Towards Event Prediction in Temporal Graphs

Wenfei Fan[1,2,3], Ruochun Jin[1], Ping Lu[3], Chao Tian[4], Ruiqi Xu[5]
[1]University of Edinburgh [2]Shenzhen Institute of Computing Sciences [3]Beihang University
[4]Chinese Academy of Sciences [5]National University of Singapore
{wenfei@inf.,ruochun.jin@}ed.ac.uk,luping@buaa.edu.cn,tianchao@iscas.ac.cn,ruiqi.xu@nus.edu.sg

ABSTRACT

This paper proposes a class of temporal association rules, denoted by TACOs, for event prediction. As opposed to previous graph rules, TACOs monitor updates to graphs, and can be used to capture temporal interests in recommendation and catch frauds in response to behavior changes, among other things. TACOs are defined on temporal graphs in terms of change patterns and (temporal) conditions, and may carry machine learning (ML) predicates for temporal event prediction. We settle the complexity of reasoning about TACOs, including their satisfiability, implication and prediction problems. We develop a system, referred to as TASTE. TASTE discovers TACOs by iteratively training a rule creator based on generative ML models in a creator-critic framework. Moreover, it predicts events by applying the discovered TACOs. Using real-life and synthetic datasets, we experimentally verify that TASTE is on average 31.4 times faster than conventional data mining methods in TACO discovery, and it improves the accuracy of state-of-the-art event prediction models by 23.4%.

PVLDB Reference Format:
Wenfei Fan, Ruochun Jin, Ping Lu, Chao Tian, and Ruiqi Xu. Towards Event Prediction in Temporal Graphs. PVLDB, 15(9): 1861-1874, 2022.
doi:10.14778/3538598.3538608

1 INTRODUCTION

Event prediction is to predict a real-world occurrence that relates to a particular topic and will take place at a specific time [109]. Events range from large-scale (*e.g.,* disease outbreaks), to medium-scale (*e.g.,* system failures), to small-scale (*e.g.,* fraud detection). Event prediction is important in a variety of domains such as disease control and business intelligence. To illustrate, consider online recommendation, a special case of event prediction ("sale events"), to recommend items to users. Recommendation models are mainly categorized into collaborative filtering (CF) by learning from user-item interactions, content-based (CB) by assessing the similarity of content features of users and items, and hybrid by integrating the two [107]. In the real world, however, e-commerce companies often monitor changes to transaction graphs, and employ rules to catch users' temporal interests and detect fraudulent behaviors.

Example 1: Below are two rules from an e-commerce platform.

(1) If a movie is nominated for a film award and if a user watches the movie within two weeks after its nomination, then recommend the movie to the friends of the user between the nomination date and the date of the award ceremony. Here the nomination indicates a "change" that triggers the recommendation in a time interval. Such cases are beyond conventional CF and CB models.

(2) If a user searched "barbecue" at least m times in June, then recommend meat to the user in the next two months.

Temporal changes are also used to predict other events.

(3) If at least m cases of an infectious disease z are reported in an area within the past 2 weeks, then offer vaccines for z to the people there.

(4) If device M is used to access k accounts only once within a short period of one hour, and if each of these accounts has been regularly accessed by other devices at least m times in the past month, then device M is likely committing account-takeover attacks. □

Challenges. No matter how important, event prediction is challenging. While rules have been studied for graphs, *e.g.,* graph functional dependencies (GFDs) [12], graph association rules (GARs) [21] and Horn rules [25], they are defined on *static* graphs and cannot express temporal changes. There are several open questions. How can we specify patterns of changes to graphs and temporal interests of users in logic rules? Can we improve the accuracy of event-prediction ML models with temporal conditions, and interpret their predictions? How expensive is it to reason about rules with temporal conditions and embedded ML models? How can we efficiently discover such rules from real-life graphs? Is it possible to scale with large graphs when we apply the rules to predict events?

Contributions & organization. This paper tackles these issues.

(1) Temporal rules (Section 2). We propose a class of temporal association rules, referred to as TACOs (TemporAl event prediCtiOn rules), to enrich event-prediction ML models with temporal conditions. TACOs are defined on temporal graphs in terms of change patterns, temporal conditions and ML prediction models (as predicates). In contrast to previous graph rules, TACOs are applied to *updates* to temporal graphs, which are typically much smaller than the entire graphs, to monitor changes and predict events.

(2) Complexity (Section 3). We study classical problems for TACOs, including (a) satisfiability to check whether a set of TACOs has no conflicts, (b) implication to decide whether a set of TACOs entails another TACO, and (c) prediction to forecast whether an event will occur. We show that they are Σ_2^p-complete, Π_2^p-complete and NP-complete, respectively. That is, despite the increased expressivity, TACOs do not make our lives much harder compared with [21, 22].

(3) TASTE *system* (Section 4). Despite the intractability, we develop a system called TASTE (TemporAl SysTEm). TASTE (a) discovers high-quality TACOs with generative ML models, and (b) applies the discovered TACOs for temporal event prediction in parallel.

Proceedings of the VLDB Endowment, Vol. 15, No. 9 ISSN 2150-8097.
doi:10.14778/3538598.3538608

(4) Rule discovery (Section 5). From the perspective of probability distribution learning, we develop a *creator-critic* framework to discover TACOs in TASTE, as a major contribution. The rule creator adopts GAN and LSTM language models to generate TACOs with candidate patterns and temporal conditions, while the critic evaluates such TACOs rules on temporal graphs, collects high-quality rules, and provides feedback to the creator for improving its quality in the next round. Different from the traditional discriminator in GAN, the "critic" is a predefined scoring algorithm that requires no training. This method avoids the exhaustive levelwise search in an exponentially large space [20] and is able to find high-quality TACOs with large graph patterns.

(5) Parallel prediction (Section 6). We develop a parallel algorithm for temporal event prediction with TACOs. In contrast to existing graph partitioning methods such as edge-cut or vertex-cut [3, 77], we propose to partition a temporal graph based on temporal locality such that event prediction can be made communication-free. We show that the parallel event prediction algorithm guarantees to run faster when given more processors, *i.e.,* parallel scalability.

(6) Experimental study (Section 7). Using real-life and synthetic datasets, we verify the accuracy, efficiency and scalability of TASTE. We find the following. On average, (1) TASTE is 23.8% and 23.0% more accurate than prior methods for temporal event prediction and dynamic recommendation, respectively. (2) The creator-critic discovery method of TASTE is more than 31 times faster than the levelwise algorithms. It finds TACOs with patterns of 20 edges in 1639s, in contrast to days by the conventional methods. (3) 84.76% of rules mined by levelwise methods can be found by TASTE using a small amount of training data and training rounds. (4) The event prediction algorithm in TASTE is parallelly scalable, *e.g.,* its runtime is reduced by 3.2 times when using 32 processors instead of 4.

Novelty. The paper proposes an approach towards event prediction by enriching ML models with temporal logic conditions in logic rules, and a practical method to disc over rules with large graph patterns. The novelty of the work consists of (1) TACOs, a class of rules for event prediction on *dynamic* graphs; (2) the complexity bounds of the satisfiability, implication and prediction problems for TACOs; (3) a creator-critic framework, the first that is able to discover high-quality graph rules with large patterns; (4) a strategy for partitioning temporal graphs based on temporal locality; (5) an algorithm for temporal event prediction that guarantees the parallel scalability; and (6) TASTE, a system based on new approaches to discovering TACOs and predicting events with TACOs.

Related work. We categorize the related work as follows.

Temporal event prediction. A variety of ML models have been explored for temporal event prediction, including (1) regression [24, 48, 64, 67]; (2) point processes [73, 99]; (3) survival analysis [16, 94]; and (4) embedding via, *e.g.,* tensor decomposition [18, 105], recurrent neural network (RNN) [36, 56, 92, 93], graph neural network (GNN) [11, 50, 58, 71, 81], autoencoder [29, 30, 78] and diachronic encoder [15, 26, 28, 101]. In particular, as a special type of event prediction, (temporal) recommendation has attracted a great deal of interest. For example, dynamic recommender systems (DRS) have been developed to improve the recommendation quality. They can incorporate temporal factors of user preferences [75] using time-dependent neighborhood models [23, 60, 89], matrix factorization [42, 55, 59, 76] and tensor models [33, 96, 97].

Temporal association rules have been developed to specify time-dependent correlations of transaction data in relational tables [95, 102]. In addition, association rules have also been studied on graphs [5, 21, 65]. GARs [21] are defined on static graphs to specify regularities among entities. GTARs (Graph Temporal Association Rules) [65] specify connections between events by means of two event patterns that share a common focus node and a single constant time interval. Similarly, GERs (Graph Evolution Rules) [5] are defined with two connected sub-patterns that are decomposed from a singe pattern, to represent local changes, *e.g.,* relabeling [83].

This work differs from the prior work in the following.

(1) As an effort to unify rule-based and ML-based methods for event prediction, we propose TACOs by embedding event-prediction ML models as predicates. This allows us to not only leverage existing ML models, but also refine the ML models with logic conditions.

(2) Unlike existing ML-based prediction strategies (*e.g.,* DRS) that adopt unexplainable blackbox models, our method with TACOs offers a logic interpretation of the ML predictions for temporal events.

(3) TACOs are more expressive than GARs, GTARs and GERs. As will be seen in Section 2, these rules can all be expressed as TACOs. TACOs may embed prediction ML methods beyond GTARs and GERs, and support various temporal conditions on different events, while GTARs and GERs can only express constant time intervals.

(4) We establish the complexity of classical problems for temporal graph rules. While these problems were studied for GARs on static graphs, no prior work has considered these for GTARs or GERs.

Rule learners. Prior graph rule discovery methods can be classified as follows: (1) levelwise data mining approaches that iteratively enumerate all possible candidate rules, and prune non-interesting rules with criteria such as support/confidence [5, 20, 65], head coverage and PCA confidence [25]; and (2) rule learners that typically enumerate paths as candidate rules and learn a weight for each candidate to quantify its quality. The weight learning methods include Markov logic networks [40, 79], path ranking [44], relational dependency networks [66, 68], neural logic programming [80, 82, 103], and reinforcement learning [14, 52, 62, 74, 87, 100].

Different from existing rule learners, (a) we study rule discovery in temporal graphs, while the prior learners focus on static graphs. (b) We propose to discover TACOs with general topological structures, beyond merely paths. (c) We discover TACOs that may carry ML predicates. (d) We propose a deep-generative-model-based approach to learning graph rules with large graph patterns. In contrast, conventional mining methods for graph rules [20] can hardly find rules with patterns of 7 edges or more.

Pattern generators. Graph generative models have been extensively studied to build graph patterns. Such models usually ensure that the generated patterns sustain certain properties, *e.g.,* the power-law node degree distribution [1], community structures [84, 98], distribution over random walks [7, 110], distribution of node and edge sequences [51, 54, 104], graphlets distribution [35, 106] and time-invariant conditions and time-variant conditions [108].

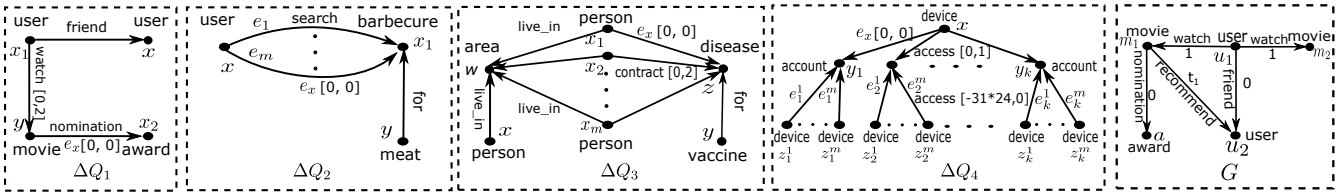


Figure 1: Δ-patterns and temporal graph

Pattern generation is a step in our creator-critic framework. However, our TACO discovery method departs from pattern generators since it incorporates (a) language models to attach node and edge labels to generated change patterns, and (b) strategies to create dependencies with temporal conditions and ML predicates. None of these has been studied before in pattern generation.

2 TEMPORAL EVENT PREDICTION RULES

We introduce TACOs in this section. We start with basic notations (Section 2.1) and then define temporal prediction rules (Section 2.2).

2.1 Preliminaries

Assume three countably infinite sets of symbols, denoted by Γ, Υ and Ω, for labels, constants and timestamps, respectively. Here Ω is linearly ordered *w.r.t.* a discrete order $\leq$. We denote a *time window* by τ, which is $[t_1, t_2]$ for *timestamps* t_1 and t_2 ($t_1 \leq t_2$) selected from Ω.

Temporal graphs. A *temporal graph* is specified as $G=(V, E, L, T, F_A)$, where (a) V is a finite set of nodes; (b) $E \subseteq V \times \Gamma \times \Omega \times V$ is the set of edges, where $e=(v, l, t, v')$ denotes an edge from v to v' that is labeled with $l \in \Gamma$ and carries timestamp $t=T(e) \in \Omega$; (c) each node $v \in V$ has label $L(v)$ from Γ; and (d) node v carries a tuple $F_A(v)=(A_1 = a_1, \ldots, A_n = a_n)$ of *attributes* of a finite arity, where A_i denotes a property and $a_i \in \Upsilon$, written as $v.A_i=a_i$, and $A_i \neq A_j$ if $i \neq j$.

We refer to G as a *graph* when it is clear from the context.

Intuitively, G is a directed labeled graph in which each edge has a timestamp recording when it is added to G or when it is last updated. Note that from a node v_1 to v_2, there may exist multiple edges, possibly with the same label but different timestamps. To simplify the presentation, we specify node timestamps by attaching timestamped self-loop edges to nodes to mark the times of node updates.

Δ-patterns. A *Δ-pattern* is $\Delta Q[\bar{x}] = (V_Q, E_Q, L_Q, T_Q, \mu)$, where (1) V_Q (resp. E_Q) is a set of pattern nodes (resp. edges); (2) L_Q assigns a label in Γ to each pattern node $u \in V_Q$ (resp. edge $e \in E_Q$); (3) T_Q assigns a time window τ to each pattern edge; in particular, one designated edge e_x is given $\tau = [0, 0]$ that fixes the current time; (4) $\bar{x}$ is a list of distinct variables; and (5) μ is a bijective mapping from $\bar{x}$ to V_Q, *i.e.,* it assigns a distinct variable to each node in V_Q. For $x \in \bar{x}$, we use $\mu(x)$ and x interchangeably when it is clear from the context.

Example 2: Figure 1 depicts Δ-patterns ΔQ_1–ΔQ_4 for the cases of Example 1: (1) ΔQ_1 specifies the change that a movie is nominated for an award, together with potential users, in which the designated edge e_x is labeled nomination; (2) ΔQ_2 depicts that a user poses m queries about barbecue in June, where edge e_x indicates the last query; (3) ΔQ_3 shows the change of m cases of disease z found in an area w, along with a vaccine y for z; here e_x indicates the first case reported; and (4) ΔQ_4 is an abnormal pattern of account accesses, where e_x is the first access edge from device x to account y_1. □

Temporal pattern matching. A *match* of Δ-pattern $\Delta Q[\bar{x}]$ in graph G is a homomorphism h from $\Delta Q[\bar{x}]$ to G such that (a) for each pattern node $u \in V_Q$, $L_Q(u) = L(h(u))$; and (b) for each pattern edge $e=(u, l, \tau, u')$ in $\Delta Q[\bar{x}]$, $e'=(h(u), l, t, h(u'))$ is an edge in G and $t - t^* \in \tau$. Here t^* denotes the *current time*, *i.e.,* the timestamp of the edge in G to which the designated edge e_x of $\Delta Q[\bar{x}]$ is mapped via h.

We denote by $\Delta Q(G)$ the set of all matches of $\Delta Q[\bar{x}]$ in graph G.

Intuitively, $\Delta Q[\bar{x}]$ is a *change pattern*. It monitors updates to graph G. The matches of $\Delta Q[\bar{x}]$ in G can be computed in ΔG_Q, which includes nodes and edges (updates) with timestamps in the range $[t_{\mathsf{min}}, t_{\mathsf{max}}]$ relative to the current time t^*, and is referred to as *updates to G relative to* ΔQ. Here t_{min} and t_{max} denote the earliest and latest timestamps in $\Delta Q[\bar{x}]$, respectively. The updates are typically much smaller than G. Consequently, it is often *more efficient* to compute matches of a Δ-pattern than that of a conventional pattern.

2.2 Definition and Semantics of Rules

TACOs are defined in terms of Δ-patterns and predicates.

Predicates. *Predicates* p of a Δ-pattern $\Delta Q[\bar{x}]$ have the form:

$$x.A \mid l(x, y) \mid \mathcal{M}(x, y, l, t) \mid x.A \oplus y.B \mid x.A \oplus c \mid e_1.t \oplus e_2.t \mid e.t \oplus c,$$

where $x, y \in \bar{x}$, e_1, e_2 and e are pattern edges of $\Delta Q[\bar{x}]$, $x.A$ denotes an attribute A of x, c is a constant, $l(x, y)$ is an edge from x to y labeled l, $e.t$ is the timestamp of edge e, and $\oplus$ is one of $=, \neq, <, \leq, >, \geq$. We write $e.t \in [t_1, t_2]$ if $e.t \geq t_1$ and $e.t \leq t_2$. We refer to $e_1.t \oplus e_2.t$ and $e.t \oplus c$ as *temporal predicates*.

In particular, $\mathcal{M}$ is an ML classifier for event prediction on temporal graphs; it returns true if it predicts that the event indicated by edge $l(x, y)$ will take place at time $t^* + t$. It can be a dynamic recommendation model, *e.g.,* SASRec [37], or a temporal graph completion model, *e.g.,* RE-GCN [50]. We refer to $\mathcal{M}$ as an *ML predicate*.

ML models. We require the model $\mathcal{M}$ to work in a "transductive setting" [10], where it infers information between nodes with observed labels. Once the training of $\mathcal{M}$ and TACO discovery complete, the embeddings of the observed labels are fixed and no new embeddings that may violate the discovered TACOs will be introduced. This enables interpreting $\mathcal{M}$ via logic predicates in TACOs.

Rules. A *temporal event prediction rule* (TACO) φ is defined as

$$\Delta Q[\bar{x}](X \rightarrow (p_0, \tau)),$$

where $\Delta Q[\bar{x}]$ is a Δ-pattern, X is a conjunction of predicates of $\Delta Q[\bar{x}]$, p_0 is a predicate of $\Delta Q[\bar{x}]$, and τ is a time window. When $p_0 = l(x, y)$, the edge represented by p_0 is predicted and is not necessarily already in $\Delta Q[\bar{x}]$. We refer to $\Delta Q[\bar{x}]$, X, p_0 and $X \rightarrow (p_0, \tau)$ as the *pattern*, *precondition*, *event* and *dependency* of φ, respectively.

Intuitively, the TACO says that if the precondition X holds on the entities matched by the change pattern $\Delta Q[\bar{x}]$, then the event specified by p_0 will take place within the time window τ.

Example 3: The rules of Example 1 can be expressed as TACOs.

(1) $\varphi_1 = \Delta Q_1[\bar{x}](X_1 \rightarrow (\text{recommend}(y, x), [0, t_1]))$, where $X_1 = \emptyset$ and t_1 is a timestamp in weeks. The TACO says that if user x_1 watches movie y in 2 weeks after the nomination of y for award z and if x is a friend of x_1, then recommend y to x by the date $t^* + t_1$ of the award ceremony. Note that the condition that user x_1 watches the movie in 2 weeks is specified by function T_Q in ΔQ_1 (see Fig. 1).

(2) $\varphi_2 = \Delta Q_2[\bar{x}](X_2 \rightarrow (\text{recommend}(y, x), [0, 60]))$, where X_2 is $(\bigwedge_{i\in[1,m-1]} e_i.t < e_{i+1}.t) \wedge (\bigwedge_{i\in[1,m]} e_i.t \in [t_{\text{june}}, t'_{\text{june}}])$. Here $e_i.t < e_{i+1}.t$ ensures that searches e_i and e_{i+1} are distinct and t_{june} (resp. t'_{june}) indicates the date of June 1 (resp. June 30). It says that if x searches barbecue at least m times in June (specified in X_2), then recommend meat to x in the next two months (60 days).

(3) $\varphi_3 = \Delta Q_3[\bar{x}](X_3 \rightarrow (\text{offer}(y, x), [2, 6]))$. Precondition X_3 is $\bigwedge_{i,j\in[1,m],i\neq j}(x_i.\text{id} \neq x_j.\text{id})$, to enforce that the cases x_i and x_j are distinct. Observe that in the Δ-pattern ΔQ_3, the time window $[0, 2]$ indicates that person x_i contracts disease z in 2 weeks after the current time t^*, *i.e.,* the earliest case confirmed. It says that if at least m cases of disease z are confirmed in area w within 2 weeks and if person x lives in w, then offer vaccine y for disease z to x in a month.

(4) $\varphi_4 = \Delta Q_4[\bar{x}](X_4 \rightarrow (x.\text{status} = \text{fraud}, [0, 0]))$, where X_4 is $\bigwedge_{i\in[1,k]}((\bigwedge_{j,l\in[1,m],j\neq l} z_i^j.\text{id}{=}z_i^l.\text{id} \wedge e_i^j.t{\neq}e_i^l.t) \wedge \mathcal{M}(x, y_i, \text{attack}, e_i.t))$. Here $\mathcal{M}$ is an ML model suspecting that x attacked account y_i at time $e_i.t$. The time windows in ΔQ_4 are in hours. The TACO says that if device x accesses $y_1, \ldots, y_k$ within one hour, $\mathcal{M}$ suspects them as attack behaviors, and if each y_i has been accessed by other devices m times in the past month, then x is likely a fraud. □

Remark. (1) GARs [21], GTARs [65] and GERs [5] are special cases of TACOs. (a) GARs are TACOs when all time windows are $[-\infty, \infty]$ and preconditions X carry no temporal predicates. We adopt a single predicate p_0 for the event in each TACO to simplify the discussion; this does not lose expressive power compared to GARs [20]. (b) A GTAR can be expressed as multiple TACOs, where each TACO (i) has a pattern that extends the antecedent pattern P_1 of the GTAR with the nodes in its consequent pattern P_2, and (ii) encodes a single edge from P_2 as the event p_0 with $\tau = [t_\alpha, t_\alpha]$, where t_α is the constant time interval specified in the GTAR. (c) Similarly, a GER can be encoded as a set of TACOs, one for each update indicated by its patterns with the maximum timestamp [5].

(2) As demonstrated by φ_4 of Example 3, one can plug a well-trained ML prediction model $\mathcal{M}$ in precondition X, and enrich $\mathcal{M}$ with temporal and logic conditions. A TACO of the form $\Delta Q[\bar{x}](X \rightarrow (\mathcal{M}(x, y, l, t), \tau))$ interprets ML predictions with logic predicates.

(3) One can associate TACOs with a probability for the predicted events. However, this would incur higher complexity (#P-hard) as indicated by probabilistic logic programming [41].

(4) TACOs can readily handle interval-timestamped graphs G [4]. The only difference is that when pattern edge e of a TACO matches edge e' in G, we require that the time range of e overlaps with at least one time interval of e', instead of covering a single timestamp.

Semantics. Denote by $h(\bar{x})$ a match of $\Delta Q[\bar{x}]$ in a graph G, and p a predicate of $\Delta Q[\bar{x}]$. We say that match $h(\bar{x})$ *satisfies* a predicate p, denoted by $h(\bar{x}) \models p$, if the following condition is satisfied: (a) if p is $x.A$, then node $h(x)$ carries attribute A; (b) if p is $l(x, y)$, then there is an edge from $h(x)$ to $h(y)$ labeled l; (c) if p is $\mathcal{M}(x, y, l, t)$, then the ML classifier $\mathcal{M}$ predicts an association between $h(x)$ and $h(y)$ with label l at time $t^* + t$; (d) if p is $x.A \oplus y.B$, then attributes A and B exist at $h(x)$ and $h(y)$, respectively, and $h(x).A \oplus h(y).B$; similarly for $x.A \oplus c$; and (e) if p is $e_1.t \oplus e_2.t$, then the timestamps of edges matching e_1 and e_2 have the $\oplus$ relationship; similarly for $e.t \oplus c$.

For a conjunction X of predicates, we write $h(\bar{x}) \models X$ if $h(\bar{x}) \models p$ for all p in X. We write $h(\bar{x}) \models \varphi$ for TACO $\varphi = \Delta Q[\bar{x}](X \rightarrow (p_0, \tau))$ if $h(\bar{x}) \models X$ implies $h(\bar{x}) \models p_0$ and p_0 occurs within time window τ.

We say that graph G *satisfies* TACO $\varphi = \Delta Q[\bar{x}](X \rightarrow (p_0, \tau))$, denoted by $G \models \varphi$, if for all matches $h(\bar{x})$ of $\Delta Q[\bar{x}]$ in G, $h(\bar{x}) \models \varphi$. We say that G *satisfies* a set Σ of TACOs, denoted by $G \models \Sigma$, if for all TACOs $\varphi \in \Sigma$, $G \models \varphi$, *i.e., G* satisfies every TACO in Σ.

Example 4: Consider the temporal graph G illustrated in Figure 1, where each edge is annotated a timestamp. Then G satisfies the TACO φ_1 of Example 3. Observe that there only exists a singe match $h(\bar{x})$ of ΔQ_1 in G: $x{\mapsto}u_2$, $x_1{\mapsto}u_1$, $x_2{\mapsto}a$ and $y{\mapsto}m_1$. As u_2 is linked from m_1 via a recommend edge with timestamp t_1 in G, we know that $h(\bar{x}) \models (\text{recommend}(y, x), [0, t_1])$, and $G \models \varphi_1$ follows. □

The notations of the paper are summarized in Table 1.

Table 1: Notations

Notations	Definitions
G, $\Delta Q[\bar{x}]$	graph and Δ-pattern, respectively
ΔG_Q	updates to G relative to pattern ΔQ
φ	TACO $\Delta Q[\bar{x}](X \rightarrow (p_0, \tau))$
$\text{supp}(\varphi, G)$	the support of TACO φ in graph G
$\text{conf}(\varphi, G)$	the confidence of TACO φ in graph G
α, β, γ, δ	thresholds (node, support, confidence, time window)

3 REASONING ABOUT TEMPORAL RULES

In this section we study fundamental problems for TACOs, including the satisfiability, implication and prediction problems. We show that although TACOs are more expressive than GARs [21], these problems for TACOs are not much harder than for GARs.

Satisfiability. The *satisfiability problem* is as follows.

- Input: A set Σ of TACOs.
- Question: Does there exist a graph G such that $G \models \Sigma$ and for each TACO $\Delta Q[\bar{x}](X \rightarrow (p_0, \tau)) \in \Sigma$, ΔQ has a match in G?

Intuitively, this is to make sure that the discovered TACOs have no conflicts and can be applied to a graph at the same time.

The problem is coNP-complete for GARs [21] and Σ_2^p-complete for graph denial constraints (GDCs) [22]. We show the following.

Theorem 1: *The satisfiability problem is Σ_2^p-complete for* TACOs. □

Here Σ_2^p denotes the class of decision problems that can be checked in NP using an NP oracle, *i.e.,* $\Sigma_2^p = \text{NP}^{\text{NP}}$. Similarly, the class $\Pi_2^p = \text{coNP}^{\text{NP}}$ is defined (see [70] for more details).

We assume *w.l.o.g.* the following about ML predicates in TACOs; (a) ML prediction (*i.e.,* testing with pre-trained $\mathcal{M}$) takes polynomial time (PTIME), and (b) there exists a "small" range of values such that for any node pairs (x, y), their attribute values can be encoded by values in the small range and yield the same truth value for $\mathcal{M}$. These are the case for ML models used by TACOs in practice.

Proof sketch: We first show a small model property: if a set Σ of TACOs is satisfiable, then there exists a temporal graph G_Σ such that $|G_\Sigma| \leq 4|\Sigma|^3$ and $G_\Sigma \models \Sigma$. Based on this, we develop an Σ_2^p algorithm to check whether a given set Σ of TACOs is satisfiable.

The lower bound follows from the Σ_2^p-completeness of the satisfiability problem for GDCs [22], since it is easy to verify that GDCs are also a special case of TACOs. □

Implication. A set Σ of TACOs *implies* a TACO φ, denoted by $\Sigma \models \varphi$, if for all graphs G, if $G \models \Sigma$ then $G \models \varphi$.

The *implication problem* is stated as follows.

- Input: A set Σ of TACOs and a TACO φ.
- Question: Does $\Sigma \models \varphi$?

The implication analysis helps us remove redundant rules.

The implication problem is NP-complete for GARs [21] and Π_2^p-complete for GDCs [22]. For TACOs it is no harder than for GDCs.

Theorem 2: *The implication problem is* Π_2^p*-complete for* TACOs. □

Proof sketch: We prove a small model property for the problem: if $\Sigma \not\models \varphi$, then there exists a temporal graph G_φ such that $|G_\varphi| \leq 8|\varphi|(|\varphi|+|\Sigma|)^2$, $G_\varphi \models \Sigma$ but $G_\varphi \not\models \varphi$. Based on this, we design an Σ_2^p algorithm to check whether $\Sigma \not\models \varphi$. The lower bound directly follows from the Π_2^p-complete implication problem for GDCs [22]. □

Prediction. We also study the *prediction problem*.

- Input: A temporal graph G, a set Σ of TACOs, a time window τ, a label l, and two nodes u and v in G.
- Question: Does there exist edge labeled l from u to v in τ by Σ?

This is to study the complexity of temporal prediction with TACOs.

A similar problem (deduction problem) was shown NP-complete for GARs [21]. It is no harder to predict events with TACOs.

Theorem 3: *The prediction problem is* NP*-complete for* TACOs. □

Proof sketch: For the upper bound, we define a notion of proof trees and show that ($\diamond$) the prediction (p_0, τ) can be deduced in G if and only if there exists a proof tree $\mathcal{T}$ with $|\mathcal{T}| \leq 7|G|^2(|G| + |\Sigma|)^2(|G|+|\tau|)^3$ that witnesses the prediction. Based on ($\diamond$), we design an NP algorithm for the problem: guess a tree $\mathcal{T}$ such that $|\mathcal{T}| \leq 7|G|^2(|G| + |\Sigma|)^2(|G|+|\tau|)^3$ and check whether it is a proof tree of the prediction (p_0, τ) in PTIME.

We show the lower bound by reduction from the graph homomorphism problem, which is NP-complete (cf. [27]). The latter is to decide, given two undirected graphs G_1 and G_2, whether there exists a homomorphism h from G_1 to G_2. We use TACO φ to encode G_1, temporal graph G to encode G_2, and (p_0, τ) to represent h. □

Remark. The complexity bounds remain unchanged for interval-timestamped graphs [4]. For the prediction problem, we only need to check whether a time interval overlaps with another, which takes PTIME. For the other two problems, the small model properties remain intact and so does the complexity. Moreover, the results also hold when timestamps are chosen from densely ordered set.

4 TEMPORAL SYSTEM

In this section we introduce TASTE (TemporAl SysTEm), a system for discovering TACOs and predicting events with TACOs. We first state the corresponding discovery and prediction problems (Section 4.1). We then present the architecture of TASTE (Section 4.2).

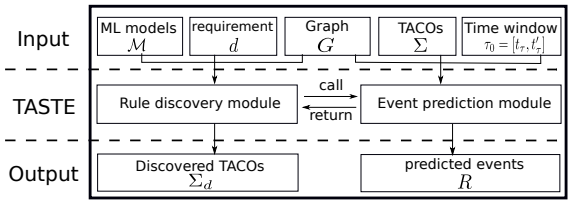


Figure 2: Architecture, inputs and outputs of TASTE

4.1 TACO Discovery and Event Prediction

We first define quality measures for TACOs.

Quality metrics. We use support and confidence to quantify the quality of TACOs, for frequency and reliability, respectively.

Support. The *support* of TACO $\varphi = \Delta Q[\bar{x}](X \rightarrow (p_0, \tau))$ in graph G is

$$\text{supp}(\varphi, G) = |\Delta Q(e_x, \varphi, G)|,$$

where $\Delta Q(e_x, \varphi, G)$ denotes the set of edges $h(e_x)$ in those matches $h(\bar{x}) \in \Delta Q(G)$ such that $h(\bar{x}) \models X$, $h(\bar{x}) \models p_0$, and p_0 occurs in τ.

Intuitively, the support counts the number of distinct "satisfiable" matches in regards to the designated edge e_x. One can verify that this measure is anti-monotonic *w.r.t.* a partial order $\preceq$ over TACOs, *i.e.,* $\text{supp}(\varphi, G) \geq \text{supp}(\varphi', G)$ for any G if $\varphi \preceq \varphi'$.

Confidence. We define the *confidence* of a TACO $\varphi = \Delta Q[\bar{x}](X \rightarrow (p_0, \tau))$ in graph G, denoted by $\text{conf}(\varphi, G)$, as follows:

$$\text{conf}(\varphi, G) = \frac{\text{supp}(\varphi, G)}{|\Delta Q(e_x, X, G)|}.$$

Here $\Delta Q(e_x, X, G)$ is the set of edges $h(e_x)$ in *all* matches $h \in \Delta Q(G)$ that satisfy X. The ratio quantifies the likelihood that event p_0 happens in τ with matches satisfying prediction X, *i.e.,* reliability.

TACO discovery. The discovery problem for TACOs is as follows.

- *Input*: A temporal graph G, a positive integer α, a support threshold $\beta > 0$, a confidence threshold $\gamma \in [0, 1]$, and a bound $\delta > 0$ on the lengths of time windows.
- *Output*: A set Σ of TACOs such that for each $\varphi \in \Sigma$, $\text{supp}(\varphi, G) \geq \beta$, $\text{conf}(\varphi, G) \geq \gamma$, $t_2 - t_1 \leq \delta$ for each time window $[t_1, t_2]$ in φ, and there are at most α nodes in the pattern of φ.

We call the tuple $(\alpha, \beta, \gamma, \delta)$ a *discovery requirement* d. The problem aims to discover the set of all TACOs that conform to d, and have the expected number of pattern nodes bounded by α.

Temporal event prediction. The problem is stated as follows.

- *Input*: A graph G, a set Σ of TACOs and a time window τ_0.
- *Output*: A set R of edges for G predicted by Σ such that the occurrence of the event encoded by each edge in R is within τ_0.

To simplify the discussion, we consider events p_0 represented by edges $l(x, y)$, as commonly found in practice.

4.2 Overview of TASTE

The architecture of TASTE is depicted in Figure 2. TASTE supports two modules for TACO discovery and event prediction, as follows.

(1) Rule discovery. The discovery module is responsible for mining a set Σ_d of TACOs that conform to the given requirement d from temporal graph G, referred to as *high-quality* rules. Apart from d, it takes pre-trained ML models $\mathcal{M}$ (Section 2) as additional input and embeds them in TACOs when discovering Σ_d.

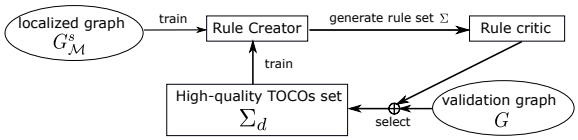


Figure 3: Dataflow of the iterative TACO discovery.

(2) Event prediction. The prediction module employs the discovered TACOs in Σ_d to predict all events (or a particular event) by running a parallel algorithm. It should be mentioned that this module can also accept TACOs provided by domain experts if available.

We will present algorithms underlying the discovery and prediction modules in Sections 5 and 6, respectively.

5 GENERATION-BASED TACO DISCOVERY

In this section we propose a Creator-Critic Discovery approach, denoted by CCD, to discovering high-quality TACOs Σ_d. We develop such an algorithm for the rule discovery module of TASTE.

Below we first introduce the generation-based approach and justify the method from the perspective of probability distribution learning. We then present CCD and its functions and models. Finally we formally prove the performance guarantees of CCD.

Generation-based discovery. One might be tempted to extend the existing levelwise search-based methods, *e.g.,* [5, 20], to enumerate candidate TACOs and identify the required ones. However, these approaches need to search in a lattice and take exponential time, despite pruning strategies to reduce the cost [5, 20, 65].

In view of this, we propose CCD, which discovers high-quality TACOs by employing generative adversarial networks (GAN), and using interactive learning with weak supervision between a rule creator and a rule critic. This GAN-based framework is inspired by the latest progress in drug discovery and compound design [6, 88], where a GAN is trained to generate promising candidates without enumerating all possible combinations in the large search space. Such generative approach significantly accelerates the drug discovery process and produces more promising compounds.

As shown in Figure 3, in each iteration, the creator of CCD first employs ML models to generate a set Σ of candidate TACOs. Then the critic selects high-quality rules from Σ and adds them to Σ_d. Taking rules in Σ_d as feedback from the critic, the creator trains itself again to improve the probability of generating high-quality TACOs in the next iteration. CCD returns Σ_d when the number of iterations reaches a user-specified bound I.

From the perspective of probability theory, each TACO φ can be viewed as an event and the union of all TACOs constitutes a discrete countably infinite sample space $\mathcal{A}$. Given requirement d, the discovery aims to find a probability distribution $\mathbb{P}$ described by a probability mass function $P\colon \mathcal{A} \to \mathbb{R}$ (real numbers), where

$$\begin{cases} P(\varphi) > 0, \text{if } \varphi \text{ is a high-quality TACO } \textit{w.r.t. } d, \\ P(\varphi) = 0, \text{if } \varphi \text{ is not a high-quality TACO } \textit{w.r.t. } d. \end{cases} \quad (1)$$

In order to find the distribution, conventional levelwise methods build a searching lattice to enumerate each rule φ in $\mathcal{A}$ and check whether it meets the requirement d. In contrast, CCD interactively trains deep generative models [63, 110] in the creator to approximate the target distribution $\mathbb{P}$, avoiding costly search in the exponentially large sample space.

Algorithm 1: Creator-Critic Discovery (CCD)

Input: A temporal graph G, discovery requirement d=(α, β, γ, δ), ML models $\mathcal{M}$, sample size N, and iteration number I.
Output: A set Σ_d of TACOs mined from G such that every rule in Σ_d conforms to the requirement d.

1 $\varepsilon \leftarrow 0$; $\Sigma_d \leftarrow \emptyset$; $G_{\mathcal{M}} \leftarrow$ MLExp(G, $\mathcal{M}$);
2 $G^s_{\mathcal{M}} \leftarrow$ LocalizedSample($G_{\mathcal{M}}$, α, N);
3 TrainCreator($G^s_{\mathcal{M}}$, α, δ); /* *Pretrain the rule creator* */
4 $\Sigma \leftarrow$ GenerateRule(α, δ, Σ_d);
5 $\Sigma_d \leftarrow$ SelectRule(Σ, β, γ, G);
/* *Generate* TACOs *via interactive training* */
6 **while** $\varepsilon < I$ **do**
7 $G^s_{\mathcal{M}} \leftarrow$ LocalizedSample($G_{\mathcal{M}}$,α, N);
8 TrainCreator($G^s_{\mathcal{M}} \cup \Sigma_d$, α, δ);
9 $\Sigma \leftarrow$ GenerateRule(α, δ, Σ_d);
10 $\Sigma' \leftarrow$ SelectRule(Σ, β, γ, G);
11 $\Sigma_d \leftarrow \Sigma_d \cup \Sigma'$; $\varepsilon \leftarrow \varepsilon + 1$;
12 **return** Σ_d;

This approach is feasible since deep generative models, *e.g.,* GAN, are able to approximate various distributions [13]. Moreover, empirical research verifies that it is practical to train such models to approximate a target distribution $\mathbb{P}$ within a limited number of iterations [31], where the training loss of the model converges. CCD also optimizes rule discovery in a data-driven manner, since generative models in the rule creator are able to generate rules that are topologically and semantically similar to rules from the training data [31, 47, 110]. Meanwhile, the creator can retrain its generative models using high-quality rules selected by the critic as feedback to improve the quality of approximation.

Algorithm. Implementing the creator-critic method, CCD discovers high-quality TACOs in three phases, as shown in Algorithm 1.

(1) It first calls function MLExp to prepare graph data $G_{\mathcal{M}}$ from G, for facilitating the discovery of TACOs with ML predicates $\mathcal{M}$.

(2) Then the rule creator is pretrained using TrainCreator with the localized graph $G^s_{\mathcal{M}}$ sampled by LocalizedSample (lines 2-3), such that it starts to generate rules that are likely to reach the thresholds of support and confidence in the input requirement d. The creator generates a set Σ of candidate TACOs by GenerateRule; the critic next evaluates Σ on temporal graph G and saves high-quality TACOs in a set Σ_d via function SelectRule (lines 4-5).

(3) In the third phase, algorithm CCD iteratively discovers high-quality TACOs and adds more such rules to Σ_d via an interactive training process of I iterations (lines 6-11). Each iteration is similar to the second phase, except that both $G^s_{\mathcal{M}}$ and Σ_d are used to train and improve the creator (line 8). Here the TACOs in Σ_d are feedback from the critic. Finally Σ_d is returned (line 12).

Parameters. In addition to the discovery requirement d of four thresholds and ML models $\mathcal{M}$, CCD takes another two input parameters: sample size N and iteration number I. Here N strikes a balance between the probability of generating high-quality rules and the workload of each iteration; and I determines the size of Σ_d and the cost of model training, where larger I usually helps generate more high-quality TACOs and train the model better.

We next present the functions and models adopted in CCD.

Assistance functions. We start with two functions MLExp and LlocalizedSample, which are invoked by CCD to prepare graph data for model training. Initially, MLExp expands graph G to $G_{\mathcal{M}}$ by faithfully adding edges predicted by the input ML models $\mathcal{M}$. This allows the creator to incorporate $\mathcal{M}$ as ML predicates and accelerates TACO discovery since there is no need to repeatedly apply $\mathcal{M}$ during the discovery process.

Besides, LocalizedSample collects a set $G^s_{\mathcal{M}}$ of N Δ-patterns by sampling localized graph structures from $G_{\mathcal{M}}$, as training data for the creator in each iteration (lines 2 and 7). When deducing a pattern ΔQ_i, LocalizedSample applies temporal random walk [69] with a randomly selected source node v to sample temporal paths, where the timestamps of all edges on a temporal path fall into a given time window. It finds top $\alpha - 1$ frequently sampled nodes around v. Then ΔQ_i is formed by these nodes and the edges connecting them. Guided by the Δ-patterns in $G^s_{\mathcal{M}}$ during pretraining (line 3), the rule creator can learn to generate patterns that are more likely to find matches in the graph. This is because each ΔQ_i must have matches as it is obtained by random walk in $G_{\mathcal{M}}$, and the generative models learn to generate "new" patterns that are semantically and structurally similar to ΔQ_i. Without pretraining, randomly initialized generative models in the creator may create meaningless TACOs.

In fact, a larger sample size N allows more sampling trails with higher probability of generating good patterns and more efficient ML training with batch optimizations [63, 110]. If N is small, CCD needs to run more iterations to find adequate number of TACOs.

Rule creator. When generating a TACO φ, the creator first generates its Δ-pattern, and then the dependency (lines 4 and 9).

Δ-pattern generation. The creator generates patterns in two steps: structure generation and semantic label generation, by employing temporal graph GAN and LSTM networks, respectively.

(1) In the first step, the creator takes each Δ-pattern ΔQ_i from $G^s_{\mathcal{M}}$ and Σ_d as input; it adopts TagGen [110], an end-to-end deep generative framework for temporal graphs, to deduce candidate Δ-patterns ΔQ^g_i. Due to the GAN module in TagGen [110], each generated ΔQ^g_i and input ΔQ_i have the same number of nodes (at most α), and bear similar topology and time constraints. No labels were generated in this step as TagGen does not support labels.

(2) The creator next employs an LSTM language model $\mathcal{M}_L$ to generate labels for each ΔQ^g_i. We adopt LSTM networks since it can model the rich semantics of labels on paths in knowledge graphs [47, 52, 53]. More specifically, it first trains $\mathcal{M}_L$ on a corpus C driven by the perplexity [63], where each word is a pair $\langle L(e), L(v)\rangle$ of edge label and node label, named "label pair", and v is the destination node of e. The corpus C is composed of label pair sequences for temporal paths, which are derived by applying temporal random walk on each ΔQ^g_i. After the training, for every two nodes u and v in ΔQ^g_i with the shortest temporal path ρ from u to v, $\mathcal{M}_L$ generates a label pair sequence with a random seed, and attaches this sequence of labels to ρ. Only shortest paths are considered since they hold stronger associations [2, 39]. Finally the creator builds a Δ-pattern $\Delta Q'_i$ from ΔQ^g_i by keeping the most frequent labels attached.

Note that the creator retrains itself using the latest Σ_d to increase the probability of generating high-quality Δ-patterns

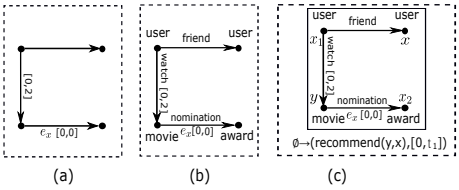


Figure 4: TACO generation process.

(line 8). The rationale behind this is that (1) in training data, the Δ-pattern ΔQ_i of a high-quality TACO in Σ_d has multiple matches; and (2) the GAN and LSTM networks enable the creator to generate new Δ-pattern $\Delta Q'_i$ that are similar to ΔQ_i in terms of topological structure, temporal constraints and label semantics [47, 52, 110]. Therefore, the generated $\Delta Q'_i$ is likely to find sufficient matches as well. The samples $G^s_{\mathcal{M}}$ returned by LocalizedSample are also used in retraining, to introduce disturbance to the creator, which prevents the ML generative models from converging at a local minimum where cliche patterns are repeatedly generated.

Dependency generation. Given a generated Δ-pattern ΔQ, the creator adapts the levelwise expansion process of [20] to construct a set of valid dependencies $X \rightarrow (p_0, \tau)$. More specifically, for each possible event (p_0, τ), it starts with $X = \emptyset$ and iteratively extends the precondition X for ΔQ. But unlike [20] that directly verifies the validity of each candidate TACO $\varphi = \Delta Q[\bar{x}](X \rightarrow (p_0, \tau))$ after its generation, the creator first checks whether φ is redundant, *i.e.,* whether φ is implied by the set Σ_d of TACOs discovered in the previous iterations. If φ is not redundant, *i.e.,* $\Sigma_d \not\models \varphi$, it proceeds with the validation of φ by the critic. We find that checking implication in advance helps reduce the discovery cost, since implication is conducted on Σ_d, which is much smaller than the graph G. In addition, we perform satisfiability checking of the TACOs in Σ and newly discovered candidate rules at this stage, to avoid further validation of inconsistent TACOs in G. This process terminates when no more predicates can be added. The termination is guaranteed since predicates in X are defined on (a) nodes in ΔQ and (b) attributes and values (including timestamps) in updates ΔG_Q; hence the number of all possible predicates is bounded by the sizes of ΔQ and ΔG_Q.

Rule critic. For each generated candidate TACO φ in Σ that passes the implication and satisfiability checking, the rule critic computes $\text{supp}(\varphi, G)$ and $\text{conf}(\varphi, G)$, and selects high-quality TACOs whose support and confidence are above the thresholds (SelectRule in lines 5 and 10). These rules are added to Σ_d and are provided to the rule creator for improving generative models in the next iteration. Computing $\text{supp}(\varphi, G)$ and $\text{conf}(\varphi, G)$ is efficient since the procedures can be parallelized with optimizations that are unique to temporal pattern matching (see Section 6) and we use existing efficient subgraph matching method, *i.e.,* DAF [32] and its proposed CS structures to reduce redundant computation.

Example 5: Continuing with Example 4, Figure 4 shows the generation of TACO φ_1 from temporal graph G. First, the creator calls TagGen to create a candidate Δ-pattern (Fig. 4(a)) with time constraints. Then, as shown in Fig. 4(b), the LSTM model $\mathcal{M}_L$ is employed to add labels to nodes and edges with semantic meanings. The creator completes the generation by constructing dependencies with variables (Fig. 4(c)). Validated by the rule critic in G, this generated candidate rule is preserved in Σ_d, since its support and

confidence are both 1. Note that if the edge labeled with nomination is dropped from φ_1, the confidence would reduce to 0.5. □

Performance guarantees. We next show that when the iteration number I is sufficiently large, CCD can return all TACOs that satisfy the requirement d with a high probability. Recall that when accumulating the training data in each round for the generative model, CCD conducts temporal random walk to sample subgraphs by function LocalizedSample. Based on this, we have the following.

Theorem 4: *Given graph G, requirement d and constant $\epsilon \in (0, 1)$, after* $\frac{|G|^\alpha}{N\beta}\left(1-\ln\frac{\beta\epsilon}{|G|^\alpha}+\sqrt{\ln\frac{\beta\epsilon}{|G|^\alpha}(\ln\frac{\beta\epsilon}{|G|^\alpha}-2)}\right)$ *iterations,* CCD *can discover all* TACOs *satisfying d with probability at least* $1-\epsilon$. □

Proof sketch: Since the levelwise expansion process of [20] is adapted to build dependencies, it suffices to show that CCD can sample all Δ-patterns that satisfy d with probability $1-\epsilon$, after sufficient iterations as given above. Using Chernoff–Hoeffding bound [17] and bonferroni inequality [9], we prove this by analyzing the expected number of Δ-patterns mined across the iterations. □

Cost analysis. To see the cost of CCD, observe the following. Function LocalizedSample takes $O(|G|)$ time for applying temporal random walk [69]. The cost for model training in TrainCreator is linear to the number of training samples, *i.e.,* $O(N + |\Sigma_d|)$. GenerateRule (creator) takes time polynomial in the size $|G|$ to generate Δ-patterns and dependencies. However, SelectRule (*i.e.,* critic) takes $O(|G|^\alpha)$ time to compute the support and confidence of each candidate TACO because of the graph homomorphism in temporal matching; nonetheless, we make use of parallelism (see Section 6) and the auxiliary structure in [32] to speed up the computation.

Remark. (1) While the theoretical iteration number I may be large, in practice, when sample size N is set 250 by default, the generative models converge within 25 iterations (*i.e., I*=25), where extra iterations add few novel TACOs to Σ_d (see Section 7). Therefore, we set I using a practical small value. This helps us reduce the overall cost and justifies the usage of deep generative models.

(2) The users may opt to inspect the generated TACOs in each round, select rules of their interest, and add to Σ_d. This incorporates user interests into discovery. They may also terminate the iterative process manually when they are satisfied with the TACOs in Σ_d so far.

6 PARALLEL EVENT PREDICTION

In this section we develop a parallel algorithm to support the event prediction module of TASTE. The algorithm is also used to compute support and confidence for the rule critic of algorithm CCD (Section 5). We propose a partitioning strategy for temporal graphs, and show that the algorithm guarantees the parallel scalability.

We start with a sequential prediction approach with TACOs.

Sequential algorithm. Given a temporal graph G, a set Σ of TACOs and a time window τ_0, a sequential prediction algorithm, denoted as SeqEP, finds all edges (events) R predicted by Σ in G as follows. For each TACO $\varphi = \Delta Q[\bar{x}](X \rightarrow (p_0, \tau))$ in Σ with $\tau = [t_1, t_2]$, SeqEP (1) finds all matches of ΔQ in ΔG_Q via graph homomorphism; and (2) for each such match $h(\bar{x})$, it checks whether $h(\bar{x}) \models X$ and the time window $[t^* + t_1, t^* + t_2]$ deduced is a subinterval of τ_0; here t^* is the current time (Section 2.1); if so, SeqEP adds to R the edge that links the nodes matching the variables in p_0.

Note that SeqEP is applied to subgraphs ΔG_Q of G for ΔQ in Σ, which is typically much smaller than the entire graph G. In the sequel we refer to the union of such ΔG_Q's simply as G. Even so, the exponential cost of graph homomorphism [27] in temporal pattern matching (step (1)) motivates us to parallelize SeqEP.

Parallel scalability. To measure the effectiveness of parallelization, we adapt the criterion introduced by [43] to graph computation. Consider a problem $\mathcal{P}$ posed on a graph G. We denote by $T_s(|I_\mathcal{P}|, |G|)$ the worst-case complexity of a sequential algorithm $\mathcal{F}$ for handling an instance $I_\mathcal{P}$ of problem $\mathcal{P}$ over G. For a parallel algorithm $\mathcal{F}_p$, we denote by $T_p(|I_\mathcal{P}|, |G|, k)$ the time taken by it for processing problem instance $I_\mathcal{P}$ on G using k processors. We say that algorithm $\mathcal{F}_p$ is *parallelly scalable relative to* $\mathcal{F}$ if

$$T_p(|I_\mathcal{P}|, |G|, k) = O\big(\frac{T_s(|I_\mathcal{P}|, |G|)}{k}\big)$$

for any instance $I_\mathcal{P}$. That is, the parallel algorithm $\mathcal{F}_p$ achieves a "linear" reduction in sequential running time of a yardstick algorithm $\mathcal{F}$, allowing us to process large graphs by adding resources.

Parallelizing event prediction. A typical strategy for parallelizing sequential graph computation is to first partition a graph into k small fragments, and then conduct the computation over fragments at k processors in parallel with necessary message passing, *e.g.,* the deduction algorithm with GARs [21]. Following this paradigm, we could partition a temporal graph via an existing graph partitioning method, *e.g.,* edge-cut or vertex-cut [3, 77]. However, this easily incurs a large amount of communication in the subsequent parallel prediction. This is because most of the previous partitioning methods aim to minimize the number of cut edges or vertices but overlook the timestamps, which are crucial to temporal pattern matching; the edges in a match of a Δ-pattern are often partitioned into different fragments and demand communication.

To rectify this problem, we partition a temporal graph G based on *temporal locality*, a unique property of temporal pattern matching with TACOs. That is, the timestamps of edges in each match $h(\bar{x})$ of Δ-pattern ΔQ are within the range of localized timespan $[t^* + t_{\mathsf{min}}, t^* + t_{\mathsf{max}}]$, where t_{min} (resp. t_{max}) is the minimum (resp. maximum) timestamp in ΔQ as stated in Section 2.1.

Temporal partitioning. We propose a temporal partitioning strategy with which parallel event prediction can be made communication-free. Intuitively, it divides a time interval into k subintervals and guarantees that every specific range of timestamps for finding match $h(\bar{x})$ is *entirely* covered by one subinterval. Guided by the resulting subintervals, the temporal graph G is partitioned into k fragments $F_1, \ldots, F_k$, such that each F_i consists of the edges whose timestamps are within one subinterval. By the temporal locality, temporal pattern matching and hence event prediction can be conducted in parallel over such F_i's with no communication.

The cost of parallel event prediction is determined by the maximum size $\max_{i\in[1,k]}|F_i|$ of fragments. Thus we want to find a good division of the time interval to minimize $\max_{i\in[1,k]}|F_i|$. To do this, we develop function BTPart, shown as part of Algorithm 2. It takes as input a candidate time interval $[t_0, t_k)$ that is deduced from the TACOs Σ and time window τ_0 for matching designated

Algorithm 2: Parallel Event Prediction (ParEP)

Input: The number k of processors, a k-way randomly partitioned temporal graph G, a set Σ of TACOs, a time window τ_0.
Output: A set R of edges predicted by Σ with events occur within τ_0.

1 $[t_0, t_k) \leftarrow$ RefTime(Σ, τ_0); $(t_{\min}, t_{\max}) \leftarrow$ ExtractTS(Σ);
2 collect the size $|G_t|$ of t-graph G_t for $t \in [t_0, t_k)$;
3 $\{t_1, \ldots, t_{k-1}\} \leftarrow$ BTPart($[t_0, t_k)$, $\{|G_t| \mid t \in [t_0, t_k)\}$, k, $t_{\min}$, $t_{\max}$);
4 $F_i \leftarrow G_{[t_{i-1}+t_{\min}, t_i+t_{\max})}$ for each $i \in [1, k]$;
5 RBalance($\{F_i \mid i \in [1, k]\}$);
6 run SeqEP(F_i, Σ, τ_0) at each fragment F_i to get R_i for $i \in [1, k]$;
7 **return** $\bigcup_{i\in[1,k]} R_i$;

Function BTPart($[t_0, t_k)$, $\{|G_t| \mid t \in [t_0, t_k)\}$, k, $t_{\min}$, $t_{\max}$):

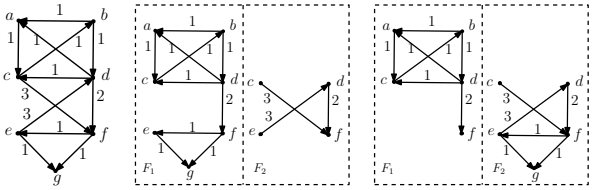
1 **foreach** $t \in [t_0, t_k)$ **do**
2 $S[t][1] \leftarrow |G_{[t_0+t_{\min}, t+t_{\max})}|$;
3 **foreach** $i \in [2, k]$ **do**
4 **foreach** $t' \in [t_0, t_k)$ **do**
5 $S[t'][i] \leftarrow \min_{t\in[t_0,t')} \max(S[t][i-1], |G_{[t+t_{\min}, t'+t_{\max})}|)$;
6 **foreach** $i \in [2, k]$ in descending order **do**
7 $t_{i-1} \leftarrow \arg\min_{t\in[t_0,t_i)} \max(S[t][i-1], |G_{[t+t_{\min}, t_i+t_{\max})}|)$;
8 **return** $\{t_1, \ldots, t_{k-1}\}$;

edges in event prediction (see below), a set $\{|G_t| \mid t \in [t_0, t_k)\}$ of sizes of t-graphs, the number k of fragments (processors), and the maximum (resp. minimum) timestamp $t_{\max}$ (resp. $t_{\min}$) in Σ. Here G_t refers to a t-graph that is composed of all edges in G bearing timestamp t. BTPart computes a set $\{t_1, \ldots, t_{k-1}\}$ of $k-1$ partition points for the interval $[t_0, t_k)$ such that fragment F_i becomes $G_{[t_{i-1}+t_{\min}, t_i+t_{\max})}$ for $i \in [1, k]$, which includes the edges of G with timestamps in the range $[t_{i-1} + t_{\min}, t_i + t_{\max})$. Here $t_{\min}$ and $t_{\max}$ are to ensure the entire coverage as mentioned above.

Procedure BTPart adopts dynamic programming. It maintains a 2D array S, where $S[t][i]$ records the minimum size of the largest fragments that are obtained by partitioning $G_{[t_0+t_{\min}, t+t_{\max})}$ into i subintervals. Hence $S[t_k][k]$ is our objective value. BTPart first handles the base case i=1 (lines 1-2), where $S[t][1]$=$|G_{[t_0+t_{\min}, t+t_{\max})}|$ since there is only one fragment *w.r.t.* the single subinterval. Here the size is derived from that of the input t-graphs. For cases where i>1, $S[t'][i]$ is determined by checking the values regarding all possible ranges $[t_0, t)$ for t<t' and their $i-1$ subintervals (lines 3-5). After $S[t_k][k]$ is computed, the corresponding $k-1$ partition points can be identified and are returned as the result (lines 6-8).

By induction on the iterations, one can verify that the returned partition points yield minimum $\max_{i\in[1,k]} |F_i|$ (*i.e.,* $S[t_k][k]$).

Parallel algorithm. Capitalizing on the temporal partitioning, we develop a parallel prediction algorithm, denoted by ParEP. As shown in Algorithm 2, initially ParEP uses function RefTime to deduce a time interval $[t_0, t_k)$ for those edges in G that can potentially match designated edges in TACOs Σ (line 1). Since we only predict edges in time window τ_0, a timestamp t is in $[t_0, t_k)$ if and only if the gap between t and τ_0 is smaller than that of τ for some (p_0, τ) in Σ. It also extracts maximum and minimum timestamps from Σ via ExtractTS (line 1). Then ParEP performs temporal partitioning to get k fragments F_1 to F_k and each F_i is assigned to a distinct processor (lines 2-4). It designates one processor as the coordinator to collect the sizes of t-graphs and apply function BTPart.



(a) G (b) A temporal partition (c) A rebalanced partition

Figure 5: Temporal partitioning and rebalancing

To further balance the workload of parallel prediction, ParEP next redistributes the data in large fragments F_i having $|F_i| > |G|/k$ by function RBalance (line 5). More specifically, for each large F_i, it evenly partitions those edges of F_i that can match the designated edges in Σ; while the set of candidate matches in F_i for other pattern edges in Σ are replicated at all processors. Here candidate matches are identified via label comparison. Then SeqEP is invoked at all processors in parallel with no communication to get the predicted edges, and their union is finally returned (lines 6-7).

Example 6: Consider partitioning the temporal graph G shown in Figure 5(a) into 2 fragments, where only the timestamp of each edge is marked while labels are omitted to simplify the discussion. Suppose that $(t_{\min}, t_{\max}) = (0, 1)$ for a given set Σ of TACOs and the candidate time interval $[t_0, t_k)$ deduced from Σ, and the input time window τ_0 is $[1, 4)$. We can see the following.

(1) The temporal partition generated by BTPart is shown in Figure 5(b). Here the candidate interval is divided into $[1, 2)$ and $[2, 4)$; thus fragments F_1 and F_2 have edges whose timestamps are within $[1, 2+1)$ and $[2, 4+1)$, respectively. Note that the edge (d, f) with timestamp 2 is replicated at both fragments. This ensures that all patterns in Σ can be matched on F_1 or F_2 locally.

(2) While Figure 5(b) is an optimal temporal partition, it is skewed, *i.e.,* the edges with timestamp 1 in F_1 make a large part of G. By moving edges $(f, e), (e, g)$ and (f, g) to fragment F_2 via function ReBalance, the partition becomes balanced (Figure 5(c)). □

The parallel scalability of ParEP is assured as follows, where the sequential SeqEP takes $O(|\Sigma||G|^{|\Sigma|})$ time in the worst case and the cost of ParEP is bounded by $O(|\Sigma|\frac{|G|^{|\Sigma|}}{k})$.

Theorem 5: ParEP *is parallelly scalable relative to* SeqEP. □

Remark. (1) ParEP can be readily adapted to predict whether a particular event p_0 will happen and when it will take place, by allowing users to set the range τ_0 and applying relevant TACOs in Σ. (2) We use ParEP to compute the support and confidence of TACOs, (*i.e.,* the rule critic in CCD). To do these, the input time window τ_0 is defined as the smallest range that covers all the timestamps in G.

7 EXPERIMENTAL STUDY

Using real-life and synthetic graphs, we experimentally evaluated (1) the efficiency and (2) the quality of the creator-critic rule discovery method, (3) the accuracy of TASTE for event prediction and dynamic recommendation, and (4) the (parallel) scalability of ParEP.

Experimental setting. We start with the experimental setting.

Datasets. We used six real-life temporal graph benchmark datasets, shown in Table 2 and classified into three different classes: (1) event-

Table 2: Datasets

Dataset	$\|V\|$	$\|E\|$	Relation type	Timestamp interval
ICEWS18	23K	469K	256	1 day
GDELT	8K	2.2M	240	15 minutes
YAGO	11K	201K	10	1 year
WIKI	13K	670K	24	1 year
MovieLens	80K	10M	10	1 day
Amazon	12.2M	30.3M	5	1 day

based temporal knowledge graphs: ICEWS18 [36] from the Integrated Crisis Early Warning System [8], and GDELT [36] from the Global Database of Events, Language, and Tone [46]; (2) knowledge graphs with temporally associated facts: YAGO [57] and WIKI [45], *e.g.,* YAGO records the time when a football player plays for a club; and (3) dynamic recommendation datasets: MovieLens [34] of movie ratings and Amazon [61] of product ratings, where original timestamps were reorganized such that the time granularity between two adjacent timestamps is one day.

As designed in [36, 50], each dataset has been divided into training, validation and test sets, with proportion of 80%, 10% and 10%, respectively, by timestamps. The training and validation sets were used for model training and rule discovery, while the test set was for accuracy test. Each dataset includes ground truth of event (temporal edge) prediction results [36, 50], and the testing set actually poses the set of "queries" *w.r.t.* predicting temporal events.

We also designed a graph generator to create larger synthetic datasets, for evaluating the scalability. The synthetic graphs had up to $10M$ nodes and $1B$ edges in the range of 10000 timestamps, with labels, attributes and values drawn from 200 symbols.

Algorithms. The creator and critic in the rule discovery module were implemented in Pytorch and C++, respectively, while the temporal event prediction module was implemented in C++. We compared TASTE with five baseline methods for event prediction: (1) AGER, which applies GERs [5] that capture local changes in temporal graphs for event prediction; (2) SACN [86], a convolution-based embedding approach for knowledge graph completion; (3) REGCN [50], a knowledge graph reasoning method based on Graph Convolution Network; (4) Caser [91], a sequential recommendation algorithm based on convolutional neural networks; and (5) SASRec [37], a transformer-based sequential recommender system. We adopted open-source codes of SACN [85], REGCN [49], Caser [90] and SASRec [38] with default configurations, and implemented AGER in C++. We have also implemented two levelwise search-based rule mining methods to discover GERs and TACOs in C++, denoted as GERMine and TACOMine, as discovery baselines.

The baselines SACN, REGCN, Caser and SASRec were parallelized with multi-threads by PyTorch. For AGER, GERMine and TACOMine, we applied the same parallelization method of ParEP (Section 6) to compute the corresponding matches, which dominate their costs. The thread number in parallelization is equal to the number k of cores used by TASTE, for a fair comparison.

ML models. REGCN and SASRec were adopted as the ML predicates in TACOs for temporal graph completion and dynamic recommendation, respectively. For models in the creator of CCD, we used the code of TagGen provided by the authors with default configurations [111], and implemented the LSTM model as [63] with its default training configuration and two 650-wide layers.

Table 3: Quality of the creator-critic discovery on ICEWS18

I \ N	50	100	150	200	250	300
15	20.00%	23.81%	28.57%	39.05%	48.57%	65.71%
20	23.81%	25.71%	40.95%	55.24%	66.67%	73.33%
25	30.47%	41.90%	44.76%	71.43%	84.76%	96.19%
30	39.05%	44.28%	48.94%	76.15%	87.23%	97.14%

We conducted experiments on a cluster of up to 72 Intel Xeon 3.1 GHz processing cores on two machines connected by 10Gbps links, with 256GB memory. By default we set α=5, β=100, γ=0.8 and δ=20 for discovery requirement; the number of iterations I=25 and sample size N=250 for discovery module CCD; and the number of cores $k = 32$ for prediction method ParEP, unless stated otherwise. All the experiments were repeated 5 times. The average is reported here.

Experimental results. We next report our findings. In every experiment, the results on at least one graph from each of the three classes are shown. We defer the other results to the full version.

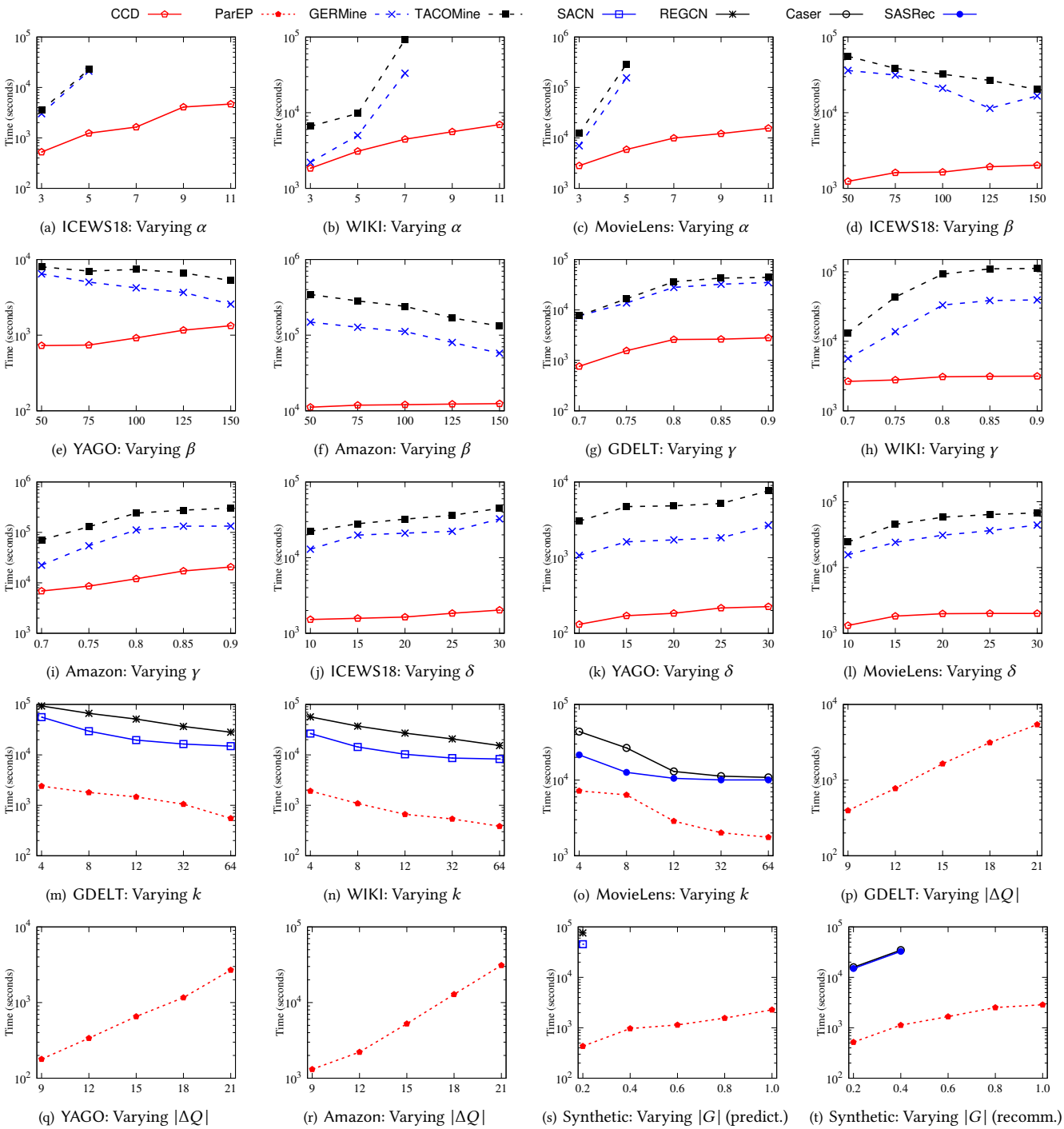
Exp-1: Efficiency. We first compared the efficiency of CCD, GERMine and TACOMine for rule discovery. Since it is very costly for levelwise methods to discover rules with large patterns, in order to compare efficiency within bearable running time, we set a target of discovering 100 high-quality rules as benchmark. That is, each discovery process terminated when 100 rules had been discovered.

(1) Varying α. We varied α from 3 to 11 to study the impact of pattern node number on discovery methods using ICEWS18, WIKI and MovieLens. As shown in Figures 6(a) to 6(c), CCD is on average 9.1 and 14.3 times faster than GERMine and TACOMine when $\alpha \leq 5$, respectively. The computation time of levelwise search-based methods grows exponentially as α gets larger, and both GERMine and TACOMine cannot terminate in 1.2 days when $\alpha > 5$, while the cost increase of CCD is mild. This is because larger number of pattern nodes incurs exponentially larger search space for levelwise mining, but little extra cost for ML generative models. In particular, CCD finds TACOs with patterns of more than 20 edges in 1639s.

(2) Varying β. We varied β from 50 to 150, to study the impact of support threshold over ICEWS18, YAGO and Amazon. As reported in Figures 6(d) to 6(f), the runtime of CCD does not change much, as it always takes multiple iterations for the creator to generate high-quality rules regardless of the value of β. In contrast, GERMine and TACOMine take less time with larger β in most cases, since higher bound on support prunes more candidates and reduces search space.

(3) Varying γ. Varying confidence γ from 0.7 to 0.9, we report the results on GDELT, WIKI and Amazon in Figures 6(g) to 6(i), respectively. It is shown that the all discovery algorithms take longer time with the increase of γ. However, the generation-based CCD is less sensitive to γ. This is because the levelwise methods have to expand their search space at an exponential scale to get the requested number of high-quality rules with higher confidence.

(4) Varying δ. We varied δ from 10 to 30. Here δ counts the number of discrete timestamps in the required time window. As shown in Figures 6(j) to 6(l) on ICEWS18, YAGO and MovieLens, respectively,



(a) ICEWS18: Varying α (b) WIKI: Varying α (c) MovieLens: Varying α (d) ICEWS18: Varying β

(e) YAGO: Varying β (f) Amazon: Varying β (g) GDELT: Varying γ (h) WIKI: Varying γ

(i) Amazon: Varying γ (j) ICEWS18: Varying δ (k) YAGO: Varying δ (l) MovieLens: Varying δ

(m) GDELT: Varying k (n) WIKI: Varying k (o) MovieLens: Varying k (p) GDELT: Varying $|\Delta Q|$

(q) YAGO: Varying $|\Delta Q|$ (r) Amazon: Varying $|\Delta Q|$ (s) Synthetic: Varying $|G|$ (predict.) (t) Synthetic: Varying $|G|$ (recomm.)

Figure 6: Performance evaluation

the runtime for all three increases as δ grows, since a longer time window bound gives more generation workload for TagGen in the creator and expands the search space for GERMine and TACOMine.

(5) Impact of N and I. We also tested the impact of the sample size N and iteration number I on the efficiency of CCD. It exhibits a moderately runtime increase with the increase of N or I (not shown). This is as expected, since a larger N (resp. I) causes more work for the generator (resp. more rounds of the entire computation).

Exp-2: Quality of discovery. Recall that TACOs obtained by the generation-based CCD are a subset of those returned by levelwise search algorithms. Thus we checked how many rules in the com-

Table 4: Event prediction/recommendation accuracy

Dataset	AGER	SACN	REGCN	Caser	SASRec	TASTE
ICEWS18	59.32%	63.28%	68.58%	-	-	75.24%
GDELT	54.60%	62.32%	66.31%	-	-	73.41%
YAGO	61.32%	64.03%	74.73%	-	-	84.14%
WIKI	58.56%	63.58%	71.32%	-	-	82.53%
MovieLens	65.80%	-	-	75.83%	80.45%	87.30%
Amazon	59.30%	-	-	63.50%	73.10%	73.10%

plete set found by levelwise method TACOMine can be discovered by CCD in all the datasets, *i.e.,* the coverage of complete TACOs. As reported in Table 3, the coverage becomes higher with larger size N of samples or more training iterations I. This is because enlarging either N or I could increase the possibility of approximating the target distribution and hence generating high-quality rules, as discussed in Section 5. We can also see that small N and I suffice to get a large portion of high-quality TACOs, *e.g.,* the coverage reaches 84.76% when $N = 250$ and $I = 25$, while the time of TACOMine is reduced by 18.5 times simultaneously (see Exp-1).

In addition to the adopted GAN model, we tested the performance of using classic graph generation models, *i.e.,* Erdös-Rényi (ER) [72] and Barabási-Albert (BA) [1] models, in CCD. We find that when $N = 300$ and $I = 30$, the coverage vlaues of the TACOs found with ER and BA are merely 33.07% and 50.89%, respectively, much lower than that by GAN. This is because these classical models cannot iteratively learn from the graph data and generate high-quality patterns in an adaptive manner.

We also manually checked the discovered TACOs. Besides the typical ones that include logic and temporal predicates only, some TACOs can also help enrich or interpret ML predictions.

Exp-3: Accuracy. As shown in Table 4, we evaluated the accuracy of TASTE with the TACOs and baselines on two tasks: temporal event prediction and dynamic recommendation [37, 50]. Note that there was no result for SACN and REGCN (resp. Caser and SASRec) on dynamic recommendation (resp. temporal event prediction) datasets since they are not designed for the task. TASTE (ParEP) applied the discovered TACOs with confidence above 0.9, in which graph patterns have at most 9 nodes. We find that very few TACOs with more than 9 pattern nodes have high confidence and support, similar to the findings of frequent pattern mining [19]. We adopted Hit Rate@10, the fraction of times that the ground-truth item is among the top 10 items [37, 50], to evaluate the accuracy.

For event prediction on ICEWS18, GDELT, YAGO and WIKI, TASTE on average outperforms AGER, SACN and REGCN by 34.9%, 24.5% and 12.2%, respectively. As for the dynamic recommendation on MovieLens and Amazon, TASTE is 35.8%, 22.5% and 10.6% more accurate than AGER, Caser and SASRec, respectively. These show that by combining rules and ML models, TASTE beats the state-of-the-art deep-learning-based REGCN and SASRec in accuracy on both tasks, while none of REGCN and SASRec works on both.

Exp-4: Scalability. We finally evaluated (1) the parallel scalability of the prediction module ParEP in TASTE system by varying the number k of processors, (2) the impact of pattern size on its efficiency, and (3) the scalability of ParEP over larger synthetic graphs. Since GERs are a special case of TACOs, ParEP is also applicable to GERs and the runtime of AGER is not shown.

(1) Parallel scalability. Varying k from 4 to 64, Figures 6(m)-6(o) report the results on GDELT and WIKI for event prediction, and on MovieLens for recommendation in the same setting as Exp-3, respectively. As shown there, (a) ParEP is parallelly scalable. When k increases from 4 to 32, it is on average 3.2 times faster on the three graphs. This verifies the effectiveness of ParEP under data-partitioned parallelism. (b) In addition to its higher accuracy, ParEP performs better in efficiency than SACN and REGCN (resp. Caser and SASRec), *e.g.,* when $k = 64$, it is on average 24.2, 45.4, 6.2 and 5.7 times faster than SACN, REGCN, Caser and SASRec, respectively.

(2) Impact of pattern size. Varying the size $|\Delta Q|$ of Δ-patterns in TACOs, which is measured as the sum of the pattern node and edge numbers in each ΔQ, we report the performance of different methods in Figures 6(p) to 6(r). The results show that ParEP becomes slower with the increase of $|\Delta Q|$, as expected. Nonetheless, it is still efficient when handling relatively large Δ-patterns. For instance, it needs 1645 seconds on GDELT when $|\Delta Q| = 15$, which is better than 16380 seconds by SACN (see Figure 6(m)).

(3) Scalability. Fixing $k = 32$, we varied the size $|G|=|V|+|E|$ of synthetic graphs G using a scale factor from 0.2 to 1.0, and evaluated all approaches, where ParEP applied 100 TACOs. As shown in Figures 6(s) and 6(t) for prediction and recommendation, respectively, ParEP outperforms the baselines in all cases. On average it takes 1403s when $|G|$=810M, while the others cannot finish in 1 day.

Summary. We find the following. (1) On average the generative ML method of CCD outperforms the levelwise algorithms in efficiency by more than 31 times. It is able to discover TACOs with patterns of 20 edges in 1639s from temporal graphs, while the levelwise methods could not finish in 1.2 days. (2) CCD is able to find as high as 84.76% of the complete rules derived by levelwise method, using 250 samples and 25 training attempts. (3) By combining rules and ML models, on average the discovered TACOs improve the existing approaches by 23.8% and 23.0% in accuracy, for event prediction and dynamic recommendation, respectively. (4) Our algorithm ParEP is parallelly scalable and scales well with the datasets, it takes less than 1403s on graphs with 810M nodes and edges using 32 processors.

8 CONCLUSION

We have proposed a new approach towards event prediction, from foundation (rules and complexity) to a system (supported by scalable algorithms), with novelty summarized in Section 1. We have experimentally verified that TASTE is promising in event prediction.

One topic for future work is to evaluate TASTE for predicting events of other types, *e.g.,* finance crisis. Another topic is to make TASTE "real-time" by incrementally discovering rules and predicting events in response to updates to temporal graphs.

ACKNOWLEDGMENTS

This work was supported by ERC 652976, Royal SocietyWolfson Research Merit Award WRM/R1/180014, and NSFC 61902274. Xu is supported by the National Research Foundation, Singapore under its Strategic Capability Research Centres Funding Initiative. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore

REFERENCES

[1] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics* 74, 1 (2002), 47.

[2] Boanerges Aleman-Meza, Christian Halaschek-Wiener, Ismailcem Budak Arpinar, and Amit P. Sheth. 2003. Context-Aware Semantic Association Ranking. In *SWDB*.

[3] Konstantin Andreev and Harald Räcke. 2006. Balanced Graph Partitioning. *Theory Comput. Syst.* 39, 6 (2006), 929–939.

[4] Marcelo Arenas, Pedro Bahamondes, and Julia Stoyanovich. 2021. Temporal Regular Path Queries: Syntax, Semantics, and Complexity. *CoRR* abs/2107.01241 (2021).

[5] Michele Berlingerio, Francesco Bonchi, Björn Bringmann, and Aristides Gionis. 2009. Mining Graph Evolution Rules. In *ECML/PKDD*.

[6] Yuemin Bian and Xiang-Qun Xie. 2021. Generative chemistry: Drug discovery with deep learning generative models. *Journal of Molecular Modeling* (2021).

[7] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. 2018. NetGAN: Generating Graphs via Random Walks. In *ICML*.

[8] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. 2015. ICEWS coded event data. *Harvard Dataverse* 12 (2015).

[9] George Casella and Roger Berger. 2001. *Statistical Inference*. Duxbury Resource Center.

[10] Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. 2020. Machine learning on graphs: A model and comprehensive taxonomy. *CoRR* abs/2005.03675 (2020).

[11] Jinyin Chen, Xuanheng Xu, Yangyang Wu, and Haibin Zheng. 2018. GC-LSTM: Graph Convolution Embedded LSTM for Dynamic Link Prediction. *CoRR* abs/1812.04206 (2018).

[12] Alvaro Cortés-Calabuig and Jan Paredaens. 2012. Semantics of Constraints in RDFS. In *AMW*.

[13] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath. 2018. Generative Adversarial Networks: An Overview. *IEEE Signal Process. Mag.* 35, 1 (2018), 53–65.

[14] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2018. Go for a Walk and Arrive at the Answer: Reasoning Over Paths in Knowledge Bases using Reinforcement Learning. In *ICLR*.

[15] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha P. Talukdar. 2018. HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding. In *EMNLP*.

[16] Walter H. Dempsey, Alexander Moreno, Christy K. Scott, Michael L. Dennis, David H. Gustafson, Susan A. Murphy, and James M. Rehg. 2017. iSurvive: An Interpretable, Event-time Prediction Model for mHealth. In *ICML*.

[17] Devdatt P. Dubhashi and Alessandro Panconesi. 2009. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press.

[18] Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. 2011. Temporal Link Prediction Using Matrix and Tensor Factorizations. *ACM Trans. Knowl. Discov. Data* 5, 2 (2011), 10:1–10:27.

[19] Mohammed Elseidy, Ehab Abdelhamid, Spiros Skiadopoulos, and Panos Kalnis. 2014. GRAMI: Frequent Subgraph and Pattern Mining in a Single Large Graph. *PVLDB* 7, 7 (2014), 517–528.

[20] Wenfei Fan, Chunming Hu, Xueli Liu, and Ping Lu. 2020. Discovering Graph Functional Dependencies. *ACM Trans. Database Syst.* 45, 3 (2020), 15:1–15:42.

[21] Wenfei Fan, Ruochun Jin, Muyang Liu, Ping Lu, Chao Tian, and Jingren Zhou. 2020. Capturing Associations in Graphs. *PVLDB* 13, 11 (2020), 1863–1876.

[22] Wenfei Fan and Ping Lu. 2019. Dependencies for Graphs. *ACM Trans. Database Syst.* 44, 2 (2019), 5:1–5:40.

[23] Sébastien Frémal and Fabian Lecron. 2017. Weighting strategies for a recommender system using item clustering based on genres. *Expert Syst. Appl.* 77 (2017), 105–113.

[24] Kaiqun Fu, Taoran Ji, Liang Zhao, and Chang-Tien Lu. 2019. TITAN: A Spatiotemporal Feature Learning Framework for Traffic Incident Duration Prediction. In *SIGSPATIAL/GIS*.

[25] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: Association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*.

[26] Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. 2018. Learning Sequence Encoders for Temporal Knowledge Graph Completion. In *EMNLP*.

[27] Michael Garey and David Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[28] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. 2020. Diachronic Embedding for Temporal Knowledge Graph Completion. In *AAAI*.

[29] Palash Goyal, Sujit Rokka Chhetri, and Arquimedes Canedo. 2020. dyngraph2vec: Capturing network dynamics using dynamic graph representation learning. *Knowl. Based Syst.* 187 (2020).

[30] Palash Goyal, Nitin Kamra, Xinran He, and Yan Liu. 2018. DynGEM: Deep Embedding Method for Dynamic Graphs. *CoRR* abs/1805.11273 (2018).

[31] Jie Gui, Zhenan Sun, Yonggang Wen, Dacheng Tao, and Jieping Ye. 2020. A Review on Generative Adversarial Networks: Algorithms, Theory, and Applications. *CoRR* abs/2001.06937 (2020).

[32] Myoungji Han, Hyunjoon Kim, Geonmo Gu, Kunsoo Park, and Wook-Shin Han. 2019. Efficient Subgraph Matching: Harmonizing Dynamic Programming, Adaptive Matching Order, and Failing Set Together. In *SIGMOD*.

[33] Negar Hariri, Bamshad Mobasher, and Robin D. Burke. 2012. Context-aware music recommendation based on latenttopic sequential patterns. In *RecSys*.

[34] F. Maxwell Harper and Joseph A. Konstan. 2016. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4 (2016), 19:1–19:19.

[35] Yuriy Hulovatyy, Huili Chen, and Tijana Milenkovic. 2015. Exploring the structure and function of temporal networks with dynamic graphlets. *Bioinform.* 31, 12 (2015), 171–180.

[36] Woojeong Jin, Meng Qu, Xisen Jin, and Xiang Ren. 2020. Recurrent Event Network: Autoregressive Structure Inference over Temporal Knowledge Graphs. In *EMNLP*.

[37] Wang-Cheng Kang and Julian J. McAuley. 2018. Self-Attentive Sequential Recommendation. In *ICDM*.

[38] Wang-Cheng Kang and Julian J. McAuley. 2019. SASRec implementation. *https://github.com/kang205/SASRec*.

[39] Yoed N Kenett, Effi Levi, David Anaki, and Miriam Faust. 2017. The semantic distance task: Quantifying semantic distance with semantic network path length. *Journal of Experimental Psychology: Learning, Memory, and Cognition* (2017).

[40] Tushar Khot, Sriraam Natarajan, Kristian Kersting, and Jude W. Shavlik. 2011. Learning Markov Logic Networks via Functional Gradient Boosting. In *ICDM*.

[41] Angelika Kimmig, Bart Demoen, Luc De Raedt, Vítor Santos Costa, and Ricardo Rocha. 2011. On the implementation of the probabilistic logic programming language ProbLog. *Theory Pract. Log. Program.* 11, 2-3 (2011), 235–262.

[42] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*.

[43] Clyde P. Kruskal, Larry Rudolph, and Marc Snir. 1990. A Complexity Theory of Efficient Parallel Algorithms. *Theor. Comput. Sci.* 71, 1 (1990), 95–132.

[44] Ni Lao, Tom M. Mitchell, and William W. Cohen. 2011. Random Walk Inference and Learning in A Large Scale Knowledge Base. In *EMNLP*.

[45] Julien Leblay and Melisachew Wudage Chekol. 2018. Deriving Validity Time in Knowledge Graph. In *WWW*.

[46] Kalev Leetaru and Philip A Schrodt. 2013. Gdelt: Global data on events, location, and tone, 1979–2012. In *ISA annual convention*.

[47] Manling Li, Qi Zeng, Ying Lin, Kyunghyun Cho, Heng Ji, Jonathan May, Nathanael Chambers, and Clare R. Voss. 2020. Connecting the Dots: Event Graph Schema Induction with Path Language Modeling. In *EMNLP*.

[48] Yang Li, Nan Du, and Samy Bengio. 2018. Time-Dependent Representation for Neural Event Sequence Prediction. In *ICLR*.

[49] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. REGCN implementation. *https://github.com/Lee-zix/RE-GCN*.

[50] Zixuan Li, Xiaolong Jin, Wei Li, Saiping Guan, Jiafeng Guo, Huawei Shen, Yuanzhuo Wang, and Xueqi Cheng. 2021. Temporal Knowledge Graph Reasoning Based on Evolutional Representation Learning. In *SIGIR*.

[51] Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, William L. Hamilton, David Duvenaud, Raquel Urtasun, and Richard S. Zemel. 2019. Efficient Graph Generation with Graph Recurrent Attention Networks. In *NeurIPS*.

[52] Xi Victoria Lin, Richard Socher, and Caiming Xiong. 2018. Multi-Hop Knowledge Graph Reasoning with Reward Shaping. In *EMNLP*.

[53] Yankai Lin, Zhiyuan Liu, Huan-Bo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *EMNLP*.

[54] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. 2018. Constrained Graph Variational Autoencoders for Molecule Design. In *NeurIPS*.

[55] Yung-Yin Lo, Wanjiun Liao, Cheng-Shang Chang, and Ying-Chin Lee. 2018. Temporal Matrix Factorization for Tracking Concept Drift in Individual User Preferences. *IEEE Trans. Comput. Soc. Syst.* 5, 1 (2018), 156–168.

[56] Yao Ma, Ziyi Guo, Zhaochun Ren, Jiliang Tang, and Dawei Yin. 2020. Streaming Graph Neural Networks. In *SIGIR*.

[57] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. 2015. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *CIDR*.

[58] Franco Manessi, Alessandro Rozza, and Mario Manzo. 2020. Dynamic graph convolutional networks. *Pattern Recognit.* 97 (2020).

[59] Pawel Matuszyk, João Vinagre, Myra Spiliopoulou, Alípio Mário Jorge, and João Gama. 2015. Forgetting methods for incremental matrix factorization in recommender systems. In *SAC*.

[60] Julian J. McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *WWW*.

[61] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR*.

[62] Christian Meilicke, Melisachew Wudage Chekol, Manuel Fink, and Heiner Stuckenschmidt. 2020. Reinforced Anytime Bottom Up Rule Learning for Knowledge Graph Completion. *CoRR* abs/2004.04412 (2020).

[63] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and Optimizing LSTM Language Models. In *ICLR*.

[64] Bryan David Minor and Diane J. Cook. 2017. Forecasting occurrences of activities. *Pervasive Mob. Comput.* 38 (2017), 77–91.

[65] Mohammad Hossein Namaki, Yinghui Wu, Qi Song, Peng Lin, and Tingjian Ge. 2017. Discovering Graph Temporal Association Rules. In *CIKM*.

[66] Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude Shavlik. 2010. Boosting relational dependency networks. In *ILP*.

[67] Lukás Neumann, Andrew Zisserman, and Andrea Vedaldi. 2019. Future Event Prediction: If and When. In *CVPR Workshops*.

[68] Jennifer Neville and David D. Jensen. 2007. Relational Dependency Networks. *J. Mach. Learn. Res.* 8 (2007), 653–692.

[69] Giang Hoang Nguyen, John Boaz Lee, Ryan A. Rossi, Nesreen K. Ahmed, Eunyee Koh, and Sungchul Kim. 2018. Continuous-Time Dynamic Network Embeddings. In *WWW*.

[70] Christos H. Papadimitriou. 1994. *Computational complexity*. Addison-Wesley.

[71] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao B. Schardl, and Charles E. Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *AAAI*.

[72] Erdős Paul and Rényi Alfréd. 1959. On random graphs I. *Publicationes Mathematicae (Debrecen)* 6 (1959), 290–297.

[73] Zhi Qiao, Shiwan Zhao, Cao Xiao, Xiang Li, Yong Qin, and Fei Wang. 2018. Pairwise-Ranking based Collaborative Recurrent Neural Networks for Clinical Event Prediction. In *IJCAI*.

[74] Meng Qu, Junkun Chen, Louis-Pascal A. C. Xhonneux, Yoshua Bengio, and Jian Tang. 2021. RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs. In *ICLR*.

[75] Idris Rabiu, Naomie Salim, Aminu Da'u, and Akram Osman. 2020. Recommender System Based on Temporal Models: A Systematic Review. *Applied Sciences* 10, 7 (2020), 2204.

[76] Dimitrios Rafailidis and Alexandros Nanopoulos. 2016. Modeling Users Preference Dynamics and Side Information in Recommender Systems. *IEEE Trans. Syst. Man Cybern. Syst.* 46, 6 (2016), 782–792.

[77] Fatemeh Rahimian, Amir H. Payberah, Sarunas Girdzijauskas, and Seif Haridi. 2014. Distributed Vertex-Cut Partitioning. In *DAIS*.

[78] Mahmudur Rahman and Mohammad Al Hasan. 2016. Link Prediction in Dynamic Networks Using Graphlet. In *ECML/PKDD*.

[79] Matthew Richardson and Pedro M. Domingos. 2006. Markov logic networks. *Mach. Learn.* 62, 1-2 (2006), 107–136.

[80] Tim Rocktäschel and Sebastian Riedel. 2017. End-to-end Differentiable Proving. In *NIPS*.

[81] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael M. Bronstein. 2020. Temporal Graph Networks for Deep Learning on Dynamic Graphs. *CoRR* abs/2006.10637 (2020).

[82] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. DRUM: End-To-End Differentiable Rule Mining On Knowledge Graphs. In *NeurIPS*.

[83] Erik Scharwächter, Emmanuel Müller, Jonathan Donges, Marwan Hassani, and Thomas Seidl. 2016. Detecting change processes in dynamic networks by frequent graph evolution rule mining. In *ICDM*.

[84] C. Seshadhri, Tamara G. Kolda, and Ali Pinar. 2011. Community structure and scale-free collections of Erdös-Rényi graphs. *CoRR* abs/1112.3644 (2011).

[85] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. ConvTransE implementation. *https://github.com/JD-AI-Research-Silicon-Valley/SACN*.

[86] Chao Shang, Yun Tang, Jing Huang, Jinbo Bi, Xiaodong He, and Bowen Zhou. 2019. End-to-End Structure-Aware Convolutional Networks for Knowledge Base Completion. In *AAAI*.

[87] Yelong Shen, Jianshu Chen, Po-Sen Huang, Yuqing Guo, and Jianfeng Gao. 2018. M-Walk: Learning to Walk over Graphs using Monte Carlo Tree Search. In *NeurIPS*.

[88] Tiago Sousa, João Correia, Vítor Pereira, and Miguel Rocha. 2021. Generative deep learning for targeted compound design. *Journal of Chemical Information and Modeling* 61, 11 (2021), 5343–5361.

[89] BaoShan Sun and Lingyu Dong. 2017. Dynamic Model Adaptive to User Interest Drift Based on Cluster and Nearest Neighbors. *IEEE Access* 5 (2017), 1682–1691.

[90] Jiaxi Tang and Ke Wang. 2018. Caser implementation. *https://github.com/graytowne/caser_pytorch*.

[91] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *WSDM*.

[92] Rakshit Trivedi, Hanjun Dai, Yichen Wang, and Le Song. 2017. Know-Evolve: Deep Temporal Reasoning for Dynamic Knowledge Graphs. In *ICML*.

[93] Rakshit Trivedi, Mehrdad Farajtabar, Prasenjeet Biswal, and Hongyuan Zha. 2019. DyRep: Learning Representations over Dynamic Graphs. In *ICLR*.

[94] Amin Vahedian, Xun Zhou, Ling Tong, W. Nick Street, and Yanhua Li. 2019. Predicting Urban Dispersal Events: A Two-Stage Framework through Deep Survival Analysis on Mobility Data. In *AAAI*.

[95] Ricardo Vilalta and Sheng Ma. 2002. Predicting Rare Events In Temporal Domains. In *ICDM*.

[96] João Vinagre and Alípio Mário Jorge. 2012. Forgetting mechanisms for scalable collaborative filtering. *J. Braz. Comput. Soc.* 18, 4 (2012), 271–282.

[97] Keqiang Wang, Yuanyuan Jin, Haofen Wang, Hongwei Peng, and Xiaoling Wang. 2018. Personalized Time-Aware Tag Recommendation. In *AAAI*.

[98] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world' networks. *Nature* 393, 6684 (1998), 440–442.

[99] Jeremy C. Weiss and David Page. 2013. Forest-Based Point Process for Event Prediction from Electronic Health Records. In *ECML/PKDD*.

[100] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *EMNLP*.

[101] Chengjin Xu, Mojtaba Nayyeri, Fouad Alkhoury, Jens Lehmann, and Hamed Shariat Yazdi. 2019. Temporal Knowledge Graph Embedding Model based on Additive Time Series Decomposition. *CoRR* abs/1911.07893 (2019).

[102] Qiang Yang, Hui Wang, and Wei Zhang. 2002. Web-log Mining for Quantitative Temporal-Event Prediction. *IEEE Intell. Informatics Bull.* 1, 1 (2002), 10–18.

[103] Yuan Yang and Le Song. 2020. Learn to Explain Efficiently via Neural Logic Inductive Learning. In *ICLR*.

[104] Jiaxuan You, Rex Ying, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. GraphRNN: Generating Realistic Graphs with Deep Auto-regressive Models. In *ICML*.

[105] Wenchao Yu, Wei Cheng, Charu C. Aggarwal, Haifeng Chen, and Wei Wang. 2017. Link Prediction with Spatial and Temporal Consistency in Dynamic Networks. In *IJCAI*.

[106] Giselle Zeno, Timothy La Fond, and Jennifer Neville. 2021. DYMOND: DYnamic MOtif-NoDes Network Generative Model. In *WWW*.

[107] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* 52, 1 (2019), 5:1–5:38.

[108] Wenbin Zhang, Liming Zhang, Dieter Pfoser, and Liang Zhao. 2021. Disentangled Dynamic Graph Deep Generation. In *SDM*.

[109] Liang Zhao. 2021. Event Prediction in the Big Data Era: A Systematic Survey. *ACM Comput. Surv.* 54, 5 (2021), 94:1–94:37.

[110] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A Data-Driven Graph Generative Model for Temporal Interaction Networks. In *KDD*.

[111] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. TagGen implementation. *https://github.com/davidchouzdw/TagGen*.