explanation $t_2'[E_4]$=19:24 is the best attempt to match the query pattern and avoid over-modifying the timestamps.

If even the minimum change explanation needs to significantly modify the timestamps on a great many of events, the non-answers may not be caused by imprecise timestamps and the timestamp modification explanation does not apply. It is interesting to investigate further explanation types in the future study.

1.2 Application Scenarios

Data-driven or data-informed decision making has become increasingly important recently [25]. In order monitoring [12], an event pattern query could be

$$\texttt{SEQ}(\texttt{AND}(\texttt{SEQ}(E_1, E_2), \texttt{SEQ}(E_3, E_4)), E_5)\ \texttt{WITHIN 12 hours}.$$

It monitors the cancelled orders that involve the participation of both suppliers and remote stocks. E_1, E_2 denote the events of ordering from the supplier and generating the quote with a high price, respectively. E_3, E_4 are the events of using remote stock and generating invoice, respectively. E_5 means the event of cancelling the order. Frequent occurrences of such patterns may indicate the need for an immediate inventory management. (1) With the pattern consistency explanation in Section 1.1.1, the user will notice the inconsistency if the sub-pattern is mistakenly written as $\texttt{SEQ}(E_1, E_2)$ ATLEAST 24 hours in the query. (2) If it is already known that the local inventory is insufficient, but no answer is returned in the aforesaid alert query, with the help of timestamp modification explanation in Section 1.1.2, one may identify that the timestamp of event E_4 (generate invoice) is reset to 00:00 and should be modified.

In vehicle tracking [20], an event pattern query could be

$$\texttt{SEQ}(E_1, \texttt{AND}(E_2, E_3)\ \texttt{ATLEAST 30 minutes}, E_4)\ \texttt{WITHIN 2 hours},$$

where $E_1, \ldots, E_4$ denote the check points of excavation, weighting, height measuring and unloading, separately. Subpattern $\texttt{AND}(E_2, E_3)$ implies that E_2 (weighting) and E_3 (height measuring) could take place in any order, both of which should occur after E_1 (excavation) before E_4 (unloading) referring to the SEQ predicate. The pattern query is used to count the number of complete trips. (1) Again, inconsistency such as ATLEAST 30 hours rather than minutes in the query can be identified by the pattern consistency explanation in Section 1.1.1. (2) If the count of matched trips is less than expected, the timestamp modification explanation in Section 1.1.2 may indicate the incomplete timestamp 11:– as well as its modification.

In monitoring the job execution in a cluster [16], suppose that a maximum number of two concurrent jobs are allowed at the same time. The following event pattern detects the termination of the first submitted job due to two new high priority jobs being scheduled, which takes at least 2 minutes,

$$\texttt{SEQ}(E_1, \texttt{AND}(E_2, E_3), E_4)\ \texttt{ATLEAST 2 minutes}$$

where E_1, E_4 denote the submission and termination of the first job, and E_2, E_3 denote the submissions of the second and third jobs. The $\texttt{AND}(E_2, E_3)$ pattern implies that two new jobs could be submitted in any order. (1) The pattern consistency explanation in Section 1.1.1 could easily detect the trivial inconsistency like ATLEAST 2 minutes

Table 2: Major results (complex temporal networks ⊃ simple temporal networks)

Event pattern	Temporal network	Pattern consistency problem	Pattern consistency Algorithm 1	Timestamp modification problem	Timestamp modification Algorithm 2
general [12] (Definition 1)	complex (Definition 5)	NP-complete (Theorem 2)	exact $O(f^{\|\Gamma\|}n^3)$	NP-complete (Theorem 3)	exact (approximate by single binding, Definition 8)
no SEQ embedded in AND	complex (Definition 5)	–	exact $O(f^{\|\Gamma\|}n^3)$	–	optimal by single binding, Proposition 8
no AND	simple [11] (Definition 6)	PTIME (Section 4.1)	exact $O(n^3)$	–	exact by ILP in Formula 3 (LP-relaxation)

WITHIN 1 minute. (2) If the user had a job terminated but failed to know why (no matched result of the aforesaid pattern query), the timestamp modification explanation in Section 1.1.2 may suggest the reverse order of E_4 (old job termination) and E_3 (new job submission), owing to latency or out-of-order messages.

1.3 Contributions

To the best of our knowledge, this is the first study on explaining non-answers over event data. Table 2 summarizes the major results. The "–" entry in Table 2 means that the theoretical bound is not established yet, and the corresponding problem is still open.

(1) We analyze the complexity of the studied problems in Section 2. The pattern consistency problem for query pattern explanation is NP-complete (Theorem 2). The timestamp modification problem for tuple timestamp explanation is not only NP-complete but also NP-hard to approximate to within any constant factor (Theorems 3 and 4).

(2) We present an encoding of event patterns in Section 3. The benefit is to identify the special cases that can be efficiently solved or approximated. It enables us to address the general case, by assembling the solutions of the aforesaid special cases.

(3) We show that the existing constraint [11] on event pairs is a special case. Existing cubic-time algorithm [11] is thus used for pattern consistency checking. Moreover, we devise an integer linear programming (ILP) formulation and the corresponding efficient LP-relaxation for generating timestamp modification.

(4) We devise exact and approximate algorithms for general cases by using the solutions in special cases. The approximation not only achieves good performances on pattern consistency checking and tuple timestamp modification (Section 6), but also guarantees theoretically the optimal solution in some case (Proposition 8).

Finally, we report an extensive experimental evaluation, in Section 6, to demonstrate the efficiency and effectiveness of explaining non-answers over event data. While existing studies on explanations of relational queries are not directly applicable over event data, we discuss the connections to related studies in Section 7.

2 PROBLEM STATEMENT AND ANALYSIS

Let $\mathcal{E} = \{E_1, \ldots, E_n\}$ be a set of n events and T be the domain of timestamps (non-negative integers). A tuple is a sequence of event instances over $\mathcal{E}$, having no duplicated events. There is a one-to-one mapping from $\mathcal{E}$ to T, i.e., $t : \mathcal{E} \to T$, where $t[E_i]$ denotes the occurrence timestamp of event $E_i \in \mathcal{E}$. For example, the arrival event E_1 of a flight is associated with one timestamp $t[E_1]$, while another departure event E_2 has its own timestamp $t[E_2]$.

2.1 Event Pattern Query

We use an existing language of event patterns in complex event processing [8, 12], presented in Definition 1. For better solution and approximation, we create a novel notation of complex temporal networks to encode event patterns, in Definition 5 in Section 3.

Definition 1 (Event Pattern). *An event pattern p_0 over $\mathcal{E}$ specifies particular occurrence orders of events in $\mathcal{E}$, defined recursively:*
(1) $p_0 = E$ is an event pattern with a single event $E \in \mathcal{E}$;
(2) $p_0 =$ SEQ$(p_1, \ldots, p_k)$ [ATLEAST a] [WITHIN b] is an event pattern in which the patterns $p_i, i \in 1, \ldots, k$, occur sequentially [in a time period with length $\geq a$ and $\leq b$];
(3) $p_0=$ AND$(p_1, \ldots, p_k)$ [ATLEAST a] [WITHIN b] is an event pattern in which the patterns $p_i, i \in 1, \ldots, k$, occur concurrently [in a time period with length $\geq a$ and $\leq b$];
where [ATLEAST a] or [WITHIN b] is optional in a pattern, and $a \leq b$.

Other attributes could be used in the query language [12, 30], such as SEQ(E_1, E_2) WHERE E1.gate="H15". The WHERE clause filters or considers a property of an event. In such a scenario, the existing studies on database query explanations [10, 18, 19, 27] could be applied first to explain the filtering on property gate of event E_1 in the WHERE clause. Then, our explanations on the event patterns are performed over the filtered events. In this sense, our proposal complements the existing database query explanations (on attribute filtering) by supporting event pattern explanations.

Definition 2. *A tuple (trace) t* matches *a pattern p_0, denoted by $t \vDash p_0$, if*
(1) for $p_0 = E$, it has $t[p_0^s] = t[p_0^e] = t[E]$, where $t[p_0^s]$ is the start timestamp of p_0, $t[p_0^e]$ is the end timestamp of p_0, and $t[E]$ is the timestamp of E recorded in the tuple;
(2) for $p_0 =$ SEQ$(p_1, p_2, \ldots, p_k)$ [ATLEAST a] [WITHIN b], it has

$$t[p_0^s] = t[p_1^s], t[p_0^e] = t[p_k^e],$$
$$t[p_1^s] \leq t[p_1^e] \leq t[p_2^s] \leq t[p_2^e] \leq \cdots \leq t[p_k^s] \leq t[p_k^e],$$
$$t \vDash p_1, \ldots, t \vDash p_k$$

[and $a \leq t[p_0^e] - t[p_0^s] \leq b$];
(3) for $p_0 =$ AND$(p_1, p_2, \ldots, p_k)$ [ATLEAST a] [WITHIN b],

$$t[p_0^s] = \min(t[p_1^s], t[p_2^s], \ldots, t[p_k^s]),$$
$$t[p_0^e] = \max(t[p_1^e], t[p_2^e], \ldots, t[p_k^e]), t \vDash p_1, \ldots, t \vDash p_k$$

[and $a \leq t[p_0^e] - t[p_0^s] \leq b$].

Occurring concurrently in AND pattern means that p_i (if occurs) could be in any order, such as $(p_1, p_2, \ldots, p_k)$ or $(p_2, p_1, \ldots, p_k)$. Referring to Definition 2, it is not surprising that whether a tuple matches the event pattern can be efficiently checked.

Proposition 1. *Given a tuple t and a pattern p_0, the determination of whether $t \vDash p_0$ can be solved in $O(n^2)$ time, where $n = |\mathcal{E}|$.*

Due to space limitations, we prove Proposition 1 in the technical report [3].

2.2 Pattern Consistency Explanation

In the why-not explanations for relational data queries, the query explanation finds the issues in the query [10] and modifies the query to make a non-answer an answer [15]. In this paper, as the first study of why-not-match explanations for event pattern queries, we propose to take the first step on whether the given event pattern query is satisfiable and leave the event query modification as the future study. The reasons are as follows. (1) The satisfiability of the event pattern query should be first checked before conducting the timestamp (data) modification explanation. That is, no timestamp modification explanation could be generated when given an inconsistent event pattern query. (2) The consistency checking for event pattern queries are already hard, as illustrated in Theorem 2 below. Thereby, we leave the challenging problem of event pattern query modification explanation as the future study.

The pattern consistency explanation indicates that no answers are returned since the patterns specified in the query are simply wrong (unsatisfiable). Indeed, determining whether a query is satisfiable has also been studied in semantic query optimization [17] to explain why a query has no need to be conducted (since unsatisfiable). For instance, the SQL query over relation Goods, [SELECT * FROM Goods WHERE Sale_Price $\leq$ 120 AND Sale_Price > 220] is unsatisfiable and has no need to be performed.

Likewise, in the why-not-match explanation for event pattern queries, our pattern consistency explanation determines whether there exists an assignment of timestamps that can satisfy the given query patterns. Only for those consistent event patterns such as p_0 in Example 1, the timestamp (data) modification explanation (presented in Section 2.3 below) is meaningful.

Problem 1. *Given a set $\mathcal{P}$ of event patterns, the* pattern consistency problem *is to decide whether there exists a tuple t of events over $\mathcal{E}$ such that $t \vDash \mathcal{P}$.*

If not exist, i.e., $\mathcal{P}$ is inconsistent, it naturally explains why there is non-answer to the query with $\mathcal{P}$. Simple invalid patterns may be easily identified, such as ATLEAST 20 WITHIN 10, referring to the constraint $a \leq b$ in Definition 1. For the general case, however, the pattern consistency problem is hard.

Theorem 2. *The pattern consistency problem is NP-complete.*

Proof sketch. To show the hardness of the pattern consistency problem, we present a reduction from the 3SAT problem, which is one of Karp's 21 NP-complete problems [22]. Consider a CNF formula $C = c_1 \wedge \cdots \wedge c_m$, where each clause $c_i = l_{i1} \vee l_{i2} \vee l_{i3}$ have three literals, $i = 1, \ldots, m$, and each literal l_{ij} is either x or $\neg x$ for some variable $x \in X$, $n = |X|$. The 3SAT problem is to determine whether there exists an assignment to make all the clauses satisfied.

The transformation first introduces a set of events

$$\mathcal{E} = \{C_0, C_1, \ldots, C_m, X_1, \ldots, X_n, \neg X_1, \ldots, \neg X_n\},$$

where each C_i corresponds to a clause c_i, $i = 1, \ldots, m$, and $X_j, \neg X_j$ represent $x_j, \neg x_j$, respectively, $j = 1, \ldots, n$. A set $\mathcal{P}$ of event patterns over $\mathcal{E}$ are constructed as follows:
(1) For each $x_j, \neg x_j$, we introduce an event pattern

$$p_j : \texttt{SEQ}(C_0, \texttt{AND}(X_j, \neg X_j) \texttt{ ATLEAST 1 WITHIN 1}) \texttt{ ATLEAST 3 WITHIN 3}.$$

(2) For each clause $c_i = l_{i1} \vee l_{i2} \vee l_{i3}$, we add an event pattern

$$p_{n+i} : \texttt{SEQ}\big(C_i, \texttt{AND}(X_{i1}, X_{i2}, X_{i3})\big) \texttt{ ATLEAST 2 WITHIN 2}.$$

(3) For each C_i, we consider

$$\texttt{SEQ}(C_0, C_i) \texttt{ ATLEAST 1 WITHIN 1}.$$

Event pattern p_j requires that the timestamp distance between X_j and $\neg X_j$ is exactly 1, while p_{n+i} requires that at least one of X_{i1}, X_{i2}, X_{i3} should have timestamp distance 2 compared to C_i.

We show that the CNF formula in transformation is satisfiable if and only if there exists a tuple t of events such that $t[C_0] = s, t[C_i] = s + 1, t[X_{ik}] = s + 3$ for some $X_{ik}, k = 1, 2, 3, i = 1, \ldots, n$, s is any timestamp in the domain, that is, $t \vDash \mathcal{P}$, the event patterns are consistent. □

2.3 Timestamp Modification Explanation

In the why-not explanations for relational data queries, the data modification explanation [18] finds changes in data that would make a non-answer an answer. For instance, to figure out why Peter does not appear as a friend of John in the SQL query over relation Network, [SELECT U1_EMail, Friend, U2_Email FROM Network], Herschel and Hernández [18] provide a set of data modifications so that the missing tuples can present in the query result, such as modifying the incorrect U1_EMail of John in the relation.

Likewise, in the why-not-match explanation for event pattern queries, our timestamp modification explanation finds changes in timestamps of the event data that would satisfy the event pattern query, i.e., again make a non-answer an answer. For example, in order to figure out why the tuple t_2 in Table 1 is not returned in the event pattern query p_0 in Example 1, a possible timestamp modification explanation is to modify the imprecise timestamp of E_4 from 20:08 to 19:24.

Rather than considering all the possible worlds with an arbitrary combination of all timestamps [32], we propose to obtain the most likely modification explanation of imprecise timestamps to satisfy the query pattern, following the minimum change idea in data cleaning [13]. Let t' be the modified tuple of t. We define the modification cost as

$$\Delta(t, t') = \sum_{E_i \in \mathcal{E}} |t[E_i] - t'[E_i]|. \tag{1}$$

Problem 2. *Given a tuple t and a set $\mathcal{P}$ of event patterns over $\mathcal{E}$, the timestamp modification problem is to minimally modify the timestamps of events in order to make them match the pattern, i.e., $t' \vDash \mathcal{P}$ and $\Delta(t, t')$ is minimized.*

It explains why t is not returned as an answer to the query with $\mathcal{P}$. The reason is on the imprecise timestamps in t, that is, those ones different from t'. In other words, by modifying t by t', it will be an answer to the query with $\mathcal{P}$.

Theorem 3. *Given a set $\mathcal{P}$ of event patterns, a tuple t of events over $\mathcal{E}$, and an integer $k \geq 0$, the problem of determining whether there is a modification t' such that $t' \vDash \mathcal{P}$ with $\Delta(t, t') \leq k$ is NP-complete.*

Proof sketch. To illustrate the hardness of the timestamp modification problem, we show a reduction from the SET COVER problem, which is one of Karp's 21 NP-complete problems [22].

Given a set of m elements $\mathcal{U} = \{u_1, \ldots, u_m\}$ and n sets $\mathcal{S} = \{s_1, \ldots, s_n\}$ such that $s_i \subseteq \mathcal{U}$ and $\cup_i s_i = \mathcal{U}$. A set cover is a $C \subseteq \mathcal{S}$ of sets whose union is still $\mathcal{U}$. The minimum set cover problem is to identify the smallest number of sets whose union still contains all elements in $\mathcal{U}$.

Consider a set of events

$$\mathcal{E} = \{S_1, \ldots, S_i, \ldots, S_n, S'_1, \ldots, S'_n, U_1, \ldots, U_j, \ldots, U_m\}$$

where each S_i, S'_i represents a set s_i, and U_j corresponds to an element u_j. Let $\mathcal{E}_j = \{S_i \in \mathcal{E} \mid u_j \in s_i\}$, that is, all the events S_i whose corresponding s_i contains u_j.

First, the event pattern set $\mathcal{P}$ is constructed as follows.
(1) For each U_j, we add into $\mathcal{P}$ a pattern

$$p_j : \texttt{SEQ}(U_j, \texttt{AND}(S_{i1}, \ldots, S_{ik}))\ \texttt{ATLEAST 2 WITHIN 2},$$

where $S_{i1}, \ldots, S_{ik}$ correspond to all the sets $s_{i1}, \ldots, s_{ik} \subseteq \mathcal{S}$ containing element u_j.
(2) For each U_j, S'_i, we introduce a pattern

$$\texttt{SEQ}(S'_i, U_j)\ \texttt{ATLEAST 1 WITHIN 1}.$$

Next, we build a tuple t of events where $t[S'_i] = 0, t[U_j] = 1, t[S_i] = 2, i = 1, \ldots, n, j = 1, \ldots, m$. The patterns require that there must exist a $S_i \in \mathcal{E}_j$ such that $t[S_i] - t[U_j] = 2$, which is not observed in the given t.

To prove NP-hardness, we show that there is a set cover C of size k if and only if there exists a tuple t' matches the event patterns in the transformation and $\Delta(t, t') = k$. □

Theorem 4. *For any constant $\alpha > 1$, unless P=NP, there is no algorithm that approximates the problem of finding the minimum modification within a factor of α, and runs in polynomial time in the number of the input events.*

Due to space limitations, we prove Theorem 4 in the technical report [3]. The size of patterns, equivalent to query size, corresponds to the SAT or SET COVER size in the reductions. In practice, event patterns are often small. For the events in $\mathcal{E}$ not specified in the pattern, since no restrictions are declared, there is no need or help to consider or change them in the explanation. Thus, in the reductions, we assume that the set of events in patterns is the same as $\mathcal{E}$.

3 ENCODING TEMPORAL NETWORKS

Referring to the hardness of pattern consistency and timestamp modification explanations, in Theorems 2, 3 and 4, we thereby investigate techniques for efficient approximation. It is worth noting that a canonical form of simple temporal network (Definition 6) has been studied in [11], whose consistency checking can be done in polynomial time. Intuitively, we identify the special event patterns, i.e., without AND as illustrated in Table 2, which can be represented by simple temporal networks and thus efficiently solved. To further represent the AND semantics, we propose to extend simple temporal networks with binding conditions (Definition 4). Enlightened by the start and end points for composite events [9], we introduce two artificial events $\texttt{AND}^s_{p_0}$ and $\texttt{AND}^e_{p_0}$ for each AND pattern p_0, which bind the start and end of the AND pattern, respectively. By extending the aforesaid simple temporal networks with bindings, it forms a general form of complex temporal network (Definition 5) to represent event patterns with both (nested) SEQ and AND. The benefit of such an encoding is not only to identify the special cases that can be efficiently solved or approximated (Sections 4.1 and 5.1), but also to address the general case of pattern consistency checking and timestamp modification with regard to complex temporal networks, by enumerating the special cases (Algorithms 1 and 2), known as full or single binding (Definitions 7 and 8).

3.1 Interval and Binding Conditions

We first define two classes of interval conditions and binding conditions (in Definitions 3 and 4). Event patterns are then succinctly encoded by complex temporal networks, using only the aforesaid interval and binding conditions (in Section 3.2).

Definition 3. *An* interval condition *between two events E_i and E_j is in the form of $\phi(E_i, E_j)$:$[a, b]$, or simply $\phi(E_i, E_j)$, where a, b are the minimum and maximum restrictions of timestamp distances.*

We say that a tuple t *satisfies* the interval condition $\phi(E_i, E_j)$:$[a, b]$,

$$\big(t[E_i], t[E_j]\big) \vDash \phi(E_i, E_j)\text{:}[a, b] \text{ or simply } t \vDash \phi(E_i, E_j),$$

if the distance of timestamps on E_i and E_j satisfies $a \leq t[E_j] - t[E_i] \leq b$. A tuple t is said to satisfy a set Φ of interval conditions, $t \vDash \Phi$, if $t \vDash \phi(E_i, E_j), \forall \phi(E_i, E_j) \in \Phi$.

Definition 4. *A* binding condition *between an event E_i and an event set $\mathcal{E}_i$ is in the form of $\gamma(E_i, \mathcal{E}_i)$:min (or $\gamma(E_i, \mathcal{E}_i)$:max), where the timestamp of E_i should be equal to the minimum (or maximum) timestamp of events in $\mathcal{E}_i$.*

We say that a tuple t *satisfies* the minimum binding condition $\gamma(E_i, \mathcal{E}_i)$:*min*, denoted as

$$\big(t[E_i], t[\mathcal{E}_i]\big) \vDash \gamma(E_i, \mathcal{E}_i)\text{:}min, \text{ or simply } t \vDash \gamma(E_i, \mathcal{E}_i)\text{:}min,$$

if $t[E_i] = \min_{E_j \in \mathcal{E}_i} \pi(E_j)$. Likewise, it has $t \vDash \gamma(E_i, \mathcal{E}_i)$:*max*, if $t[E_i] = \max_{E_j \in \mathcal{E}_i} t[E_j]$. A tuple t is said to satisfy a set Γ of binding conditions, $t \vDash \Gamma$, if $t \vDash \gamma(E_i, \mathcal{E}_i), \forall \gamma(E_i, \mathcal{E}_i) \in \Gamma$.

3.2 Complex Temporal Networks

We now represent each event pattern p_0 as the succinct encoding $(\Phi_{p_0}, \Gamma_{p_0})$ of complex temporal networks (in Definition 5), and show their equivalence (in Proposition 5).

Let w be the maximum distance of two timestamps. Enlightened by the start and end points for composite events [9], we introduce two artificial events $\texttt{AND}^s_{p_0}$ and $\texttt{AND}^e_{p_0}$ for each AND pattern p_0, which denote the start and end of the AND pattern, respectively.

Definition 5 (Complex Temporal Networks)**.** *The temporal networks with regard to a pattern p_0 are defined as a tuple $(\Phi_{p_0}, \Gamma_{p_0})$ of interval and binding conditions, where*
(1) for $p_0 = E$, we have $\Phi_{p_0} = \emptyset$, $\Gamma_{p_0} = \emptyset$ and $E^s_{p_0} = E^e_{p_0} = E$, where $E^s_{p_0}$ and $E^e_{p_0}$ are the start and end events of p_0, corresponding to the conditions (1) in Definitions 1 and 2;

(2) for $p_0 = \texttt{SEQ}(p_1, p_2, \ldots, p_k)$ [ATLEAST a] [WITHIN b],

$$\Phi_{p_0} = \{\phi(E^e_{p_1}, E^s_{p_2}):[0, w], \ldots, \phi(E^e_{p_{k-1}}, E^s_{p_k}):[0, w]\} \cup \Phi_{p_1} \cup \cdots \cup \Phi_{p_k} \big[\cup \{\phi(E^s_{p_0}, E^e_{p_0}):[a, b]\}\big]$$

where $E^s_{p_0} = E^s_{p_1}, E^e_{p_0} = E^e_{p_k}$, *corresponding to the conditions (2) on* SEQ *pattern in Definitions 1 and 2;*
(3) for $p_0 = \texttt{AND}(p_1, p_2, \ldots, p_k)$ [ATLEAST a] [WITHIN b],

$$\begin{aligned}\Phi_{p_0} =& \{\phi(E^s_{p_0}, E^s_{p_1}):[0, w], \phi(E^e_{p_1}, E^e_{p_0}):[0, w], \ldots, \\ & \phi(E^s_{p_0}, E^s_{p_k}):[0, w], \phi(E^e_{p_k}, E^e_{p_0}):[0, w]\} \\ & \cup \Phi_{p_1} \cup \cdots \cup \Phi_{p_k} \big[\cup \{\phi(E^s_{p_0}, E^e_{p_0}):[a, b]\}\big], \\ \Gamma_{p_0} =& \{\gamma(E^s_{p_0}, \{E^s_{p_1}, \ldots, E^s_{p_k}\}):min, \\ & \gamma(E^e_{p_0}, \{E^e_{p_1}, \ldots, E^e_{p_k}\}):max\} \cup \Gamma_{p_1} \cup \cdots \cup \Gamma_{p_k}\end{aligned}$$

where $E^s_{p_0} = \texttt{AND}^s_{p_0}, E^e_{p_0} = \texttt{AND}^e_{p_0}$, *corresponding to the conditions (3) on* AND *pattern in Definitions 1 and 2.*

Referring to the recursive definitions of satisfaction in Definition 2 and the encoding in Definition 5, it is easy to see the equivalence of satisfaction between event patterns and the corresponding complex temporal networks.

Proposition 5. *A tuple t satisfies p_0, $t \vDash p_0$, iff the corresponding $(\Phi_{p_0}, \Gamma_{p_0})$ of p_0 have $t \vDash \Phi_{p_0}$ and $t \vDash \Gamma_{p_0}$, denoted by $t \vDash (\Phi_{p_0}, \Gamma_{p_0})$.*

Due to space limitations, we prove Proposition 5 in the technical report [3].

Example 2 (Example 1 continued)**.** Figure 1 illustrates the complex temporal network for representing the event pattern p_0 in Example 1 for monitoring COVID-19 cases. The motivation is that some special event patterns can be represented by a canonical form of simple temporal networks [11], in Example 3 below, and efficiently solved. More importantly, for the general case, the complex temporal network such as p_0 can be equivalently represented by a set of simple temporal networks and solved (Examples 4 and 6).

Specifically, each event corresponds to a node, such as E_1, referring to item 1 in Definition 5. For the nested pattern p_1 : $\texttt{AND}(E_1, E_3)$ WITHIN 30 minutes, two artificial events $\texttt{AND}^s_{p_1}$ and $\texttt{AND}^e_{p_1}$ are added, which denote the start and end points of the AND pattern, respectively. Its interval conditions are $\phi(\texttt{AND}^s_{p_1}, E_1):[0, w], \ldots, \phi(E_3, \texttt{AND}^e_{p_1}):[0, w]$ in Φ_{p_1}, and the binding conditions are $\gamma(\texttt{AND}^s_{p_1}, \{E_1, E_3\}):min$, $\gamma(\texttt{AND}^e_{p_1}, \{E_1, E_3\}):max$ in Γ_{p_1}, according to item 3 in Definition 5. Another interval condition $\phi(\texttt{AND}^s_{p_1}, \texttt{AND}^e_{p_1}):[0, 30$ minutes$]$ is added referring to WITHIN 30 minutes in p_1. The other nested pattern p_2 : $\texttt{AND}(E_2, E_4)$ WITHIN 30 minutes is represented similarly. For SEQ in p_0, according to item 2 in Definition 5, we add interval conditions $\phi(\texttt{AND}^e_{p_1}, \texttt{AND}^s_{p_2}):[0, w]$, in Φ_{p_0}, together with $\phi(\texttt{AND}^s_{p_1}, \texttt{AND}^e_{p_2}):[2$ hours$, w]$ referring to ATLEAST 2 hours of p_0.

For tuple $t_1 \vDash p_0$ in Table 1, according to the binding conditions in item 3 in Definition 5, we have $t_1[\texttt{AND}^s_{p_1}] = \min(t_1[E_1], t_1[E_3]) =$ 17 : 08, and $t_1[\texttt{AND}^e_{p_1}] = \max(t_1[E_1], t_1[E_3]) =$ 17 : 25. For each interval condition in Φ_{p_0}, say $\phi(\texttt{AND}^s_{p_1}, \texttt{AND}^e_{p_1})$, it has $(t_1[\texttt{AND}^s_{p_1}], t_1[\texttt{AND}^e_{p_1}]) \vDash \phi(\texttt{AND}^s_{p_1}, \texttt{AND}^e_{p_1}):[0, 30$ minutes$]$, namely, the time difference is within 30 minutes. In other words, $t_1 \vDash (\Phi_{p_0}, \Gamma_{p_0})$. □

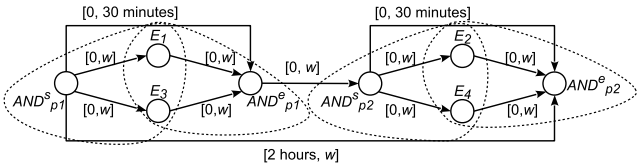

Figure 1: Example complex temporal network with interval conditions (arrows) and binding conditions (in dashed line)

4 PATTERN CONSISTENCY EXPLANATION

Referring to the equivalence of event patterns and complex temporal networks (Proposition 5), we rewrite the pattern consistency Problem 1: Given (Φ, Γ) of interval and binding conditions for event patterns, to decide whether there is a tuple t such that $t \vDash (\Phi, \Gamma)$.

4.1 Special Case: Simple Temporal Networks

We identify an interesting category of event patterns, where the AND predicate does not appear (i.e., case 3 in Table 2). As illustrated in Definition 6, the corresponding temporal networks are indeed simple temporal networks [11] that can be represented by interval conditions Φ only.

Definition 6 (Simple Temporal Networks)**.** *The temporal networks of a pattern p_0 without* AND *predicates are defined as Φ_{p_0}, where*
(1) for $p_0 = E$, $\Phi_{p_0} = \emptyset$ and $E^s_{p_0} = E^e_{p_0} = E$, where $E^s_{p_0}$ and $E^e_{p_0}$ represent the start and end events of p_0, corresponding to the conditions (1) in Definitions 1 and 2;
(2) for $p_0 = \texttt{SEQ}(p_1, p_2, \ldots, p_k)$ [ATLEAST a] [WITHIN b],

$$\Phi_{p_0} = \{\phi(E^e_{p_1}, E^s_{p_2}):[0, w], \ldots, \phi(E^e_{p_{k-1}}, E^s_{p_k}):[0, w]\} \cup \Phi_{p_1} \cup \cdots \cup \Phi_{p_k} \big[\cup \{\phi(E^s_{p_0}, E^e_{p_0}):[a, b]\}\big]$$

where $E^s_{p_0} = E^s_{p_1}, E^e_{p_0} = E^e_{p_k}$, corresponding to the conditions (2) on SEQ *pattern in Definitions 1 and 2;*

Corollary 6. *A tuple satisfies $t \vDash p_0$, iff the corresponding Φ_{p_0} of p_0 has $t \vDash \Phi_{p_0}$.*

Due to space limitations, we prove Corollary 6 in the technical report [3].

Example 3 (Example 1 continued)**.** Suppose that some passengers are known to arrive later than E_1 and also depart later than E_2 (for other cases, please see Examples 4 and 6 below). For such a special case, the event pattern query could be simply written as

$$\texttt{SEQ}(\texttt{SEQ}(E_1, E_3)\ \texttt{WITHIN}\ 30\ \text{minutes},\ \texttt{SEQ}(E_2, E_4)\ \texttt{WITHIN}\ 30\ \text{minutes})\ \texttt{ATLEAST}\ 2\ \text{hours}$$

where E_3 is the arrival time and E_4 is the departure time of the traced passenger. Figure 2 illustrates the corresponding simple temporal network referring to Definition 6. Since there is no AND, we do not need to introduce the artificial events, and the binding conditions denoted by dashed lines are naturally absent. We only need 4 interval conditions to represent the pattern, $\phi(E_1, E_3):[0, 30$ minutes$]$, $\phi(E_3, E_2):[0, w]$, $\phi(E_2, E_4):[0, 30$ minutes$]$, and $\phi(E_1, E_4):[2$ hours$, w]$. It is easy and efficient to check the consistency of the simple temporal network by [11]. □

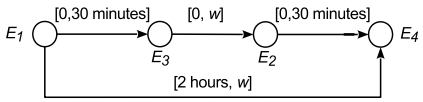

Figure 2: Example simple temporal network

As indicated in [11], the consistency problem for a set of simple temporal networks Φ is solvable in $O(n^3)$ time, where n is the number of events $n = |\mathcal{E}|$.

4.2 General Case

In addition to the interval conditions Φ, the general case needs to further handle the binding conditions Γ. Intuitively, we consider all the possible bindings for binding conditions in Γ (Definition 7). The pattern consistency problem is thus to determine whether there exists one binding with feasible timestamps of events (Proposition 7).

First, we capture below all the possible bindings in a binding condition γ by the disjunction of interval conditions.

Definition 7 (Full binding)**.** *For a binding condition in Γ in the form of $\gamma(E_i, \mathcal{E}_i)$:min or $\gamma(E_i, \mathcal{E}_i)$:max, we study a set Φ_γ of interval conditions $\Phi_\gamma = \{\phi(E_i, E_j)\text{:}[0, 0] \mid E_j \in \mathcal{E}_i\}$.*

Let $\aleph_\Gamma = \Phi_{\gamma_1} \times \cdots \times \Phi_{\gamma_{|\Gamma|}}$, where $\gamma_1, \ldots, \gamma_{|\Gamma|} \in \Gamma$. It denotes all the possible bindings with regard to Γ.

Proposition 7. *A tuple (Φ, Γ) of interval and binding conditions with regard to event pattern is consistent, if and only if there exists at least one $\Phi_k \in \aleph_\Gamma$ such that $\Phi \cup \Phi_k$ is consistent.*

Due to space limitations, we prove Proposition 7 in the technical report [3]. Referring to Proposition 7, Algorithm 1 determines the consistency of (Φ, Γ), by examining the consistency on each $\Phi_k \in \aleph_\Gamma$ together with Φ in Line 2.

Algorithm 1 Consistency(Φ, Γ)

Input: A tuple (Φ, Γ) of interval and binding conditions
Output: Whether (Φ, Γ) is consistent (true or false)
1: **for** each $\Phi_k \in \aleph_\Gamma$ **do**
2: **if** $\Phi \cup \Phi_k$ is consistent **then**
3: **return** true
4: **return** false

Let f be the average size of $\mathcal{E}_i$ in the binding conditions in Γ. That is, the average size of full binding Φ_γ in Definition 7 is f. The total number of Φ_k in $\aleph_\Gamma$ is thus $O(f^{|\Gamma|})$. As presented at the end of Section 4.1, the pattern consistency problem for a set of simple temporal networks $\Phi \cup \Phi_k$ is solvable in $O(n^3)$ time, where n is the number of events. Therefore, Algorithm 1 runs in $O(f^{|\Gamma|} n^3)$ time.

Example 4 (Example 2 continued)**.** While Example 3 considers a special case, the passengers may arrive or depart either earlier or later in the general case. According to Definition 7 and Proposition 7, we can equivalently represent the complex temporal network in Example 2 by a set of simple temporal networks, including the one presented in Example 3. Specifically, consider all the binding conditions in Example 2. We have $\aleph_\Gamma = \Phi_{\gamma_1} \times \Phi_{\gamma_2} \times \Phi_{\gamma_3} \times \Phi_{\gamma_4}$, where $\Phi_{\gamma_1} = \{\phi(\mathsf{AND}^s_{p_1}, E_1)\text{:}[0, 0], \phi(\mathsf{AND}^s_{p_1}, E_3)\text{:}[0, 0]\}$, $\Phi_{\gamma_2} = \{\phi(\mathsf{AND}^e_{p_1}, E_1)\text{:}[0, 0], \phi(\mathsf{AND}^e_{p_1}, E_3)\text{:}[0, 0]\}$, $\Phi_{\gamma_3} = \{\phi(\mathsf{AND}^s_{p_2}, E_2)\text{:}[0, 0], \phi(\mathsf{AND}^s_{p_2}, E_4)\text{:}[0, 0]\}$, and $\Phi_{\gamma_4} = \{\phi(\mathsf{AND}^e_{p_2}, E_2)\text{:}[0, 0], \phi(\mathsf{AND}^e_{p_2}, E_4)\text{:}[0, 0]\}$. Algorithm 1 solves the pattern consistency checking problem by iteratively considering all the possible bindings $\Phi_k \in \aleph_\Gamma$ in Line 1. For $\Phi_1 = \{\phi(\mathsf{AND}^s_{p_1}, E_1)\text{:}[0, 0], \phi(\mathsf{AND}^e_{p_1}, E_3)\text{:}[0, 0], \phi(\mathsf{AND}^s_{p_2}, E_2)\text{:}[0, 0], \phi(\mathsf{AND}^e_{p_2}, E_4)\text{:}[0, 0]\}$ in $\aleph_\Gamma$, we have consistent $\Phi_{p_0} \cup \Phi_1$. The binding Φ_1 corresponds to exactly the simple temporal network in Figure 2 in Example 3. Therefore, the algorithm returns true in Line 3 without considering the remaining Φ_k in $\aleph_\Gamma$. □

Randomized Algorithm. Rather than considering each $\Phi_k \in \aleph_\Gamma$ in Line 1 in Algorithm 1, we sample Φ_k from $\aleph_\Gamma$ by randomly drawing binding from Φ_γ in Definition 7 (with the same probability) for each binding condition γ in Γ. If all the s random samples Φ_k have inconsistent $\Phi \cup \Phi_k$, we heuristically return inconsistency on (Φ, Γ). It is notable that the randomized algorithm never returns false positive (FP), i.e., return consistent but the truth is inconsistent.

5 TIMESTAMP MODIFICATION EXPLANATION

Again, using the equivalent complex temporal networks of event patterns (Proposition 5), we rewrite the timestamp modification Problem 2: Given a tuple t, the interval and binding conditions (Φ, Γ), to minimally modify the timestamps of events such that $t' \vDash (\Phi, \Gamma)$ and $\Delta(t, t')$ is minimized.

5.1 Special Case: Simple Temporal Networks

For the special simple temporal networks, we solve the timestamp modification problem (in Formula 3) as integer linear programming (ILP). Efficient LP-relaxation can thus be applied.

For a set Φ of simple temporal networks, we rewrite the timestamp modification problem as:

$$\begin{aligned} \min \quad & \sum_{E_i \in \mathcal{E}} |t[E_i] - t'[E_i]| && && (2)\\ \text{s.t.} \quad & t'[E_j] - t'[E_i] \geq a && , \quad \phi(E_i, E_j)\text{:}[a, b] \in \Phi \\ & t'[E_j] - t'[E_i] \leq b && , \quad \phi(E_i, E_j)\text{:}[a, b] \in \Phi \\ & t'[E_i] \in T && , \quad E_i \in \mathcal{E} \end{aligned}$$

referring to the modification cost function in Formula 1 and the interval conditions in Definition 3. It can be formulated as integer linear programming (ILP). Let

$$u_i = \frac{|t[E_i] - t'[E_i]| + t[E_i] - t'[E_i]}{2} \quad ,$$
$$v_i = \frac{|t[E_i] - t'[E_i]| - t[E_i] + t'[E_i]}{2} \quad ,$$

having $|t[E_i] - t'[E_i]| = u_i + v_i$ and $t[E_i] - t'[E_i] = u_i - v_i$.

The timestamp modification problem is rewritten again as:

$$\begin{aligned} \min \quad & \sum_{E_i \in \mathcal{E}} u_i + v_i && (3)\\ \text{s.t.} \quad & u_i - v_i + v_j - u_j - t[E_i] + t[E_j] \geq a, \phi(E_i, E_j)\text{:}[a, b] \in \Phi \\ & u_i - v_i + v_j - u_j - t[E_i] + t[E_j] \leq b, \phi(E_i, E_j)\text{:}[a, b] \in \Phi \\ & u_i, v_i \geq 0, u_i, v_i \in T, E_i \in \mathcal{E} && (4) \end{aligned}$$

where the modified timestamp is $t'[E_i] = t[E_i] - u_i + v_i$.

Efficient LP-relaxation can be employed by removing $u_i, v_i \in T$ in Formula 4, with time cost $O(n^{3.5}L)$, where n is event number and L is the number of bits of input [21].

Example 5 (Example 3 continued)**.** Consider again the simple temporal network in Figure 2 and the tuple t_2 in Table 1. The timestamp modification explanation of t_2 for the simple temporal network is solved as follows. According to Formula 2, the conditions (the time unit is minute) in the rewritten problem are: $t_2'[E_3] - t_2'[E_1] \le 30$, $t_2'[E_3] - t_2'[E_1] \ge 0$, $t_2'[E_2] - t_2'[E_3] \ge 0$, $t_2'[E_4] - t_2'[E_2] \le 30$, $t_2'[E_4] - t_2'[E_2] \ge 0$, $t_2'[E_4] - t_2'[E_1] \ge 2 * 60$. Referring to the transformation, we have

$$u_1 = \frac{|t_2[E_1] - t_2'[E_1]| + t_2[E_1] - t_2'[E_1]}{2},$$
$$v_1 = \frac{|t_2[E_1] - t_2'[E_1]| - t_2[E_1] + t_2'[E_1]}{2}, \quad \dots$$

The corresponding transformed conditions are $u_1 - v_1 + v_3 - u_3 - (17 * 60 + 6) + (17 * 60 + 24) \le 30 \dots$ By solving the transformed problem (using LP solvers), the result is $t_2'[E_1]$=17:06, $t_2'[E_2]$=18:54, $t_2'[E_3]$=17:24, $t_2'[E_4]$=19:24, with modification cost 44 minutes. □

5.2 General Case

Similar to Section 4.2, for the binding conditions Γ not considered in the aforesaid special case, first, we capture all the possible bindings in a binding condition γ by the disjunction of interval conditions. The timestamp modification problem in the general case is thus to find the timestamp modification explanation having the minimum modification cost with regard to all the possible bindings. A natural approximation idea is to consider only the most likely binding (Definition 8) rather than all. We show that such a binding (1) practically obtains good explanations with relatively small modification cost in general cases (see experiments in Section 6), and (2) theoretically guarantees the explanations with optimal modification cost in certain cases (Proposition 8).

5.2.1 Full Binding Algorithm. For each binding $\Phi_k \in \aleph_\Gamma$ as defined in Section 4.2, we find the timestamp modification with minimum cost under the simple temporal networks $\Phi \cup \Phi_k$, denoted by $t_k^\# = \text{ILP}(t, \Phi \cup \Phi_k)$. The optimal explanation is $t^* = \arg\min_{t_k^\#} \Delta(t, t_k^\#)$.

Algorithm 2 MODIFICATION(t, Φ, Γ)

Input: A tuple t and a pair (Φ, Γ) of interval and binding conditions
Output: A timestamp modification explanation t'

```
1: for each Φ_k ∈ ℵ_Γ do
2:     t_k^# = ILP(t, Φ ∪ Φ_k)
3:     if Δ(t, t_k^#) < Δ(t, t') then
4:         t' := t_k^#
5: return t'
```

Let f be the average size of $\mathcal{E}_i$ in the binding conditions in Γ, that is, the average size of full binding Φ_γ in Definition 7. The number of Φ_k in $\aleph_\Gamma$ is $O(f^{|\Gamma|})$. Algorithm 2 runs in $O(f^{|\Gamma|} n^{3.5} L)$ time, where n is the number of events and L is the number of bits of input.

Example 6 (Example 5 continued)**.** Consider again tuple t_2 in Table 1 and the complex temporal networks in Figure 1 of Example 2. As presented in Example 4, there are 4 possible bindings in $\aleph_\Gamma$. For $\Phi_1 = \{\phi(\text{AND}^s_{p_1}, E_1)$:[0, 0], $\phi(\text{AND}^e_{p_1}, E_3)$:[0, 0], $\phi(\text{AND}^s_{p_2}, E_2)$: [0, 0], $\phi(\text{AND}^e_{p_2}, E_4)$:[0, 0]$\}$ in $\aleph_\Gamma$, the timestamp modification explanation for $\Phi_{p_0} \cup \Phi_1$ is $t_1^\#[E_1]$=17:06,$t_1^\#[E_2]$=18:54,$t_1^\#[E_3]$=17:24,$t_1^\#[E_4]$=19:24 with modification cost 44 minutes. Similarly, for $\Phi_2 = \{\phi(\text{AND}^s_{p_1}, E_1)$: [0, 0], $\phi(\text{AND}^e_{p_1}, E_3)$:[0, 0], $\phi(\text{AND}^s_{p_2}, E_4)$:[0, 0], $\phi(\text{AND}^e_{p_2}, E_2)$:[0, 0]$\}$ in $\aleph_\Gamma$, the explanation is $t_2^\#[E_1]$=17:06, $t_2^\#[E_2]$ =19:06, $t_2^\#[E_3]$=17:24,$t_2^\#[E_4]$= 19:06 with modification cost 74 minutes. By considering all the bindings in $\aleph_\Gamma$ in Line 1, Algorithm 2 solves the timestamp modification problem by returning $t_1^\#$ as the optimal explanation with the minimum modification cost. □

5.2.2 Single Binding Approximation. Rather than full binding in Definition 7, we heuristically consider the most likely bindings for approximation, e.g., binding only the event E with the minimum timestamp $t[E]$ for *min* binding conditions, and similarly the maximum one for *max* binding conditions.

Definition 8 (Single binding)**.** *For any binding condition in Γ in the form of $\gamma(E_i, \mathcal{E}_i)$:min, we consider one binding $\Phi_\gamma = \{\phi(E_i, E_j)$:[0, 0]$\}$, where $E_j = \arg\min_{E_j \in \mathcal{E}_i} t[E_j]$.*

A similar definition applies to $\gamma(E_i, \mathcal{E}_i)$:*max*, having one binding $\Phi_\gamma = \{\phi(E_i, E_j)$:[0, 0]$\}$, where $E_j = \arg\max_{E_j \in \mathcal{E}_i} t[E_j]$. Instead of $O(f^{|\Gamma|})$ possible bindings, only one iteration will be performed in Line 1 in Algorithm 2.

The modification with this single binding indeed obtains the optimal explanation when given AND patterns without SEQ predicate embedded, i.e., case 2 in Table 2.

Proposition 8. *For a tuple (Φ, Γ) of temporal networks corresponding to an event pattern without* SEQ *embedded in* AND*, that is, in the form of* AND$(E_1, \dots, E_n)$*, Algorithm 2 with single binding, having only one Φ_1 in $\aleph_\Gamma$, returns the optimal explanation for full binding.*

Due to space limitations, we prove Proposition 8 in the technical report [3]. When efficient LP-relaxation is employed in Line 2, Algorithm 2 with single binding runs in $O(n^{3.5}L)$ time, where n is the number of events and L is the number of bits of input.

6 EXPERIMENTS

In this section, we evaluate the performances of our proposals, on both pattern consistency and timestamp modification explanations. All programs are implemented in Java, and the experiment runs on a computer with 2.4GHz CPU and 16GB memory. The code and data are available at [4].

6.1 Implementation

Figure 3 illustrates the system structure for explaining non-answers to event pattern queries. It takes the event pattern query and the desired tuple that does not appear in the query answer as the input. (1) The event patterns will first be encoded as complex temporal networks as presented in Section 3. (2) The pattern consistency explanation is then performed by Algorithm 1 in Section 4.2. It utilizes (in Line 2) the consistency checking of simple temporal networks presented in Section 4.1. If the given event patterns are found to be inconsistent, it is directly returned as the explanation of non-answers. (3) Otherwise, the timestamp modification explanation is conducted by Algorithm 2 in Section 5.2. Again, as presented in Line 2, the ILP solution for simple temporal networks in

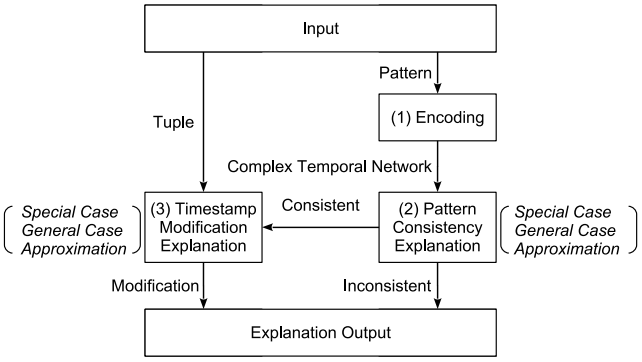

Figure 3: System structure

Section 5.1 is utilized. The modification of the given tuple will be returned as the timestamp modification explanation.

6.2 Pattern Consistency Explanation

To evaluate the methods proposed in Section 4 for providing query pattern explanations, i.e., checking pattern consistency, we employ 20 pattern sets as illustrated in Figure 4. Each pattern set has an AND (in solid lines)

$$\texttt{AND}(\texttt{SEQ}(E_{11}, E_{12}), \texttt{SEQ}(E_{13}, E_{14}), \ldots, \texttt{SEQ}(E_{n1}, E_{n2}), \texttt{SEQ}(E_{n3}, E_{n4}))\ \texttt{ATLEAST 1 WITHIN}\ b,$$

and a number of SEQ conditions (in dashed lines), including

$$\texttt{SEQ}(E_{11}, E_{14})\ \texttt{ATLEAST 0 WITHIN 0}, \ldots,$$
$$\texttt{SEQ}(E_{n1}, E_{n4})\ \texttt{ATLEAST 0 WITHIN 0}.$$

It states that the timestamp distances are at least 1 between E_{11} and E_{12}, as well as between E_{13} and E_{14}. As illustrated in Figure 4, E_{11} and E_{14} have the same timestamp which means that timestamp distance between E_{12} and E_{13} is at least 2. For the first 10 pattern sets, we have $b = 1$, for $n = 1, \ldots, 10$. It implies that E_{12} and E_{13} should have timestamp distance WITHIN $b = 1$. That is, these 10 pattern sets are inconsistent. For the other 10 pattern sets, we have $b = 2$, for $n = 1, \ldots, 10$, which are consistent. For $n = 1$, the event pattern set is defined by an AND pattern

$$p_1 : \texttt{AND}(\texttt{SEQ}(E_{11}, E_{12})\ \texttt{ATLEAST 1}, \texttt{SEQ}(E_{13}, E_{14})\ \texttt{ATLEAST 1})\ \texttt{ATLEAST 1 WITHIN}\ b,$$

and a SEQ pattern $p_2 : \texttt{SEQ}(E_{11}, E_{14})\ \texttt{ATLEAST 0 WITHIN 0}$. Different from the series-parallel graphs [28], the predicates WITHIN 1 and ATLEAST b in the event patterns further specify the constraints on timestamp distances of events.

Different binding strategies are compared, by considering various number of random samples Φ_k from all the possible bindings in $\aleph_\Gamma$, as shown in Section 4.2. The Full binding approach is the exact algorithm considering all the possible bindings, while the other randomized binding methods, e.g., 4-binding, consider a number of 4 random samples Φ_k in $\aleph_\Gamma$. Since the randomized binding never returns false positive (FP), i.e., return consistent but the truth is inconsistent, the returned results of the randomized algorithm consist of true positives (TP), true negatives (TN), and false negatives (FN). Compared to the exact answers of Full binding, we study the accuracy of Randomized binding $\frac{\#TP+\#TN}{\#TP+\#TN+\#FN}$.

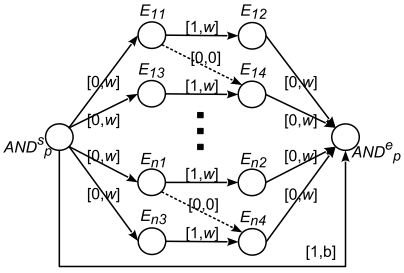

Figure 4: Pattern set for consistency evaluation

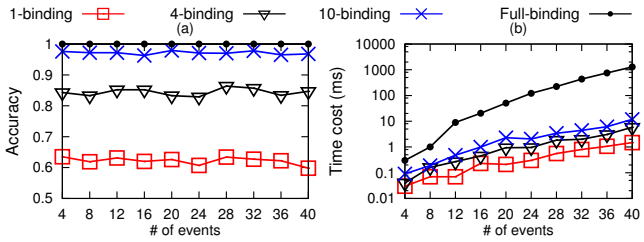

Figure 5: Varying the number of events in a pattern set in consistency evaluation

As shown in Figure 5, the time cost increases with the number of events in a pattern set. The consistency checking accuracy of various randomized binding strategies is generally stable for different pattern sets. The more the random samples are considered, e.g., 10-binding, the higher the accuracy is. The corresponding time cost is higher as well. Nevertheless, by only considering a number of 10 random samples Φ_k in $\aleph_\Gamma$, the consistency checking accuracy is already close to 1 (the exact result with Full binding), while the 10-binding approach shows about 2 orders of magnitude improvement in time cost compared to the Full binding.

In short, the method considering a number of bindings provides a trade-off between checking accuracy (best by Full binding) and efficiency (1-binding wins).

6.3 Timestamp Modification Explanation

To evaluate the methods proposed in Section 5 for timestamp modification explanations, we employ two real datasets, (1) the Flight data [2] with real-world imprecise timestamps naturally embedded, (2) the RTFM data [1] by synthetically introducing incorrect timestamp, and (3) the synthetic data for various special cases.

The approaches in comparison are (1) our timestamp modification Algorithm 2 with LP-relaxation, including Pattern(Full) using full binding in Definition 7 and Pattern(Single) using single binding in Definition 8. (2) The brute-force algorithm adapts [32] by enumerating the timestamps of each event as possible worlds and returning the one with the minimum change as explanation. (3) The greedy algorithm iteratively checks the interval condition between two events, and greedily selects the minimum change timestamp for either of the events if they do not satisfy the condition. It

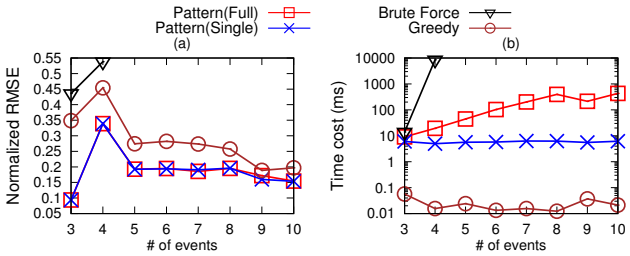

Figure 6: Varying the number of events in Flight data with real-world imprecise timestamps and labeled truth

is worth noting that the greedy method cannot guarantee to find a modification explanation that satisfies the event pattern query.

We report the normalized root-mean-square error (NRMSE) [26] between the truth timestamp t^* and the modification t' as NRMSE $= \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(t'[E_i]-t^*[E_i])^2}}{\frac{1}{n}\sum_{i=1}^{n}|t^*[E_i]|}$, where n is the number of events in the patterns. The smaller the NRMSE is, the closer the modification t' is to the truth t^*, and thus the better explanation for non-answer.

Apart from the streaming scenario of querying events today, interactive query over historical data is also the target scenario. As discussed in the Introduction, event pattern queries on the flights in the past weeks are conducted to study the spread of COVID-19 cases. A lower time cost (in milliseconds or seconds rather than hours or days) is necessary for query debugging and interactive analysis. Thereby, we evaluate the time cost of explanations.

6.3.1 Flight Dataset with Real-World Imprecise Timestamps. Flight dataset [2] collects the departure and arrival timestamps of flights in a month from different sources. The whole set of flights for a day is a single tuple, i.e., one tuple per day. The original datasets have replicated events coming from data sources. There may exist multiple versions of the same event from heterogeneous data sources, some of which could be imprecise (with labeled truth in the dataset). We randomly select one of the timestamps for each event in a tuple. Owing to the imprecise timestamps, expected answers may not be returned and need explanation.

The query patterns are from real-world scenarios. For instance, we use the pattern SEQ(EWR, MCO) ATLEAST 120 WITHIN 200 in the Flight dataset. It specifies a query on the flights from EWR to MCO without a stopover, taking at least 120 minutes but at most 200 minutes. All the tuples of flights from EWR to MCO should be returned. If there is a missing answer, it should be explained why.

Figure 6 presents the results on various number of (departure or arrival) events for flights from EWR to MCO. We do not consider a very large number of events in a query, since the number of events in real business processes (event patterns) is often bounded, according to the survey [29]. Indeed, flights in different days (tuples) do not affect each other and can be considered separately. In Figure 6(a), Pattern(Single) achieves modification accuracy comparable to Pattern(Full) considering full binding, while the corresponding time cost of Pattern(Single) is much lower in Figure 6(b).

For the brute-force algorithm, we enumerate possible timestamps in a unit of 10 minutes. Even with such a coarse-grained level, the time costs are too high to obtain an explanation with more than 5

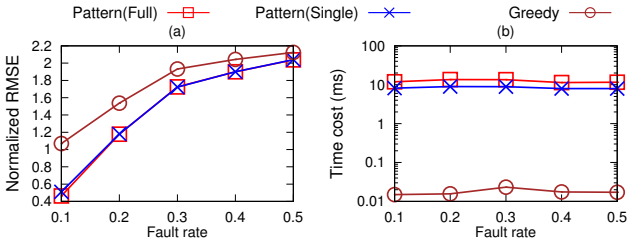

Figure 7: Varying fault rate over real RTFM data, with fault distance 200, tuple number 10k

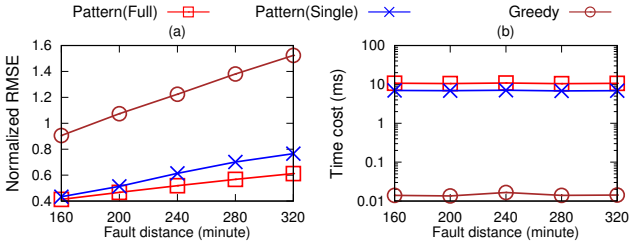

Figure 8: Varying fault distance over real RTFM data, with fault rate 0.1, tuple number 10k

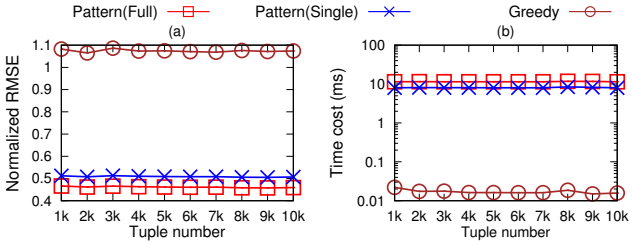

Figure 9: Varying tuple number over real RTFM data, with fault rate 0.1, fault distance 200

events in Figure 6. The corresponding explanation RMSE is high as well, again owing to the coarse-grained level of timestamp enumeration. It is not surprising that the simple greedy algorithm has the lowest time cost, while its modification explanation accuracy is not comparable to Pattern(Full) and Pattern(Single).

6.3.2 RTFM Dataset with Synthetic Imprecise Timestamps. Tuples of activities are collected in this dataset with timestamps from the Road Traffic Fine Management (RTFM) process [1]. Since it is originally clean, we follow the same line of randomly generating errors for benchmarking data cleaning algorithms [7], that is, randomly modify the timestamps of events to introduce synthetic errors. A fault rate, say 0.1, denotes that the probability of an event having fault timestamp introduced is 0.1. A fault distance, e.g., 200, means that the fault timestamp is a random number $t \pm 200$, where t is the original timestamp without fault introduced. The larger the fault rate and the fault distance are, the more likely the tuples originally matching the query patterns become non-answers. It is also more difficult to explain by timestamp modification (in Figures 7 and 8).

We extract event patterns from clean data as queries and manually confirm them, e.g., AND(Payment, Add penalty) ATLEAST 10 WITHIN 480. It states that "Add penalty" may occur 10 minutes before or after "Payment", while their timestamp distance should not exceed 480 minutes, i.e., 8 working hours in a day.

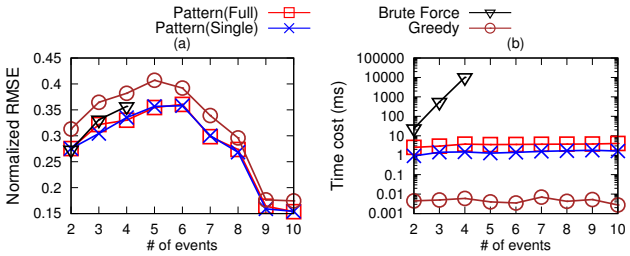

Figure 10: Varying event number n in AND with embedded SEQ, AND(SEQ($E_1, \ldots, E_{\lfloor \frac{n}{2} \rfloor}$), SEQ($E_{\lfloor \frac{n}{2} \rfloor+1}, \ldots, E_n$)) ATLEAST 900 WITHIN 1000

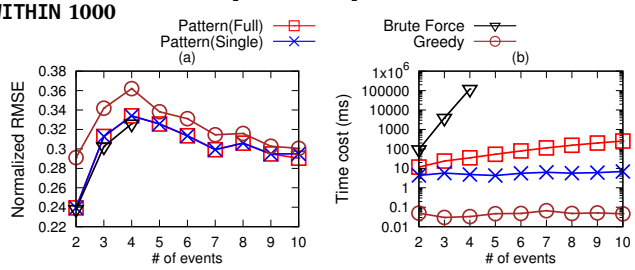

Figure 11: Varying event number n of AND conditions without SEQ embedded, AND($E_1, \ldots, E_n$) ATLEAST 900 WITHIN 1000

Figure 7 presents the results over various fault rates. As shown in Figure 7(a), it is not surprising that the RMS error increases with the increase of fault rate. The results are generally similar to those on Flight data with real imprecise timestamps in Figure 6. That is, Pattern(Single) has lower time cost than Pattern(Full) without losing much accuracy. While the greedy algorithm has a low time cost but much higher RMSE, the brute-force algorithm takes too long to obtain an explanation in the RTFM dataset and thus is omitted.

Figure 8 reports the results with various fault distances. RMS error increases with the increase of fault distance. It is generally similar to those over various fault rates in Figure 7.

Figure 9 shows the average RMS error and time cost over various numbers of tuples. The results are generally the same as in the aforesaid experiments. The stable results demonstrate that the approaches are applicable to a large set or streaming traces of events.

6.3.3 Synthetic Data with Various Conditions. To evaluate timestamp modification explanation under various event patterns as illustrated in Table 2, we consider synthetic data, including (1) general case with both AND and SEQ in Figure 10, and (2) special case with no SEQ embedded in Figure 11 Tuples are generated by randomly assigning timestamps to events, which match the aforesaid event pattern. Similar to the RTFM data in Section 6.3, we again randomly introduce incorrect timestamps. The experiments are performed over 1k tuples with fault rate 0.4 and fault distance 500. Similar to Figure 6 over dataset Flight, the brute-force algorithm can return an explanation only when given a small number of events in the synthetic data.

Figure 10 considers the general queries with both AND and SEQ(i.e., case 1 in Table 2). Pattern(Full) and Pattern(Single) achieve a similar RMS error. An interesting result is that the time cost of Pattern(Full) does not increase heavily with the increase of event number. The reason is that the number of events specified in binding

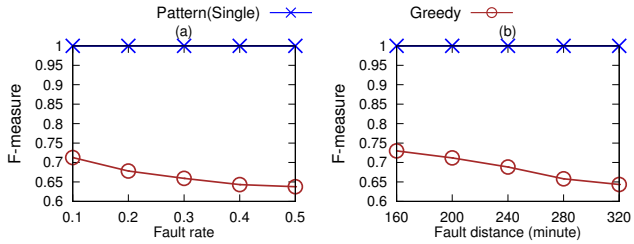

Figure 12: Application in event pattern matching debugging with (a) fault distance 160, (b) fault rate 0.1

conditions is a constant 2 in the given event patterns. That is, only 4 possible bindings $\Phi_k \in \aleph_\Gamma$ need to be considered in Line 1 in Algorithm 2. Consequently, the time cost of Pattern(Full) is about 4 times that of Pattern(Single).

Figure 11 evaluates queries without SEQ embedded in AND, i.e., AND($E_1, \ldots, E_n$) ATLEAST 900 WITHIN 1000. As illustrated in Figure 11(a), when the event number specified in the queries is small, e.g., 2 or 3, RMS error is lower. The reason is that the AND pattern specifies conditions on only (two) events in $E_1, \ldots, E_n$ with the minimum or maximum timestamps. For the conflicts occurring on events that are neither with the minimum nor the maximum timestamps, it cannot be modified. The less the events are in the AND conditions, the more likely the events have the minimum/maximum timestamps and thus will be modified. Moreover, when the event number is smaller, the corresponding possible bindings $\Phi_k \in \aleph_\Gamma$ are less as well. Therefore, in Figure 11(b), the time cost of Pattern(Full) reduces significantly with the decrease of event number. Nevertheless, Pattern(Single) with single binding keeps low time cost in all the event numbers, while the modification result for explanation is as good as Pattern(Full). The result is not surprising referring to their equivalence with regard to AND patterns without SEQ embedded, as illustrated in Proposition 8.

To sum up, Pattern(Full) is preferred when more accurate modification is desired. However, in most cases, Pattern(Single) is good enough due to its lower time cost without losing much accuracy.

6.4 Application

To evaluate how a human perceives the different responses, we report the accuracy of event pattern query over the explained tuples with various RMSE. As the scenario introduced in the Introduction, tuples are modified to explain why expected answers of an event pattern query are absent. Therefore, we evaluate the accuracy of query answers over the data after the timestamp modification explanation. Let truth be the set of expected answers and found be the set of returned tuples over the data after the modification explanation. The accuracy of query answering after explanation is evaluated by f-measure [14], precision $= \frac{|\text{found} \cap \text{truth}|}{|\text{found}|}$, recall $= \frac{|\text{truth} \cap \text{found}|}{|\text{truth}|}$, f-measure $= 2\frac{\text{precision} \cdot \text{recall}}{\text{precision}+\text{recall}}$. The higher the accuracy is, the better the explanation performs.

Figure 12 reports the accuracy of event pattern queries over the RTFM dataset in Section 6.3.2. Since the RMSE results of Pattern(Full) and Pattern(Single) explanations are close, we compare Pattern(Single) with Greedy which has a higher RMSE. As shown,

the Pattern(Single) explanations with lower RMSE in Figures 7(a) and 8(a) have a clearly better query accuracy than Greedy.

In summary, the timestamp modifications with lower RMSE indeed lead to more accurate event pattern query answers after perceiving these data explanations.

7 RELATED WORK

7.1 Semantic Query Optimization

Semantic query optimization (SQO) detects unsatisfiable queries based on application knowledge in order to avoid evaluating such queries [17] as well. For instance, Ding et al. [12] use pre-defined business logic or rules, such as a workflow model, as the application knowledge, to identify and then terminate the long running query processes that are guaranteed to not lead to successful matches. It requires that business events must be generated based on pre-defined business logics, which however is not always the case in practice. Since the event occurrence is constrained, the consistency checking is to simply check whether the event pattern query is compatible to the event constraints, which can be done in polynomial time [12].

In contrast, in this study, we determine the consistency of an event pattern query without such an assumption of event data generation. The novelty of Algorithm 1 is thus we do not need to pre-compute and check any query failure conditions based on pre-defined business logics. Indeed, without the assumption of predefined models on event occurrence, more combinations of events need to be considered in a larger space. Therefore, it is not surprising that the consistency checking problem of SEQ and AND patterns in general is more challenging (NP-complete) compared to the polynomial time solvable case (with pre-assumed application knowledge) in prior work on SQO [12].

7.2 Imprecise Timestamps

Zhang et al. [32] employ a temporal uncertainty model that assigns a time interval to each event to represent all of its possible occurrence times. The event pattern matching is thus evaluated in a set of possible worlds with deterministic timestamps. A confidence of match is computed by counting the possible worlds of matches, rather than explaining why the event pattern does not match.

Indeed, not each (matched) possible world is meaningful, with an arbitrary combination of all timestamps. Instead, our timestamp modification explanation proposes to consider the most likely modification of imprecise timestamps to satisfy the query pattern, following the minimum change idea in data cleaning [13] and time series repairing [31]. The rationale is that people or systems always try to minimize their mistakes in practice. Remarkably, with such a minimum modification intuition, we do not need to specify any knowledge on the time interval of possible occurrence times for each event (often impractical). While enumerating the possible worlds is no longer necessary, the timestamp modification problem is still challenging, not only NP-complete but also NP-hard to approximate to within any constant factor (Theorems 3 and 4).

7.3 Event Pattern Representation

The series-parallel graphs [28] can also capture the sequential and parallel relationships among vertexes, analogous to SEQ and AND over events in our study. In addition to SEQ and AND, the predicates WITHIN and ATLEAST in event patterns further specify the constraints on timestamp distances of events. It corresponds to the intervals attached on edges in the complex temporal networks, for example, in Figure 1, which are not considered in the series-parallel graphs. While the nested SEQ and AND patterns may be consistent, it is not in the presence of WITHIN and ATLEAST, as illustrated in Section 1.1.1. Owing to such differences in characterization properties, the (consistency checking) algorithms for series-parallel graphs are not applicable in our study.

Simple event patterns can be represented as SQL queries using a greater or less than filter expression, but with great complexity. Similar to introducing the MIN and MAX aggregation functions in SQL, the SEQ and AND patterns are introduced [12]. For example, a simple pattern AND(E_1, E_2) WITHIN 30 is equivalent to the complicated SQL query over relation Flight in Table 1, SELECT * FROM Flight WHERE ($E_1 \geq E_2$ AND $E_1 \leq E_2+30$) OR ($E_2 \geq E_1$ AND $E_2 \leq E_1 + 30$). However, such SQL notations for event patterns are not supported in the existing studies of explanations. Different from checking the consistency of the conjunctive query [17], the SQL representation of event pattern could also involve the disjunction of predicates, each of which corresponds to a possible binding. Existing data modification explanation [18] considers again a conjunction of predicates, which specify the relationship between an attribute and a constant. The event pattern needs to further handle the disjunction of predicates, comparing two timestamp attributes in a tuple.

8 CONCLUSIONS

In this paper, we study the non-answer explanations on event pattern queries. To this end, a number of theoretical results are first presented, including NP-hardness of pattern consistency explanation, timestamp modification explanation and approximation (Theorem 2-4). We then encode event patterns by a novel notation of complex temporal networks, such that interesting special cases as summarized in Table 2 are identified for solving the problems in the general case. In particular, for the special case of simple temporal networks, the timestamp modification problem can be solved as ILP and approximated by LP-relaxation. The general case is hence addressed by assembling the explanations of special cases. Remarkably, the single binding approximation returns the optimal explanation in certain cases (Proposition 8). Extensive experiments demonstrate that the approximate approaches return explanations comparable to the exact ones, but more efficient in time cost.

A possible constraint for the target environment could be the response time owing to the high complexity of generating explanations for event patterns. As analyzed in Theorems 2, 3 and 4, the explanation problems are generally hard. It limits the supported query patterns, for instance, Kleene star [33] is not considered in the current solution. We leave this challenging problem as the future study.

Acknowledgement

This work is supported in part by the National Key Research and Development Plan (2019YFB1705301), the National Natural Science Foundation of China (62072265, 61572272, 71690231), and the MIIT High Quality Development Program 2020.

REFERENCES

[1] http://data.4tu.nl/repository/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.
[2] http://lunadong.com/fusionDataSets.htm.
[3] https://explanation-event-pattern.github.io/explanation/full.pdf.
[4] https://github.com/explanation-event-pattern/explanation.
[5] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *SIGMOD Conference*, pages 147–160, 2008.
[6] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. *PVLDB*, 1(1):66–77, 2008.
[7] P. C. Arocena, B. Glavic, G. Mecca, R. J. Miller, P. Papotti, and D. Santoro. Messing up with BART: error generation for evaluating data-cleaning algorithms. *PVLDB*, 9(2):36–47, 2015.
[8] R. S. Barga, J. Goldstein, M. H. Ali, and M. Hong. Consistent streaming through time: A vision for event stream processing. In *CIDR 2007, Third Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 7-10, 2007, Online Proceedings*, pages 363–374, 2007.
[9] S. Chakravarthy, V. Krishnaprasad, E. Anwar, and S. Kim. Composite events for active databases: Semantics, contexts and detection. In *VLDB*, pages 606–617, 1994.
[10] A. Chapman and H. V. Jagadish. Why not? In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pages 523–534, 2009.
[11] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95, 1991.
[12] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W. Hsiung, and K. S. Candan. Runtime semantic query optimization for event stream processing. In *ICDE*, pages 676–685, 2008.
[13] W. Fan. Constraint-driven database repair. In *Encyclopedia of Database Systems*, pages 458–463. 2009.
[14] W. B. Frakes and R. Baeza-Yates. Information retrieval: data structures and algorithms. 1992.
[15] Y. Gao, Q. Liu, G. Chen, B. Zheng, and L. Zhou. Answering why-not questions on reverse top-k queries. *Proc. VLDB Endow.*, 8(7):738–749, 2015.
[16] L. George, B. Cadonna, and M. Weidlich. Il-miner: Instance-level discovery of complex event patterns. *Proc. VLDB Endow.*, 10(1):25–36, 2016.
[17] S. Guo, W. Sun, and M. A. Weiss. On satisfiability, equivalence, and impication problems involving conjunctive queries in database systems. *IEEE Trans. Knowl. Data Eng.*, 8(4):604–616, 1996.
[18] M. Herschel and M. A. Hernández. Explaining missing answers to SPJUA queries. *Proc. VLDB Endow.*, 3(1):185–196, 2010.
[19] J. Huang, T. Chen, A. Doan, and J. F. Naughton. On the provenance of non-answers to queries over extracted data. *Proc. VLDB Endow.*, 1(1):736–747, 2008.
[20] R. Huang. Approximate event pattern matching over heterogeneous and dirty sources. In M. d'Aquin, S. Dietze, C. Hauff, E. Curry, and P. Cudré-Mauroux, editors, *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, pages 3237–3240. ACM, 2020.
[21] N. Karmarkar. A new polynomial-time algorithm for linear programming. In *STOC*, pages 302–311, 1984.
[22] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.
[23] S. Lee, B. Ludäscher, and B. Glavic. Approximate summaries for why and why-not provenance. *Proc. VLDB Endow.*, 13(6):912–924, 2020.
[24] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: Is the problem solved? *PVLDB*, 6(2):97–108, 2012.
[25] C. H. B. Liu, B. P. Chamberlain, and E. J. McCoy. What is the value of experimentation and measurement? *Data Sci. Eng.*, 5(2):152–167, 2020.
[26] X. Ma, L. Zhang, L. Xu, Z. Liu, G. Chen, Z. Xiao, Y. Wang, and Z. Wu. Large-scale user visits understanding and forecasting with deep spatial-temporal tensor factorization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pages 2403–2411, 2019.
[27] Q. T. Tran and C. Chan. How to conquer why-not questions. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010, Indianapolis, Indiana, USA, June 6-10, 2010*, pages 15–26, 2010.
[28] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 1–12, 1979.
[29] J. Wang, T. Jin, R. K. Wong, and L. Wen. Querying business process model repositories - A survey of current approaches and issues. *World Wide Web*, 17(3):427–454, 2014.
[30] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pages 407–418, 2006.
[31] A. Zhang, S. Song, J. Wang, and P. S. Yu. Time series data cleaning: From anomaly detection to anomaly repairing. *Proc. VLDB Endow.*, 10(10):1046–1057, 2017.
[32] H. Zhang, Y. Diao, and N. Immerman. Recognizing patterns in streams with imprecise timestamps. *PVLDB*, 3(1):244–255, 2010.
[33] H. Zhang, Y. Diao, and N. Immerman. On complexity and optimization of expensive queries in complex event processing. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 217–228, 2014.