Interactive Temporal Association Analytics*

Xiao Qin, Ramoza Ahsan, Xika Lin, Elke A. Rundensteiner, and Matthew O. Ward

Department of Computer Science
Worcester Polytechnic Institute
100 Institute Road, Worcester MA, USA

{xqin,rahsan,xika,rundenst,matt}@cs.wpi.edu

ABSTRACT

Traditional temporal association mining systems, once supplied with a specific parameter setting such as time periods of interest, minimum support and confidence, generate the rule set from scratch. This one-at-a-time paradigm forces the analysts to perform successive trial-and-error iterations to finally discover interesting temporal patterns. This process is not only prohibitively time and resource consuming, but also ineffective in providing meaningful feedback for improving the desired rule outcome.

In this work, we introduce the first solution to interactively explore temporal associations from evolving data at multiple levels of abstraction, henceforth referred to as temporal association rule analytics (**TARA**). The offline rule preparation phase of the **TARA** infrastructure extracts the temporal associations from the raw data and compresses them into a knowledge-rich yet compact evolving parameter space (*EPS*) structure. The online exploration phase of **TARA** leverages this *EPS* structure to offer rich classes of novel exploration operations from parameter recommendations and time-travel queries to the discovery of hidden insights of associations with near instantaneous responsiveness. As demonstrated by our extensive experiments on real-world data sets, **TARA** accomplishes three to five orders of magnitude improvement over state-of-the art approaches while offering a rich interactive exploration experience.

1. INTRODUCTION

1.1 Motivation

Nowadays batches of data are continuously transmitted from a rich variety of sources including websites, mobile devices and other data sources, henceforth referred to as *evolving datasets*. Discovering associations and their dynamics hidden in such large evolving datasets has been recognized as critical for domains ranging from market products analysis, stock trend monitoring, targeted advertising to weather forecasting.

For example, in the retail businesses, the arrival of new fashions or gadgets may boost unprecedented sales while seasonal products may gain or lose customers' interest. Some products are purchased together more frequently in the days leading to a large sports event or during a traditional holiday like Thanksgiving. Companies such as Amazon, eBay, Walmart and other retail businesses apply temporal association mining techniques to their transaction logs to identify popular product combination at specific times and their behavior over time. Such information is critical for deciding the times when products can be placed together on a web page or configured into attractive bundle-offers to be used for recommendations to encourage sales.

Interactive data mining models, crucial for discovering knowledge from data, enable analysts to actively engage in the analysis process. State-of-the-art temporal association mining systems [2, 9, 14, 18], once supplied with a specific parameter setting, tend to generate the ruleset for each request from scratch. This one-at-the-time request model suffers from severe limitations described below.

1.2 Limitations of State-of-the-Art

Lack of instantaneous responsiveness. Lag in responsiveness is known to risk losing an analyst's attention during the exploration process. In applications like targeted ad placement such delay in decision making may prove to be the cause of missed business opportunities and thus a potentially huge loss in profit. Unfortunately, *temporal association mining algorithms* [2, 14] are known to be computationally intensive. To overcome this challenge, [18] pregenerates the *intermediate itemsets* that are subsequently used to derive the temporal associations instead of extracting them from the huge raw data store. With this promising one-time preprocessing strategy, the response time has been shown to be greatly reduced. However, the process of the final rule derivation remains a query-time task. This results in the shortcoming that the response times for mining such requests are not sufficient to support truly interactive exploration as confirmed by our experiments (Sec. 8).

Lack of parameter recommendations. Temporal association mining algorithms are parametrized not only by traditional measures like *support* and *confidence* but also by *time-variant* measures [11, 16, 17]. Parameter settings

*This work was partly supported by the National Science Foundation under grants IIS-1117139, CRI-1305258, IIS-0812027 and CCF-0811510 and Fulbright program.

used for one batch of data may produce insignificant rules for a newly incoming data batch. Thus the data analysts often must perform numerous successive trial-and-error iterations to find an appropriate parameter configuration out of a seemingly infinite number of possible settings. Existing state-of-the-art models tend to correspond to a black-box [2, 6, 9, 14, 18] - providing little to no feedback about which parameter settings best capture the analyst's interest. To tackle this, [10] incorporates an indexing technique to swiftly produce parameter recommendations. However it is restricted to static data and thus does not support *time variant operations* essential for temporal association mining.

Lack of evolving ruleset comparison. Analysis of the data in finer time granularity may reveal that associations exist only in certain time periods. Some may fluctuate as new data arrives while others may remain stable. Furthermore, two seemingly similar parameter settings can generate different results. Systems like [2, 6, 14, 18] independently generate the ruleset for each parameter settings. Worse yet, analysts then have to go through a tedious process to manually investigate the results generated by different parameter settings to extract their differences. This can be extremely tedious and impractical for large data sets.

Lack of insights into the evolving associations. Given a parameter configuration, a system often generates a huge number of rules. Analysts would benefit from being able to quickly identify the most interesting ones, such as the most stable rules [11] within the last week, the most significant rules that occur every weekend, or the rules concerning specific products. Offering such rich insights into time-variant rule behavior would provide the analysts with the opportunity to leverage their domain knowledge to drive the discovery process. Unfortunately, most existing parameter-driven exploration systems [10, 16, 18] do not support the analyst in the discovery of such useful *time-sensitive* insights.

1.3 Research Challenges

To develop an interactive temporal analytic system, the following research challenges must be tackled.

Processing time-variant evolving data. Given a time-variant data set containing n unique items, the maximum number of possible associations are bounded by $3^n - 2^n + 1$ [10]. The significance of associations may vary over time, as newly incoming data may bring new items and associations. Being able to quickly extract these associations and their behavior w.r.t different time horizons to answer analysts' requests is the key to providing an interactive mining experience. However, it is almost impossible to pre-generate all such information. Thus the system must have an efficient preprocessing strategy that pregenerates a minimal yet sufficient amount of information as its critical knowledge store to support interactive temporal association exploration.

Managing temporal associations for all parameters. Typical input parameters, such as *minimum support* and *confidence*, can be configured using any real number restricted to a certain range. Similarly, the time specification can be composed of one or multiple time periods along the continuous timeline. Clearly, an infinite number of possible parameter settings exists. Maintaining the corresponding ruleset for each parameter setting individually thus is impractical. Therefore, an efficient mechanism is needed to map the pregenerated temporal associations to the space of parameter settings.

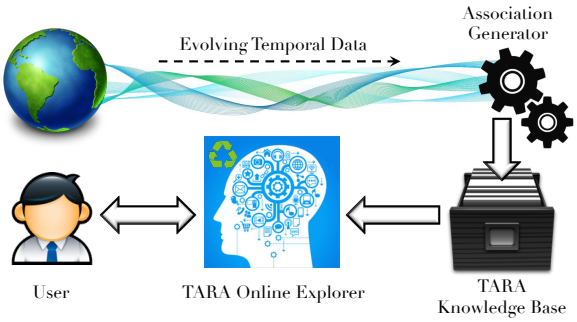


Figure 1: The TARA Approach

Maintaining parameter values for different time periods. The prominence level of an association may vary significantly for some associations while remaining stable for others. Such time-variant properties of parameter values may reveal important evolving patterns of an association in the evolving dataset. Yet keeping each single historical parameter value for each association is inefficient, resulting in large storage and search space. Therefore, a compact archive structure is needed to efficiently maintain the parameter values of the associations across time while supporting fast system access to retrieve any desired information.

Supporting advanced temporal association exploration. Rule mining algorithms tend to generate too many rules - making it extremely hard for the analysts to quickly identify the interesting ones. The problem of interestingness of temporal rules has been previously investigated [12, 17]. An interactive temporal association exploration system must integrate such interestingness measures to provide critical insights about the associations such as their evolving behaviors across time. The retrieved rules w.r.t particular parameter settings must be efficiently evaluated using these measures so that the instant responsiveness of the system is safeguarded.

1.4 The TARA Approach

We propose a novel temporal association rule analytics (**TARA**) framework that addresses the above challenges. The **TARA** infrastructure depicted in Fig. 1 employs an offline preprocessing phase composed of Association Generator and Knowledge Base Constructor followed by TARA Online Explorer that enables analysts to interactively explore the evolving data with support by the knowledge base.

The Association Generator extracts temporal associations from the evolving data and compactly stores them in the Temporal Association Rule Archive (*TAR Archive*) of TARA knowledge base. Later, by request, the parameter values of a particular association w.r.t various fine granularities can be quickly computed without processing the raw data again. These pregenerated temporal associations are compressed into a knowledge-rich yet compact *evolving parameter space* (*EPS*) that encodes the relationships among the temporal associations. Next, the TARA knowledge base explicitly extracts and then models the distribution of the pregenerated temporal associations with respect to their parameters, e.g. support, confidence and time periods.

Beyond achieving speedup in response time, the online processing strategies leverage the *EPS* index to offer analysts an innovative "rule-centric panorama" into the temporal associations present within the evolving dataset. The framework supports rich classes of novel exploration opera-

tions from time-travel queries and parameter recommendations to evolving ruleset comparisons.

1.5 Contributions

Key contributions of this work include:

• We propose the first *interactive temporal association rule mining* analytics framework called **TARA** that enables analysts to explore associations across time and pinpoint appropriate parameter settings in a systematic way.

• The **TARA** model organizes the temporal associations in the space of query parameters. It abstracts the temporal associations at the coarse granularity of *time-aware stable regions* across multiple time periods.

• The **TARA** model is supported by *evolving parameter space* (*EPS*) index structure that indexes *time-aware stable regions* along with the associated domination graph. TARA offers efficient algorithms for offline *EPS* index construction.

• For the rules generated, we design a temporal association rule archive, called *TAR Archive*, that compactly encodes the parameter values of each rule across time. Our specially designed encoding and decoding strategies achieve fast access to the requested information from this archive.

• We propose a rich set of novel temporal rule exploration operations beyond traditional temporal rule mining. Effective strategies for the online processing of the proposed operations that leverage our precomputed TARA index structures are provided.

• TARA framework supports the exploration of the associations at coarser or finer time granularities by roll-up and drill down operations. We provide a theoretical bound on the approximation of the solution under roll-up operations.

• Our extensive experiments using IBM Quest [1], *retail* [3] and *webdocs* [13] datasets demonstrate that **TARA** is 3 to 5 orders of magnitude faster than its state-of-the-art competitors for traditional temporal association mining, while in addition supporting novel analytics within milliseconds.

2. PRELIMINARIES OF TEMPORAL ASSOCIATION

$\mathcal{T} = \{..., t_i, ..., t_j, ...\}$ denotes a set of **times**, countably infinite, over which a linear order $<_\mathcal{T}$ is defined, where $t_i <_\mathcal{T} t_j$ means t_i occurs strictly before t_j. Let $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ represent a set of **items**. $\mathcal{D} = \{d_1, d_2, ..., d_m\}$ is a collection of subsets of I called the **transaction database**. Each **transaction** d_i in $\mathcal{D}$ is a set of items such that $d_i \subseteq \mathcal{I}$. Each d_i has an associated timestamp t_j, denoted by $d_i.time = t_j$. Let $\mathcal{X} \subseteq \mathcal{I}$ be a set of items, called **itemset**. If $\mathcal{X} \subseteq d_i$, d_i *contains* $\mathcal{X}$. If the cardinality of $\mathcal{X}$ is k, $\mathcal{X}$ is called a **k-itemset**. Given a closed **time period** $[t_i, t_j]$ where $t_i <_\mathcal{T} t_j$, then the set of transactions in the range $[t_i, t_j]$ of $\mathcal{D}$ that *contain* $\mathcal{X}$ is indicated by $\mathcal{F}(\mathcal{X}, \mathcal{D}, [t_i, t_j]) = \{d_k \mid d_k \in \mathcal{D} \wedge t_i \leq d_k.time \leq t_j \wedge \mathcal{X} \subseteq d_k\}$.

Definition 1. *A **temporal association rule** is an expression of the form $\mathcal{R}^{[t_i,t_j]} \equiv (\mathcal{X} \Rightarrow \mathcal{Y})$, where $\mathcal{X} \subseteq \mathcal{I}$, $\mathcal{Y} \subseteq \mathcal{I} \setminus \mathcal{X}$, and $[t_i, t_j]$ indicates that $\mathcal{R}$ is derived from all the transactions in $\mathcal{D}$ whose timestamps fall into $[t_i, t_j]$.*

A **temporal association rule** defaults to the traditional association rule if the *time period* is set to the entire timeline. This time restriction $[t_i, t_j]$ empowers the data analysts to discover associations that are not significant throughout the entire data set. Moreover, an association may reappear in multiple time periods expressing some periodicity. Furthermore, the association may behave differently in terms of its measured values. The evolution of the associations over time can lead to insightful observations [11].

Table 1: Example of pregenerated temporal association rule

(a) Item set (min supp = 0.05)

Itemset	Support	
	$\mathcal{T}_1$	$\mathcal{T}_2$
a	0.36	0.44
b	0.45	0.22
c	0.36	0.44
ab	0.18	0.11
ac	0.18	0.33
bc	0.09	0.11

(b) Rule (min conf = 0.25)

Rule	(Support, Confidence)	
	$\mathcal{T}_1$	$\mathcal{T}_2$
$\mathcal{R}_1$: a->b	(0.18, 0.5)	(0.11, 0.25)
$\mathcal{R}_2$: b->a	(0.18, 0.4)	(0.11, 0.5)
$\mathcal{R}_3$: a->c	(0.18, 0.5)	(0.33, 0.75)
$\mathcal{R}_4$: c->a	(0.18, 0.5)	(0.33, 0.75)
$\mathcal{R}_5$: c->b	(0.09, 0.25)	(0.11, 0.25)
$\mathcal{R}_6$: b->c	-	(0.11, 0.5)

Many measurements [17] have been proposed to evaluate the interestingness of associations. Out of these measurements, we work with the most common measures of *support* and *confidence* to demonstrate the key principles of our framework, though others can be plugged in the future.

$$Support(\mathcal{R}^{[t_i,t_j]}) = \frac{|\mathcal{F}(\mathcal{X} \cup \mathcal{Y}, \mathcal{D}, [t_i, t_j])|}{|\mathcal{F}(\emptyset, \mathcal{D}, [t_i, t_j])|} \quad (1)$$

$$Confidence(\mathcal{R}^{[t_i,t_j]}) = \frac{|\mathcal{F}(\mathcal{X} \cup \mathcal{Y}, \mathcal{D}, [t_i, t_j])|}{|\mathcal{F}(\mathcal{X}, \mathcal{D}, [t_i, t_j])|} \quad (2)$$

The **support** defined in Formula 1 describes the proportion of the transactions within the defined time period that *contains* all items in the association. The **confidence** defined in Formula 2 describes the probability of finding the **consequent** $\mathcal{Y}$ of the association within the defined time period under the condition that these transactions also *contain* the **antecedent** $\mathcal{X}$.

3. THE TARA MODEL

We now introduce our **TARA** model framework for interactive exploration of associations from evolving data.

3.1 Time Dimension of the TARA Model

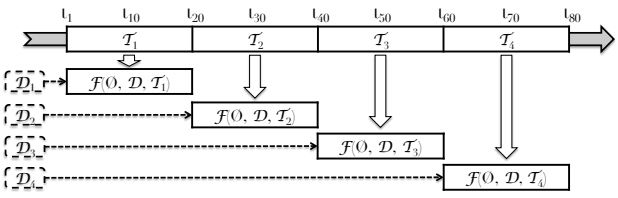


Figure 2: Tumbling Window Model of TARA

Data analysts often are interested in exploring the associations that hold in particular time periods, such as an hour or a day. Coarser time specifications can be broken down to ranges of smaller granularities. Moreover, the measures of an association in a longer time period can then be computed based on the measures of the associations in the shorter periods that are part of this longer period. Based on this observation, **TARA** partitions the data set into disjoint time periods, called *windows*. Mining queries with a coarser time granularity settings than this basic *window size* are then supported using roll-up operations.

Let $\mathcal{D}$ be the evolving data set and w be the basic window

size that represents the minimum granularity. Therefore, the set of *times* $\mathcal{T}$ contains disjoint but consecutive time periods each of size w denoted by $\mathcal{T} = \{..., \mathcal{T}_i, ..., \mathcal{T}_j, ...\}$, $(\forall \mathcal{T}_i, \mathcal{T}_j)$, if $\mathcal{T}_i \neq \mathcal{T}_j$, and $\mathcal{T}_i \cap \mathcal{T}_j = \emptyset$. The evolving data set $\mathcal{D}$ is partitioned into small chunks according to each time period $\mathcal{T}_i$ in $\mathcal{T}$ denoted by $\mathcal{D} = \{..., \mathcal{D}_i, ..., \mathcal{D}_j, ...\}$ where $\mathcal{D}_i = \mathcal{F}(\emptyset, \mathcal{D}, \mathcal{T}_i)$. In Fig. 2, for example we set the *window size* $w = 20$. That is, the time frame is partitioned into a set of time periods of length 20, e.g $\mathcal{T}_2 = [t_{21}, t_{40}]$. The evolving data set $\mathcal{D}$ is partitioned into time-oriented data subsets $\mathcal{D}_i$ according to these time periods, e.g. $\mathcal{D}_2 = \mathcal{F}(\emptyset, \mathcal{D}, \mathcal{T}_2)$. For each data partition $\mathcal{D}_i$, **TARA** pregenerates the associations off-line (See Sec. 4). Table 1.(b) shows an example of the associations generated for the time periods $\mathcal{T}_1$ and $\mathcal{T}_2$. **TARA** processes the raw data $\mathcal{D}$ once to pregenerate the temporal associations held in these windows. A query with the coarser time specification can then be answered based on these pregenerated associations.

DEFINITION 2. ***Time availability**: Let w be the finest time granularity. Then $\mathcal{T}^w = \{..., \mathcal{T}_i^w, ..., \mathcal{T}_j^w, ...\}$ corresponds to the basic time periods of $\mathcal{T}$ that are generated by **TARA** through time partitioning by w. A time specification $\mathcal{T}_k$ supported in **TARA** thus is $\mathcal{T}_k = \bigcup_{m=i}^{j} \mathcal{T}_m^w$, where $i \leq j$.*

This strategy allows us to support **roll-up** and **drill-down** of time periods at run time such as days, months or years to support long and short term goals.

3.2 Evolving Parameter Space Model

In association rule mining, the input parameter values of *minimum support* and *confidence* can be any real number within [0,1]. Each combination, referred to as **parameter setting**, corresponds to a set of rules generated by using this parameter setting. We now extend this into the notion of an *Evolving Parameter Space* (*EPS*) that models relationships and distribution of rules across the multi-dimensional temporal parameter space.

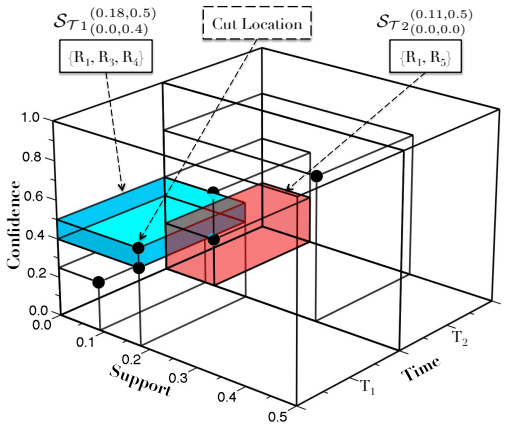


Figure 3: Evolving Parameter Space

DEFINITION 3. ***Evolving Parameter Space**: Let $\mathcal{D}$ be an evolving data set, $\mathcal{D}_i$ be a **partition** of $\mathcal{D}$ by a basic time granularity $\mathcal{T}_i$, $\forall \mathcal{T}_i \in \mathcal{T}$. Let p_j be one of the n parameters. The (n+1)-dimensional space, denoted by $\mathcal{E} = \{ p_1,...,p_n, \mathcal{T}\}$ and called **Evolving Parameter Space (EPS)**, organizes the rules $\{\mathcal{R}\}^{\mathcal{T}}$ where $\{\mathcal{R}\}^{\mathcal{T}} = \bigcup_{i=1}^{k} \{\mathcal{R}\}^{\mathcal{T}_i}$ and k is the total number of time partitions of $\mathcal{D}$. A temporal association rule $\mathcal{R}$ is associated with its **temporal parametric locations** $(\mathcal{R}.value(p_1),..., \mathcal{R}.value(p_n))^{\mathcal{T}_i}$ where $\mathcal{R}.value(p_j)$ denotes the value of the j^{th} parameter for rule $\mathcal{R}$ in time $\mathcal{T}_i$.*

For simplicity, we use two parameters, namely *support* and *confidence* while others could be defined as well. Thus henceforth, the *EPS* $\mathcal{E}$ is a 3-dimensional space with *support*, *confidence* and *time* as its dimensions. A *temporal parametric location* depicting a rule $\mathcal{R}$ in time $\mathcal{T}_i$, denoted as $\mathcal{R}(supp, conf)^{\mathcal{T}_i}$, is represented as a line segment indicating the parameter values of $\mathcal{R}$ in $\mathcal{T}_i$. Fig. 3 shows the *EPS* for the rules in Table 1(b). Rules $\mathcal{R}_1$, $\mathcal{R}_3$ and $\mathcal{R}_4$ map to the same *temporal parametric location* $(0.18, 0.5)^{\mathcal{T}_1}$ in the time period $\mathcal{T}_1$. However in time $\mathcal{T}_2$, $\mathcal{R}_1$ travels in the space so that now it maps to same location as $\mathcal{R}_5(0.11, 0.5)^{\mathcal{T}_2}$.

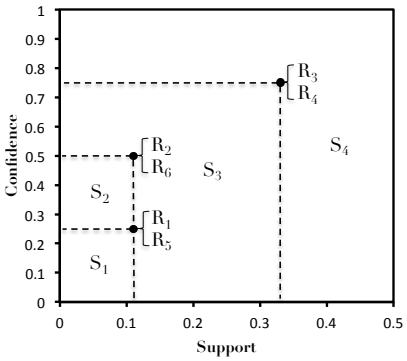


Figure 4: Evolving Parameter Space slice at Time $\mathcal{T}_2$

LEMMA 3.1. *Let $\mathcal{L}$ denote a temporal parametric location in the EPS $\mathcal{E}$, $\mathcal{L}.p_i$ be the value of parameter p_i for location $\mathcal{L}$. Given a set of temporal parametric locations in the same time period $\mathcal{T}_i$, $\forall \mathcal{L}_m, \mathcal{L}_n \in \{\mathcal{L}\}$, where $m \neq n$, if there exists a p_i such that $\mathcal{L}_m.p_i \neq \mathcal{L}_n.p_i$, then the temporal association rules that map to $\mathcal{L}_m$ are guaranteed to be distinct from those that map to $\mathcal{L}_n$.*

PROOF. Rules' *temporal parametric locations* in time $\mathcal{T}_i$ are generated from the same data partition $\mathcal{D}_i$. Any given rule at time $\mathcal{T}_i$ cannot have two distinct values for one parameter. Therefore, a rule $\mathcal{R}$ cannot map to two distinct *temporal parametric locations* within the same time. □

Each rule's *temporal parametric location* can either remain steady or change over multiple time periods. We call this stream of locations the **trajectory of the association**.

DEFINITION 4. ***Trajectory of an association**: Given a sequence of time periods $\{\mathcal{T}\} = \{\mathcal{T}_1..., \mathcal{T}_m\}$, the **trajectory of an association** $\mathcal{R}$ in $\{\mathcal{T}\}$ is the set of temporal parametric locations that represent its parameter values in the time periods in $\{\mathcal{T}\}$.*

This trajectory of a rule allows us to compute different measures about the rule that summarize its evolving patterns like *coverage* [16], *stability* [11] and *standard deviation*. These measures can be computed for each individual rule or even for a set of rules to provide individual or global summarization respectively.

Given a data set with n unique items, the maximum number of rules is finite, bounded by $3^n - 2^n + 1$ [10]. Therefore, some set of parameter settings must correspond to same set

of rules. Fig. 4 shows a slice of the evolving parameter space for time $\mathcal{T}_2$. Rules with identical parameter values are represented by the same point in this space. These points partition the space into 4 regions marked by dashed lines. If a user specified *minimum support* and *confidence* configuration for mining falls into region $\mathcal{S}_3$, then regardless of its actual position within the region, the output ruleset is always $\{\mathcal{R}_3, \mathcal{R}_4\}$. This observation is inspired by the work presented in [10]. Thus the entire evolving parameter space at a time $\mathcal{T}_i$ can be partitioned into a finite set of regions referred to as *time-aware stable regions*. This notion of *time-aware stable regions* forms our coarse granularity abstraction of the temporal association rules generated from an evolving data set $\mathcal{D}$.

DEFINITION 5. ***Time-Aware Stable Regions**: Given an EPS $\mathcal{E}$ of n parameters $\{p_1,...,p_n\}$ and times $\mathcal{T}$ as $(n+1)$ dimensions for an evolving data set $\mathcal{D}$, then a **time-aware stable region** in a time period $\mathcal{T}_i$ is a closed hyper-box denoted by ${\mathcal{S}_{\mathcal{T}_i}}^{(upper(p_1),...upper(p_n))}_{\{(lower(p_1),...lower(p_n))\}}$ with its boundary specified by locations $(\mathcal{S}.upper(p_1),...,\mathcal{S}.upper(p_n))^{\mathcal{T}_i}$ and $\{(\mathcal{S}.lower(p_1),...,\mathcal{S}.lower(p_n))^{\mathcal{T}_i}\}$ within each of which no matter how the parameter values are adjusted, the set of rules generated from $\mathcal{D}_i$ remains unchanged.*

Considering the 3-dimensional *EPS* in Fig. 3, a *time-aware stable region* is bounded by an upper location $(supp_u, conf_u)^{\mathcal{T}_i}$ and k lower locations $\{(supp_{l_j}, conf_{l_j})^{\mathcal{T}_i}\}$ where $j \in [1,k]$. The *support* and *confidence* values of the upper location will always be greater than those of all its lower points, i.e., $\forall j$ $(supp_u \geq supp_{l_j})$ and $(conf_u \geq conf_{l_j})$. The upper location of a *time-aware stable region* is called its **cut location**.

DEFINITION 6. ***Cut Location**: Let EPS $\mathcal{E}$ be a 3-dimensional space with support x, confidence y and time z as its dimensions, $\{\mathcal{X}\}$ be a set of the intersections formed by the perpendicular projections of each temporal parametric location onto x and y planes. The cut locations within $\mathcal{E}$ are then denoted by $\{\mathcal{C}\}$, where $\{\mathcal{X}\} = \{\mathcal{C}\} \cup \{\mathcal{L}\}$.*

Fig. 3 depicts *time-aware stable regions* ${\mathcal{S}_{\mathcal{T}_1}}^{(0.18,0.5)}_{(0,0.4)}$ and ${\mathcal{S}_{\mathcal{T}_2}}^{(0.11,0.5)}_{(0,0)}$. For region ${\mathcal{S}_{\mathcal{T}1}}^{(0.18,0.5)}_{(0,0.4)}$, the *cut location* is $(0.18, 0.5)^{\mathcal{T}_1}$. It is bounded by the parametric locations $(0.18,0.5)^{\mathcal{T}_1}$ and $(0,0.4)^{\mathcal{T}_1}$ and contains rules $\mathcal{R}_1$, $\mathcal{R}_3$ and $\mathcal{R}_4$.

LEMMA 3.2. *Given a set of time-aware stable regions $\{\mathcal{S}\}$ for the same $\mathcal{T}_i$, $\forall$ $\mathcal{S}_m$, $\mathcal{S}_n \in \{\mathcal{S}\}$, where $m \neq n$, the associations that map to the cut location of $\mathcal{S}_m$ are guaranteed to be distinct from the ones that map to the cut location of $\mathcal{S}_n$.*

PROOF. By Lemma 3.1, rules generated in the same time period but map to different *temporal parametric locations* are guaranteed to be distinct. The locations in $\{\mathcal{X}\}$ either belong to $\{\mathcal{L}\}$ or have no rule. Therefore, within a time period $\mathcal{T}_i$, rules that map to different *time-aware stable regions* are guaranteed to be distinct. □

DEFINITION 7. ***Dominating Stable Region**: A time-aware stable region $\mathcal{S}_m$ **dominates** region $\mathcal{S}_n$ where $m \neq n$, if and only if $\forall p_i \in \mathcal{P}$ $\mathcal{S}_m.\mathcal{C}.p_i \leq \mathcal{S}_n.\mathcal{C}.p_i$, and $\mathcal{S}_m$ and $\mathcal{S}_n$ are in same $\mathcal{T}_i$ where $\mathcal{S}_m.\mathcal{C}$ refers to the cut location of stable region $\mathcal{S}_m$.*

LEMMA 3.3. *Considering two time-aware stable regions $\mathcal{S}_m$ and $\mathcal{S}_n$ where $m \neq n$. If $\mathcal{S}_m$ dominates $\mathcal{S}_n$, then rules valid within the dominated region $\mathcal{S}_n$ are also valid in the dominating region $\mathcal{S}_m$ but not vice versa.*

PROOF. A temporal rule $\mathcal{R}_i$ is in the final output ruleset if in the specified $\mathcal{T}_k$, $\forall$ p_j, $\mathcal{R}_i.\text{value}(p_j) \geq min\ parameters$ where $p_j \in \{p_1,...,p_n\}$. If $\mathcal{R}_i$ belongs to region $\mathcal{S}_n$, the *temporal parametric location* of $\mathcal{R}_i$ is the upper location of $\mathcal{S}_n$. Because $\mathcal{S}_m$ dominates $\mathcal{S}_n$, $\forall$ p_j, $\mathcal{S}_m.upper(p_j) \leq \mathcal{S}_n.upper(p_j)$, meaning $\forall$ p_j, $\mathcal{S}_m.upper(p_j) \leq \mathcal{R}_i.\text{value}(p_j)$. So $\mathcal{R}_i$ is valid in $\mathcal{S}_m$ as well. However, vice versa is not true, as can be trivially shown. □

Consider $\mathcal{S}_1 = {\mathcal{S}_{\mathcal{T}_1}}^{(0.18,0.5)}_{(0,0.4)}$ and $\mathcal{S}_2 = {\mathcal{S}_{\mathcal{T}_1}}^{(0.09,0.25)}_{(0,0)}$ in Fig 3. Based on Def. 7, $\mathcal{S}_2$ dominates $\mathcal{S}_1$ because every parameter value in the upper location of $\mathcal{S}_2$ is smaller than the corresponding value of $\mathcal{S}_1$.

If the rules in region ${\mathcal{S}_{\mathcal{T}_1}}^{(0.09,0.25)}_{(0,0)}$ are included in the final result, then region must also contain the rules that are valid in ${\mathcal{S}_{\mathcal{T}_1}}^{(0.18,0.5)}_{(0,0.4)}$. By Lemma 3.3, given a user-specified parameter setting, once a region is identified as a valid region to produce the final ruleset, all its dominated regions should then also be included in the user output.

Using this concept of dominating stable regions [10], each rule is stored once in the stable region and by iterating over its dominating regions the final ruleset can simply be obtained for a given pair of *support* and *confidence* values.

3.3 Supported Queries on TARA Model

We now propose TARA operations that offer a rich classes of novel temporal analytical queries.

Temporal Association Mining. Given a parameter setting and time periods, $Q1$ returns the associations that satisfy the minimum parameters, such as *minimum support* and *confidence*. The evolving trajectory and measures of associations for each of the specified time periods are also returned. The Measures including *coverage* [16], *stability* [11] and *standard deviation* summarize the evolving patterns. *Exact match* option returns the associations that are consistently valid in *all* of the specified time periods. *Single match* returns the associations that are valid in exactly one of the specified time periods. *Fuzzy match* returns the associations that are valid in *at least* one or more of the specified time periods. $Q2$ returns the differences of associations with regards to two different parameter settings.

```
Q1 RETURN Rule, Trajectory, Measures
   FROM  Evolving Data Set D
   PARAMETER ⋀_{i=1}^{n} MinParameter_i = P_i
   IN-TIME = Time Periods & Granularity
   MATCH = Exact|Single(Time Period)|Fuzzy
```

Use Case for Temporal Association Mining. In the retail dataset [3], over the time period of one year broken in windows of a week, $Q1$ may return the rule (*turkey* $\Longrightarrow$ *pumpkin pie*) with relatively low support and confidence value (0.4,0.5) throughout the year. However, this rule periodically rises to high support and confidence value of (0.6,0.7) in the week before *Thankgiving*. If we had examined this rule in the roll-up view over the complete data (whole year), it's support and confidence would overall have been too low. With Q1, an analyst can find the rules that are frequent

only within certain intervals. It also allows to find all time periods during which this rule was more popular.

```
Q2 RETURN Rule, Trajectory, Measures
   FROM  EPS E
   PARAMETER ⋀_{i=1}^{n} MinParameter1_i = P1_i
   COMPARE-TO ⋀_{i=1}^{n} MinParameter2_i = P2_i
   IN-TIME = Time Periods & Granularity
   MATCH = Exact|Single(Time Period)|Fuzzy
```

Use Case of Unnoticeable Changes. For a dense data set like webdocs [13], the size of the returned rulesets is huge, making it difficult for the analysts to perform manual inspection of the differences produced by two different parameter settings. To effectively tune the best parameter setting that includes the most important associations across the specified time intervals, analyst would benefit from being able to quickly explore the differences in the results. $Q2$ allow to find all rules that were generated in last month by parameter configuration setting like (0.4, 0.8) but not by configuration setting (0.5, 0.95).

Stable Region Exploration. These queries provide meta information about the *time-aware stable regions*. In particular, $Q3$ returns the sets of stable regions identified for the given parameter settings across the time periods specified by the *match* clause. $Q4$ returns region parameters that contain the specified associations.

```
Q3 RETURN Stable Region
   FROM  EPS E
   PARAMETER ⋀_{i=1}^{n} MinParameter_i = P_i
   IN-TIME = Time Periods & Granularity
   MATCH = Exact|Single(Time Period)|Fuzzy
```

Use Case of Parameter Recommendation. For a sparse dataset like retail [3], often despite submitting several successive mining requests with different parameter settings, the system repeatedly returns the same set of rules due to a sparse distribution of rules. This trial and error process can be avoided by using query $Q3$. For example for the retail dataset [3] broken in windows of a week, $Q3$ can easily inform an analyst that during the last week of a month, the same set of rules will be generated for all parameter configurations within (0.88, 0.76) and (0.91,0.78).

```
Q4 RETURN Region
   FROM  Ruleset ⋃_{i=1}^{n} R_i
   IN-TIME = Time Periods & Granularity
   MATCH = Exact|Single(Time Period)|Fuzzy
```

Time Specification: Roll-up and Drill-down. For each of the above queries, a time granularity must be specified. For example, a time period $\mathcal{T}_i$ can refer to a particular *hour*, a *day*, or a *week*.

Rule Search. As [12] pointed out, supporting content-based rule search can leverage analysts' domain knowledge to efficiently narrow down the result into a more manageable smaller set. $Q5$ allows analysts to filter a ruleset, generated by any of the above queries, based on the absence or presence of certain *items* given in the MATCH clause.

```
Q5 RETURN Rule
   FROM Ruleset ⋃_{i=1}^{n} R_i
   MATCH {I}^+ ∪ {I}^-
```

Use Case of Content-based Association Exploration. In the retail dataset using query $Q1$ with *Single match* option and the time equal to week before *Christmas* returns a huge number of rules for the support and confidence values of (0.4,0.5). With $Q5$ an analyst can quickly find all the rules that contains *"Ipads"* and do not include *"Laptops"*.

4. OFF-LINE EPS-INDEX CONSTRUCTION

Our proposed *Evolving Parameter Space* (EPS) Index construction is composed of three tasks that are performed off-line. Task 1 generates the temporal association rules; task 2 computes the *time-aware stable regions* and constructs the domination graph, and task 3 constructs the *EPS-Index*. These tasks three are explained below (See Algo. 1).

Task 1: Temporal Association Rule Generation. **TARA** first pre-generates the rules from the evolving dataset using the lowest meaningful parameter settings, called the *primary support* θ and the *confidence* λ, to prevent excess pre-generation [12]. Based on the *minimum time granularity*, **TARA** mines the rules from the current window, whose *support* and $confidence$ value exceed θ and λ. The rules and their parameter values w.r.t to the transactions within this window are then archived (See Sec. 5). In our work, we plug in FP-Tree [7] as the rule generation algorithm. The rule mining module in **TARA** framework is extendable to any incremental or parallel rule mining solution [8].

Task 2: Time-Aware Stable Region. To construct the *time-aware stable regions* (see Def. 5), **TARA** first computes the *cut locations* using a two step approach. In the first step, the *temporal parametric locations* of rules generated in the previous task are initialized as the first set of *cut locations*. Each *cut location* maintains a set of references to its rules. In the second step, let x, y, and z be the axes representing *support*, $confidence$ and *time* measures and o be the origin of all measures. The intersections formed by the perpendicular projections of each point onto the x and y planes are added to the *cut location* set.

By Lemma 3.2, each *cut location* identifies a unique *time-aware stable region*. For each *cut location*, the lower bounds are computed to form a complete *time-aware stable region*. Simultaneously, the *immediate dominated regions* are identified and connected to construct the domination graph. Upon user request, the *time-aware stable regions* can be constructed for the coarser time granularity over basic granularity used by the system. New parameter values of rules forming these *time-aware stable regions* are computed by Formulas 4 & 5.

Task 3: EPS-Index Construction. Within each *time-aware stable region*, all rules are indexed by their respective attributes, called a *region shard*. Next, the two layered Evolving Parameter Space Index (*EPS-Index*) is created to efficiently answer the **TARA** model requests. The top level of the *EPS-Index* facilitates the search to locate a particular *time-aware stable region* given its input parameters. Regions are indexed separately by different time periods. For regions within a single time period, a grid-based spatial index is utilized to partition the EPS into equal-sized grid cells. The time-aware stable regions are then allocated to their respective positions in the grid. The stable regions in

Algorithm 1: Offline EPS Construction

Input: Dataset $\mathcal{D}$
begin
  for *each* $\mathcal{D}_i$ *from* $\mathcal{D}$ **do**
    $\{\mathcal{R}\} \longleftarrow \emptyset$; $\{\mathcal{S}\} \longleftarrow \emptyset$;
    $\{\mathcal{R}\} \longleftarrow$ RuleGenerator($\mathcal{D}_i$, θ, λ);
    $\{\mathcal{S}\} \longleftarrow$ RegionAbstractor($\{\mathcal{R}\}$);
    $\mathcal{E} \longleftarrow$ EPSIndexConstructor($\{\mathcal{S}^+\}$,$\{\mathcal{R}\}$);

1.A: RuleGenerator
Input: $\mathcal{D}_i$, θ, λ
Output: RuleSet $\{\mathcal{R}\}$
begin
  $\{\mathcal{R}\} \longleftarrow \emptyset$;
  $\{\mathcal{R}\} \longleftarrow$ FP-Tree($\mathcal{D}_i$, θ, λ);
  Archive($\{\mathcal{R}\}$);

1.B: RegionAbstractor
Input: $\{\mathcal{R}\}$
Output: RegionSet $\{\mathcal{S}\}$
begin
  $\{\mathcal{C}\} \longleftarrow \emptyset$; /*Initial cut location set*/
  $\{\mathcal{S}\} \longleftarrow \emptyset$;
  $\{\mathcal{C}\} \longleftarrow$ GetCutLocation($\{\mathcal{R}\}$);
  for *each* $\mathcal{C}_i \in \{\mathcal{C}\}$ **do**
    $\mathcal{S} \longleftarrow$ ConstructShard($\mathcal{C}_i$);
    $\mathcal{S} \longleftarrow$ GetRegion($\mathcal{C}_i$, $\{\mathcal{C}\}$);
    $\mathcal{S} \longleftarrow$ ConnectDominatedRegion($\mathcal{S}$,$\{\mathcal{C}\}$);
    $\{\mathcal{S}\} \longleftarrow \{\mathcal{S}\} \cup$ S;

1.C: EPSIndexConstructor
Input: $\{\mathcal{S}\}$
Output: EPS-Index $\mathcal{E}$
begin
  GridIndexer($\{\mathcal{S}\}$);

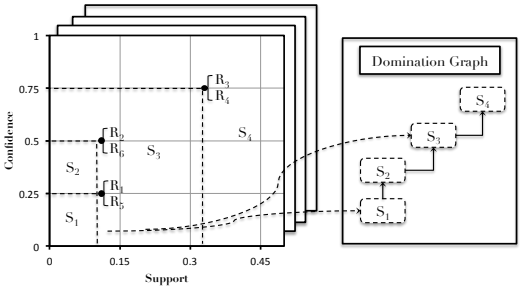


Figure 5: The EPS-Index

each cell point to the corresponding nodes in the next level of the *EPS-Index*. Fig. 5 shows the spatial index of the *time-aware stable regions* for $\mathcal{T}_1$.

Using the proposed grid structure, the online search for a stable region can be performed in near constant time. The online processing of our **TARA** exploration using this index is described in Sec. 7.

The second level of the *EPS-Index*, namely, the region domination graph (Fig. 5), is designed to expedite the collection of rules from dominating regions [10]. Each stable region forms a node in the graph with each node linked to its closest dominating neighbors. The region domination graph enables us to locate the closest dominating neighbor regions in near constant time and also produce complete rule sets in linear time in the number of rules involved.

5. TEMPORAL ASSOCIATION ARCHIVE

As explained before, the rules themselves can be huge. We describe an efficient storage structure for managing the rules generated across time called temporal association rule archive (*TAR Archive*).

5.1 TAR Archive Design

Our proposed archive structure consists of a directory and a number of index entries. The directory contains all temporal association rules. Each rule in the directory has a pointer to its index entry. The index entry stores the history of its parameters w.r.t a specific time granularity. The index entry can be implemented using a *time sequence* [6] as illustrated in Fig. 6. For simplicity only one parameter is shown. The size of the *time sequence* is equal to the number of basic windows. The advantage of using such naive *time sequence* is that every element in this structure maps to a parameter value of the rule within a specific window. Given a window and rule id, the search of the parameter value takes only $O(1)$ time.

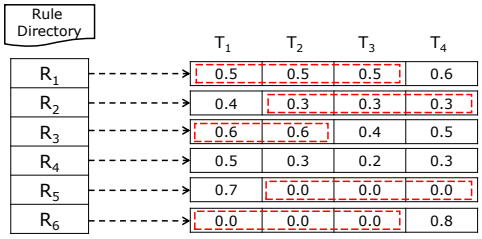


Figure 6: Rule Index with Time Sequence

Interestingly, [15] observed that daily updates in transaction databases more often than not affect only a small part of the ruleset. [6] also indicates that parameter values of the frequent patterns mined from the data stream often remain stable within a period of time. In Fig. 6, $\mathcal{R}_1$,$\mathcal{R}_2$ and $\mathcal{R}_3$ are stable over several consecutive windows. Therefore, the time sequence structure may contain redundant values.

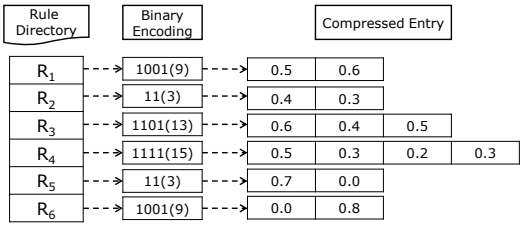


Figure 7: Rule Index with Compact Time Sequence

To avoid the above problems while still achieving efficient access, we propose the *TAR Archive* with *compact time sequence* structure as depicted in Fig. 7. Each *compact time sequence* for one parameter consists of a binary code $\mathcal{B}$ and an array of distinct parameter values, called *value sequence*, denoted as $\mathcal{V} = \{v_1, ...v_n\}$. The ith bit from the lowest order indicates whether or not the parameter value within the ith window is identical to the value within the previous window. The encoding strategy is shown in Formula 3.

$$Bit(\mathcal{R}.p_i^j) = \begin{cases} 0 & \text{if } \mathcal{R}.p_i^j = \mathcal{R}.p_i^{j-1} \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

For example, in Fig. 6 the parameter values of R_1 are stable in the first three windows with only the value in the latest window being different. "1001" denotes the evolution of this parameter across 4 windows. The lowest order digit represents the parameter value in the oldest window and the highest order digit represents the parameter value in the newest window. In this case, the parameter values in $\mathcal{T}_1$, $\mathcal{T}_2$

and $\mathcal{T}_3$ only need to be stored once in the structure because they are identical.

5.2 TAR Archive Operations

Next we discuss the supported operations, namely *append*, *access*, *purge* and *merge* on the *TAR Archive*.

Append: As the new window is being processed, rules are added into the *TAR Archive*. We distinguish between three cases when a rule is being appended: (1) The new rule already exists in the archive; (2) The new rule does not exist in the archive; (3) A rule that is in the archive does no longer appear in the newly generated ruleset.

For case 1, if the new value of the rule is different than in the last window, then "1" is added to the binary code otherwise no action is needed. To address case 2, first, this new rule is inserted in the rule directory. Second, for each parameter, if the current window is the very first window, the binary code is set to "1" and p_i^0 is inserted into the empty *value sequence*. If $j > 0$, the binary code is first initialized as "1" and "0.0" is inserted into the value sequence. Then the procedure in case 1 is followed to process the new information. To address case 3, the system simply appends the parameter value "0.0" to all parameters of such rule. The append procedure is the same as the procedure for case 1.

Access: Our decoding strategy for *compact time sequence* structure allows to search *TAR Archive* takes O(1) time. Given a window id j, the parameter value p_i^j from $\mathcal{V}_i$ is retrieved. For instance, in Fig. 7, for $\mathcal{R}_1$, if $j = 3$, the system needs to locate the correct value p^3 in the value sequence. In B_i, the count of "1" is the length of the value sequence because every time a bit is set to "1", a new value is then appended to the *value sequence*. Finding the offset of the element in the sequence value corresponding to the queried window j is equivalent to counting "1"s up until the jth bit in B_i. For the previous example, if $j = 3$, the binary code up until 3rd bit in B is "001" which only has 1 bit set to "1". So s_1 in the value sequence represents the value of p^3. Given a j, we calculate the offset of the element in the *value sequence* as follows *Offset(j) = HammingWeight(B AND (2^j - 1))*. HammingWeight is a O(1) algorithm for bit counting.

Purge: As parameter values begin to accumulate, the size of the archive grows bigger and bigger. For some applications, historical information inserted at the very beginning may become insignificant. Therefore, an operation that purges such data is necessary. Specifically, the operator deletes the parameter values of the entire ruleset in the archive w.r.t a set of consecutive windows from $\mathcal{T}_1$ to $\mathcal{T}_k$ where $k \leq$ *total number of windows*. For each rule R, the purge operation performs a two-step procedure: (1) Delete the values that correspond to $< \mathcal{D}_1, ..., \mathcal{D}_k >$ for each parameter from the *value sequence*. (2) Update the binary code so that it reflects the changes of the parameter within the rest of the windows. For an input k, if $Offset(k)$ equals to $Offset(k+1)$ which means the values of a parameter in $\mathcal{D}_k$ and $\mathcal{D}_{k+1}$ are mapped to the same s_i in S, then $\{s_j | 1 \leq j \leq i-1\}$ are removed from S. The corresponding B is right shifted $i-1$ times and the first bit of the modified B' is set to "1". If the results of $Offset(k)$ and $Offset(k+1)$ are not identical, then $\{s_j | 1 \leq j \leq i, i = Offset(k)\}$ are removed from S. The corresponding B vector is right shifted i times.

Merge: For some applications, analysts may be interested in recent changes at a fine granularity, but longer term changes at a coarse granularity. To support such time-sensitive roll-up, also called *Natural tilted-time window* or *logarithmic tilted-time window* [6] an efficient merge operation is needed for the *TAR Archive*.

$\{\mathcal{R}\}_i$ denotes the ruleset generated from $\mathcal{D}_i$. The merge operation unions the rulesets, $\bigcup_{w=i}^{j}\{\mathcal{R}\}_w$, that correspond to the set of consecutive windows $[\mathcal{T}_i, \mathcal{T}_j]$ and computes the new parameter value for each rule in the merged window. $[\mathcal{T}_i, \mathcal{T}_j]$ denotes a set of consecutive windows that are requested to be merged, $|\mathcal{D}_j|$ denotes the total number of transactions in $\mathcal{T}_j$. The *support* and *confidence* of a rule in the merged window is computed as follows:

$$Support(\mathcal{R}) = \frac{\sum_{w=i}^{j} |\mathcal{D}_w| \times \mathcal{R}.supp^w}{\sum_{w=i}^{j} |\mathcal{D}_w|} \quad (4)$$

$$Confidence(\mathcal{R}) = \frac{\sum_{w=i}^{j} |\mathcal{D}_w| \times \mathcal{R}.supp^w}{\sum_{w=i}^{j} |\mathcal{D}_w| \times \frac{\mathcal{R}.supp^w}{\mathcal{R}.conf^w}} \quad (5)$$

Based on the definition of *support* and *confidence*, Formula 4 describes the proportion of the transactions within the merged window that *contains* all items of the association; Formula 5 describes the ratio of the number of the transactions that *contains* all items of the association to the number of the transactions that *contains* antecedent items in the merged window. With these new values, the merge operator then further updates both the binary encodings $\mathcal{B}_i$ and value sequences $\mathcal{V}_i$ in the rule entry. For a particular parameter of a rule, if the windows from i to j are merged, then the bits from ith to jth position in its binary code are also merged into one bit. The value of this new bit and $(i+1)$th bit depends on whether the represented parameter value is different from its last parameter value.For example, in Fig. 7 when $\mathcal{T}_2$ and $\mathcal{T}_3$ are requested to be merged, the new binary code becomes "11" and value is 0.3.

6. ROLL-UP SUPPORT ACROSS TIME

In the rule generation step, rules with parameter values below minimal system thresholds are not maintained. Therefore, we may not have the exact parameter values for each rule when multiple windows are merged.The parameter value of a rule in a coarser time period (merged window) is computed based on its parameter values in all periods that compose the window. The calculation is described in Formula 4 & 5. For example, let θ be 0.1, support values of $\mathcal{R}$ in $\mathcal{T}_1$ and $\mathcal{T}_2$ be 0.2 and 0.08 respectively. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are requested to be merged, the exact *support* of R in the new window is $\frac{|\mathcal{D}_1| \times 0.2 + |\mathcal{D}_2| \times 0.08}{|\mathcal{D}_1| + |\mathcal{D}_2|}$. However, let us assume that R had been withdrawn from $\mathcal{T}_2$ because $0.08 < \theta$. Therefore the *support* value of R in $\mathcal{T}_2$ becomes unknown and thus is treated as 0. As a result, we no longer have the exact parameter value of rules in a merged coarser-time-granularity window, rather only an approximate value. Rules withdrawn from a window may fall into 3 cases depending on whether they either fail to satisfy one of the system thresholds or both. The margin of error of the parameters value in merged windows varies case by case. Since the withdrawn rules are absent from the system, the exact reason of the withdrawal becomes unknown. Therefore, we introduce the worst sce-

nario below:

Let $[\mathcal{T}_i, \mathcal{T}_j]$ be a set of consecutive windows that are requested to be merged, $\mathcal{R}$ be a rule that at least appears once in any of these windows, $\mathcal{T}'$ be the windows in which R is absent, $\mathcal{T}''$ be the windows in which $\mathcal{R}$ appears, $S(\mathcal{R})$ and $C(\mathcal{R})$ be the approximated *support* and *confidence* of $\mathcal{R}$ calculated based on Formulas 4&5, $\hat{S}(\mathcal{R})$ and $\hat{C}(\mathcal{R})$ be the exact *support* and *confidence* of $\mathcal{R}$. The worst scenario arises where all absents of $\mathcal{R}$ are caused by $\mathcal{R}.supp \geq \theta$ and $\mathcal{R}.conf < \lambda$.

In this case, $\mathcal{R}$ has a qualified *support*. In order to have a *confidence* value smaller than the threshold, $\mathcal{R}$'s *support* must be larger than θ, however, smaller than λ. Therefore,

$$S(\mathcal{R}) \leq \hat{S}(\mathcal{R}) \leq \frac{\sum_{k=1}^{m} |\mathcal{D}'_k| \times \lambda + \sum_{k=1}^{n} |\mathcal{D}''_k| \times \mathcal{R}.supp_{\mathcal{D}''_k}}{\sum_{w=i}^{j} |\mathcal{D}_w|} \tag{6}$$

The error in confidence is caused by miss counting the withdrawn *support* of R, as well as miss counting of the withdrawn *support* of the antecedent of R. The *confidence* with maximum margin of errors is the following:

$$margin = \frac{\sum_{k=1}^{m} |\mathcal{D}'_k| \times \lambda + \sum_{k=1}^{n} |\mathcal{D}''_k| \times \mathcal{R}.supp_{\mathcal{D}''_k}}{\sum_{k=1}^{m} |\mathcal{D}'_k| + \sum_{k=1}^{n} |\mathcal{D}''_k| \times \frac{\mathcal{R}.supp_{\mathcal{D}''_k}}{\mathcal{R}.conf_{\mathcal{D}''_k}}} \tag{7}$$

Note that this value does not guarantee to be smaller or larger than $\hat{C}(R)$, because the errors are both introduced in the numerator and denominator.

Formula 6 gives the worst scenario for the approximation of *support* and Formula 7 the worst scenario for the approximation of *confidence*. The smaller θ and λ are, the more accurate our approximation will be. Because with a smaller system threshold, less rules would be withdrawn from the window reducing the chance of miss counting.

7. ONLINE QUERY PROCESSING

In this section, we explain how the analytical TARA queries introduced in Section 3 are handled by the **TARA** framework (Algo. 2). Depending upon the query type, the appropriate subroutine is invoked. $Q1$, $Q2$ and $Q3$ are handled by subroutines 2.A, 2.B, 2.C respectively. For $Q4$ once the ruleset is selected by any of the subroutines, routine 2.C is used to return the stable region.

The **response time** for query processing mainly consists of 3 components, namely, $Cost(SearchRegion)$, $Cost(GetRule)$ and $Cost(GetDominatedRegions)$. The TARA storage structures namely *EPS-Index* and *TAR Archive* are in-memory structures. The cost for a region search against *EPS-Index* is $\mathcal{O}(1)$. As illustrated in Fig. 5, locating the regions within the same time interval takes $\mathcal{O}(1)$. By converting input parameters into offsets, the appropriate cell can be found in $\mathcal{O}(1)$ as well. Thus $Cost(SearchRegion) = \mathcal{O}(1)$. Note that the system must generate the stable regions for such time periods in which they don't exist. To iteratively collect all the immediate dominated regions in the domination graph, a breadth-first search (BFS) is required starting at the node containing (*minsupp*,*minconf*) in $\mathcal{T}$. The time complexity of BFS is $\mathcal{O}(|V| + |E|)$. In our case, E $\leq$ (2 $\times$ V) as each vertex has a fanout of at most two edges. Thus,

Algorithm 2: Online Temporal Association Exploration

```
2.A: Temporal Association Mining Query
Input: s,c,{T},matchtype
Output: Ruleset
begin
    {R} ⟵ ∅; {S} ⟵ ∅;
    switch matchtype do
        case single(T)
            {S} ⟵ SearchRegion(s, c, T);
            {S} ⟵ {S} ∪ GetDominatedRegion({S});
        case exact
            T ⟵ any T ∈ {T};
            {S} ⟵ SearchRegion(s, c, T);
            {S} ⟵ {S} ∪ GetDominatedRegion({S});
        case fuzzy
            for each T_i ∈ {T} do
                {S} ⟵ SearchRegion(s, c, T_i);
                {S} ⟵ {S} ∪ GetDominatedRegion({S});
    {R} ⟵ GetRule({S});
```

```
2.B Ruleset Comparison Query
Input: s1,c1,s2,c2,T,matchtype
Output: Rulesets {R}1,{R}2
begin
    s_c ⟵ max(s1, s2); c_c ⟵ max(c1, c2);
    {S}1 ⟵ SearchRegion(s1, c1, T);
    {S}2 ⟵ SearchRegion(s2, c2, T);
    //Collect the regions till it reaches a parameter value
    for i = 1 ; i ≤ 2 ; i++ do
        if s_i ≠ s_c then
            {S}_i ⟵ GetDominatedRegion(S_i, s_c);
        if c_i ≠ c_c then
            {S}_i ⟵ GetDominatedRegion(S_i, c_c);
    {R}1 ⟵ GetRule({S}1);
    {R}2 ⟵ GetRule({S}2);
```

```
2.C Stable Region Query
Input: s,c,{T},matchtype
Output: Region S
begin
    {S} ⟵ ∅;
    switch matchtype do
        case single(T)
            S ⟵ SearchRegion(s,c,T);
        case exact or fuzzy
            for each T_i ∈ {T} do
                {S} ⟵ SearchRegion(s,c,T);
```

$Cost(GetDominatedRegions) = \mathcal{O}(|V|)$. If the trajectory of a rule cannot be obtained from the retrieved regions (*exact* or *single* match), *GetRule* searches the parameter values of the rule in the specified periods in the *TAR Archive*. If the granularity of the specified time periods is coarser than the ones available in the archive, *GetRule* finds the windows that are contained in coarser time periods and computes the parameter value for the merged window. See Sec. 5 for the complexity analysis of accessing the *TAR Archive* and the computation for obtaining the parameter value in the merged window.

8. EXPERIMENTAL EVALUATION

Experimental Setup. Experiments are conducted on a OS X machine with 2.4 GHz Intel Core i5 processor and 8 GB RAM. The system and its competitors are implemented in C++ using Qt Creator with Clang 64-bit compiler.

Datasets. We select a variety of datasets with diverse characteristics here. The benchmark datasets, *T5kL50N100* and *T2kL100N1k*, are generated by the *IBM Quest data generator* [1] modeling transactions in a retail store. We

Table 2: Datasets

	100retail	*T5k*	*T2k*	*webdocs*
Transactions	8,816,200	5,000,000	2,000,000	1,692,082
Unique Items	16,470	23,870	30,551	5,267,656
Avg Len of Tran	10	50	100	177
Size	416.8 MB	1.48 GB	1.38 GB	1.48 GB

Table 3: Thresholds for Indexes

Dataset	H-Mine	TARA&PARAS (supp, conf)
retail	0.0002	(0.0002, 0.1)
T5k	0.0012	(0.0012, 0.2)
T2k	0.001	(0.001, 0.2)
webdocs	0.1123	(0.1123, 0.2)

partition these datasets into 5 equal-sized batches to form the evolving data sources. The *retail* dataset [3] contains 88,163 transactions collected from a Belgian retail supermarket store in 5 months. To study scalability, we replicate this *retail* dataset 100 times. The *webdocs* dataset [13] is built from a spidered collection of web html documents. Both of these real datasets are partitioned into 10 equal-sized batches to form evolving data sources.The statistics of the datasets are summarized in Tab. 2.

Alternate State-of-the-art Techniques. The performance of **TARA** is compared against three competitors. **DCTAR** [9] derives the ruleset directly from the raw data given a parameter configuration. It computes the associations from scratch whenever a new batch of data arrives. **H-Mine** [18] instead pregenerates the intermediate frequent item sets offline. For specific parameter settings, the algorithm utilizes the itemsets to generate the associations online instead of extracting them from the raw data. **PARAS** [10] pregenerates frequent itemsets and rules offline for the entire data set assuming all data is static and given apriori. That is, time is ignored. For our experiments, we construct the PARAS index for a single time period. However at online time if request comes for different periods it then generates the associations from scratch.

Experimental Methodologies: The performance of our approach and state-of-the-art algorithms is measured by:

Offline Preprocessing Time. We measure the single and multiple data batches preprocessing time for **TARA**, H-Mine and PARAS. Since DCTAR does not involve any preprocessing, it is excluded from this measurement.

Online Processing Time. We measure the online processing time for a query averaged over multiple runs (explained in Sec 8.2) to evaluate the speedup.

Size of Pregenerated Information. We compare the sizes of the preprocessed information. DCTAR is again excluded. The size of the tree structure in H-Mine and the size of the *TAR Archive* in **TARA** are thus compared.

8.1 Evaluation of Preprocessing Time

We first compare the preprocessing times for H-Mine, PARAS and **TARA**. In the offline step, as the window slides, H-Mine (1) precomputes the frequent item sets and (2) stores them along with their associated *support* value into a tree structure. Whereas **TARA** (1) precomputes the frequent item sets, (2) derives the ruleset, (3) archives them along with the associated *support* and *confidence* values and (4) updates the *EPS*-Index. PARAS proceeds with the same process as **TARA** except that it does not utilize the archive nor does it keep the pregenerated information from the previous windows. Therefore, the total preprocessing time of PARAS is similar to **TARA** except for the archival time. Fig. 10 compares the preprocessing time of H-Mine and **TARA** for all windows for the *retail*, *T5kL50N100*, *T2kL100N1k* datasets, with the system threshold settings summarized in Tab. 3. As shown, frequent item set generation occupies a relatively large portion of the preprocessing time as compared to other tasks. This confirms prior works [7] that rule generation is more efficient compared to frequent itemset generation. Overall, the additional preprocessing tasks in **TARA** require no more than 20% extra time than H-Mine. This extra time gives significant advantage to **TARA** in terms of truly interactive online performance and support of many advanced exploration operations.

8.2 Evaluation of Online Processing Time

Next, we compare the online processing times (y-axis in log scale) for our proposed operations. The user-specified parameters, namely *minsupp*, *minconf* and *time periods*, are varied. The examined queries fall into two categories: (1) Rule trajectory and parameter recommendation queries and (2) Ruleset comparison queries. In the first experiment, we test the performance of **TARA** against the three competitors using several query types, namely $Q1$ and $Q3$ in *single match* mode. Second, we use $Q2$ in *exact match* mode to test the performance of **TARA** against others. We choose $Q1$, $Q2$ and $Q3$ because they cover the major exploration operations and subroutines in the online processing phase.

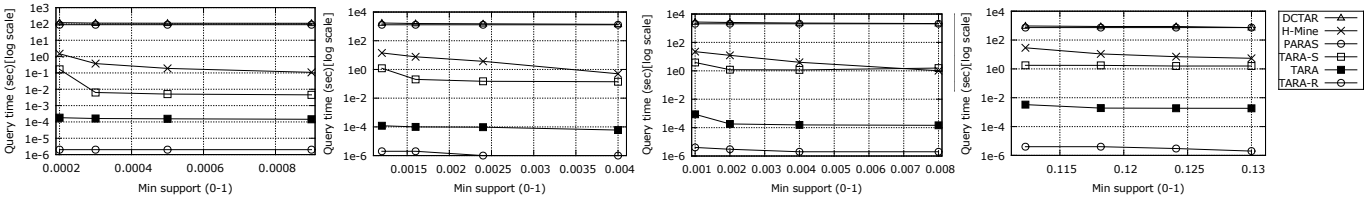
8.2.1 Trajectory and Parameter Recommendation

To process $Q1$, the system needs to find the rules that satisfy *minsupp* and *conf* in a single *time period* and examine their parameter values in other specified time periods. For DCTAR, it mines the rules from the transactions that fall into the last window and examines their parameter values by processing the transactions that fall into the 3 previous windows. For PARAS, the process is identical except that the rules are retrieved from the PARAS index built based upon the newest window. For H-Mine, the rules are derived and examined by using its item set index.

Impact of Varying Support and Confidence. To determine the effect of *minsupp*, we conduct several experiments by fixing *minconf* to a constant value and varying the *minsupp* value. Fig. 8 illustrate the query processing times for *retail*, *T5kL50N100*, *T2kL100N1k* and *webdocs* datasets with fixed *minconf* 0.4, 0.2, 0.2 and 0.4, respectively.

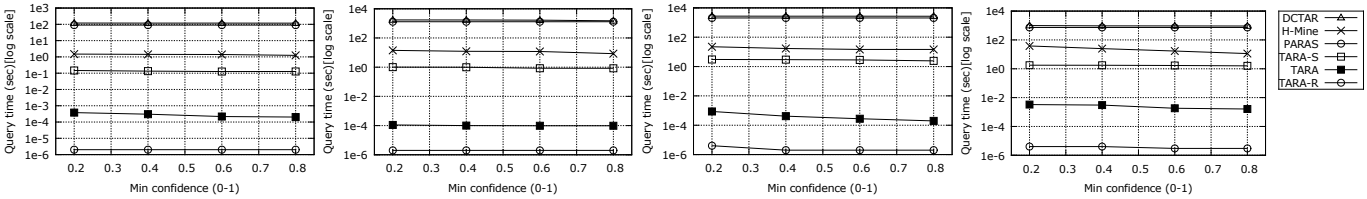
We observe that, **TARA** consistently outperforms DCTAR and PARAS by 6,7,7 and 5 orders and H-Mine by 3, 4, 4 and 4 orders of magnitude for *retail*, *T5kL50N100*, *T2kL100N1k* and *webdocs* datasets respectively. **TARA-S** stands for the implementation of **TARA** with the rule index inside each *time sensitive stable region* to support content based exploration ($Q5$). The merging of indexes when *dominated regions* are being collected incurs extra costs as compared to the TARA system without these rule indexes. Especially when the number of rules in the result is small, this extra cost results in similar or slower response time compared to H-Mine as shown in Figs. 8(b) and (c). The reason of the fast response of **TARA** is that it prepares sufficient amount of information in the offline stage, so that answering such queries is simply about searching the **TARA** index.

TARA-R shows the response time of returning the *time-sensitive stable region* which answers $Q3$. Since PARAS always builds the index for the latest window, in this particular experiment, it achieves the same response time as



(a) retail (b) T5kL50N100 (c) T2kL100N1k (d) webdocs

Figure 8: Rule Trajectory and Parameter Recommendation: Varying Support



(a) retail (b) T5kL50N100 (c) T2kL100N1k (d) webdocs

Figure 9: Rule Trajectory and Parameter Recommendation: Varying Confidence

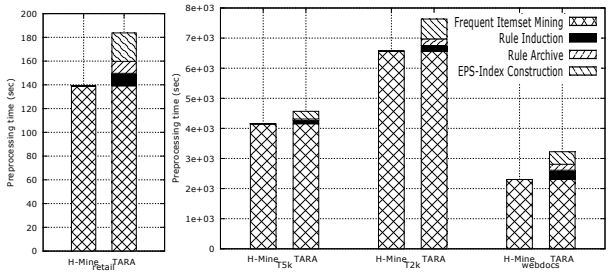


Figure 10: Preprocessing Time

TARA because only regions that fall into the latest window are considered. All other systems are not capable of answering $Q3$. That is, using DCTAR and H-Mine, an analyst would need to generate all possible rules in the specified time period and then investigate all to find the answer.

Impact of Varying Confidence. Next, we fix the *minsupp* to a constant value and vary the *minconf* value. Fig. 9 illustrates the query processing times for *retail*, *T5kL50N100*, *T2kL100N1k* and *webdocs* datasets with fixed *minsupp* 0.0002, 0.0012, 0.0012 and 0.1123, respectively. Overall, both **TARA** and **TARA-S** consistently perform several orders of magnitude better than the three competitors.

8.2.2 Ruleset Comparison Queries

$Q2$ returns the differences of the rulesets w.r.t two parameter settings that share the same time specification. In this particular experiment, the query is configured with the *exact match* mode. It returns the differences of two parameter setting across 4 windows. Since the DCTAR and H-Mine do not support such query, we implement a subroutine in their rule derivation module to determine if the parameter value of the rule satisfies one setting but not the other. This subroutine is optimized so that it does not generate the overlapping ruleset w.r.t 2 different settings. In this experiment, we either fix *minsupp* or *minconf* and vary the other one.

Impact of Varying 2^{nd} Support. Fig. 11 illustrates the query processing times for *retail*, *T5kL50N100*, *T2kL100N1k* and *webdocs* datasets. The fixed *min parameters* for these datasets are ($minsupp_1$, $minconf_1$, $minconf_2$): (0.0002, 0.4, 0.4), (0.0012, 0.2, 0.2), (0.0012, 0.2, 0.2) and (0.1123, 0.4, 0.4), respectively. The query processing times increase with an increase in the *minsupp* because the increase of the deviation from $minsupp_1$ to $minsupp_2$ results in larger differences between the two parameter settings. In particular, **TARA** outperforms DCTAR and PARAS by 6,7,6 and 6 orders, H-Mine by 4, 5, 4 and 4 orders for *retail*, *T5kL50N100*, *T2kL100N1k* and *webdocs* datasets, respectively.

Impact of Varying 2^{nd} Confidence. Fig. 12 illustrates the query processing times for *retail*, *T5kL50N100*, *T2kL100N1k* and *webdocs* datasets. The fixed *min parameters* for these four datasets are ($minsupp_1$, $minconf_1$, $minsupp_2$): (0.0002, 0.4, 0.0002), (0.0012, 0.2, 0.0012), (0.0012, 0.2, 0.0012) and (0.1123, 0.4, 0.1123), respectively. **TARA** consistently performed several orders of magnitude better than the three competitors.

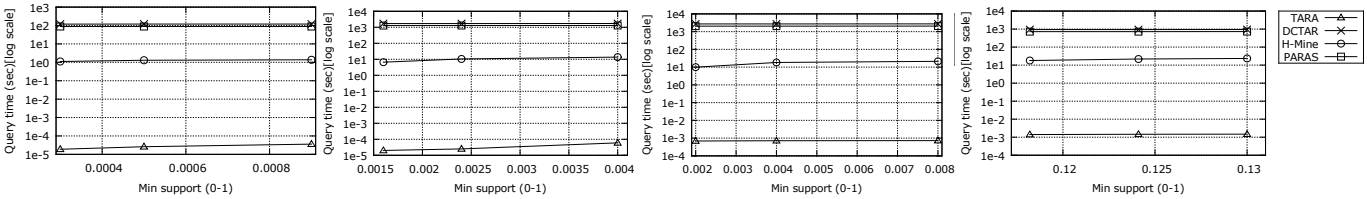
8.3 Evaluation of Archive Size

We compare the sizes of the pregenerated information in **TARA**, H-Mine and PARAS. For H-Mine, the size of the structure is determined by the number of frequent item sets times the number of processed partitions, while the size of pre-stored information in **TARA** is determined by the size of the *TAR Archive*. PARAS only pregenerates the association in a single window. Its maximum size is $3^n - 2^n + 1$ where n is the unique items in that particular window. The actual index sizes can be estimated by multiplying the number of instances with the average space required per instance.

Fig 13 shows the size of the H-Mine Index, *TAR Archive* and the actual number of uncompressed rule parameter values for our four datasets with the system threshold settings summarized in Tab. 3. As **TARA** pre-generates rules instead of only the item sets, the size of the *TAR Archive* is larger than the H-Mine index. However, our encoding technique achieves favorable compression as compared to uncompressed rule parameter values.

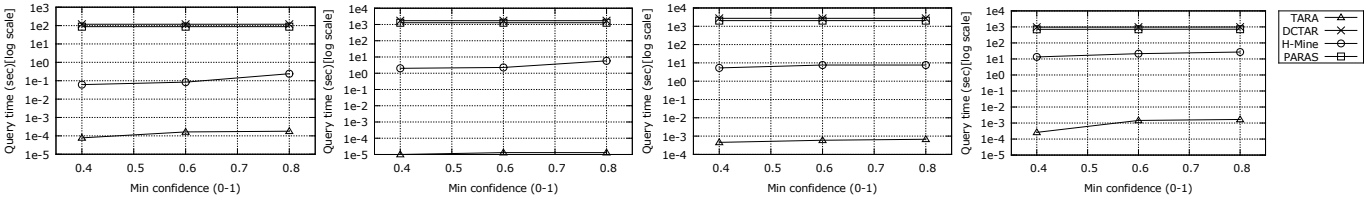
9. RELATED WORK

Temporal association mining. Adding the time dimension in the context of association rules was first mentioned in [14]. However, while more follow-on works [6, 9, 18] improve the efficiency of temporal association mining by maintaining intermediate frequent item sets, all of these approaches require the user to input a specific parameter setting. This one-at-a-time approach not only limits efficiency,



(a) retail (b) T5kL50N100 (c) T2kL100N1k (d) webdocs

Figure 11: Ruleset Comparison: Varying 2^{nd} Support



(a) retail (b) T5kL50N100 (c) T2kL100N1k (d) webdocs

Figure 12: Ruleset Comparison: Varying 2^{nd} Confidence

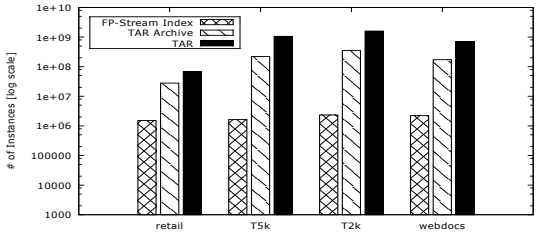


Figure 13: Size of the *TAR Archive*

but also provides very limited feedback for the user.

Interestingness of temporal associations. [12, 17] identify the importance of analyzing the interestingness measures of associations. In the context of time-variant data, [11] measures the changes of the interestingness of the association w.r.t its histories. It is suggested that the interest in the rule itself is primarily determined by the interestingness of its change over time. Neither of these works tackle interactive mining through precomputation. In contrast, we explore the space of interestingness parameters for prestoring data mining results to facilitate fast online mining.

Interactive association mining. Prior works [4, 5, 10] have explored the space of parameters for handling data mining requests. However this work is restricted to static data. These approaches do not consider the time dimension as a property of the pattern. Instead we now study the problem of incorporating the time dimension into the association mining exploration process.

10. CONCLUSION

We present the first framework for interactive temporal association analytics. Our **TARA** framework employs a novel evolving parameter space model for pre-generating rules such that near real-time performance is guaranteed for online mining. In a variety of tested cases, **TARA** outperforms the three state-of-the-art competitor techniques, each by several orders of magnitude, while offering a holistic exploration experience supporting new classes of time-variant rule analytics.

11. REFERENCES

[1] R. Agrawal, M. Mehta, J. C. Shafer, R. Srikant, A. Arning, and T. Bollinger. The quest data mining system. In *KDD*, volume 96, pages 244–249, 1996.

[2] J. M. Ale and G. H. Rossi. An approach to discovering temporal association rules. In *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1*, pages 294–300. ACM, 2000.

[3] T. Brijs, G. Swinnen, K. Vanhoof, and G. Wets. Using association rules for product assortment decisions: A case study. In *KDD*, pages 254–260, 1999.

[4] L. Cao, M. Wei, D. Yang, and E. A. Rundensteiner. Online outlier exploration over large datasets. In *Proceedings of the 21th ACM SIGKDD*, pages 89–98. ACM, 2015.

[5] S. Chaudhuri, H. Lee, and V. R. Narasayya. Variance aware optimization of parameterized queries. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 531–542. ACM, 2010.

[6] C. Giannella, J. Han, J. Pei, X. Yan, and P. S. Yu. Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining*, 212:191–212, 2003.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[8] H. Li, Y. Wang, D. Zhang, M. Zhang, and E. Y. Chang. Pfp: parallel fp-growth for query recommendation. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 107–114. ACM, 2008.

[9] Y. Li, P. Ning, X. S. Wang, and S. Jajodia. Discovering calendar-based temporal association rules. *Data & Knowledge Engineering*, 44(2):193–218, 2003.

[10] X. Lin, A. Mukherji, E. A. Rundensteiner, C. Ruiz, and M. O. Ward. Paras: A parameter space framework for online association mining. *Proceedings of the VLDB Endowment*, 6(3):193–204, 2013.

[11] B. Liu, Y. Ma, and R. Lee. Analyzing the interestingness of association rules from the temporal dimension. In *Proceedings of IEEE ICDM*, pages 377–384. IEEE, 2001.

[12] B. Liu, K. Zhao, J. Benkler, and W. Xiao. Rule interestingness analysis using olap operations. In *Proceedings of the 12th ACM SIGKDD*, pages 297–306. ACM, 2006.

[13] C. Lucchese, S. Orlando, R. Perego, and F. Silvestri. Webdocs: a real-life huge transactional dataset. In *FIMI*, 2004.

[14] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic association rules. In *Data Engineering, 1998. Proceedings., 14th International Conference on*, pages 412–421. IEEE, 1998.

[15] X. Qin, R. Ahsan, X. Lin, E. A. Rundensteiner, and M. O. Ward. iparas: Incremental construction of parameter space for online association mining. In *Proceedings of the 3rd BigMine*, pages 149–165, 2014.

[16] S. Ramaswamy, S. Mahajan, and A. Silberschatz. On the discovery of interesting patterns in association rules. In *VLDB*, volume 98, pages 368–379. Citeseer, 1998.

[17] S. Sahar. Interestingness preprocessing. In *Proceedings of IEEE ICDM*, pages 489–496. IEEE, 2001.

[18] K. Verma and O. P. Vyas. Efficient calendar based temporal association rule. *ACM SIGMOD Record*, 34(3):63–70, 2005.