InfoMiner: Mining Surprising Periodic Patterns

Jiong Yang
IBM Watson Research Center
jiyang@us.ibm.com

Wei Wang
IBM Watson Research Center
ww1@us.ibm.com

Philip S. Yu
IBM Watson Research Center
psyu@us.ibm.com

ABSTRACT

In this paper, we focus on mining surprising periodic patterns in a sequence of events. In many applications, e.g., computational biology, an infrequent pattern is still considered very significant if its actual occurrence frequency exceeds the prior expectation by a large margin. The traditional metric, such as *support*, is not necessarily the ideal model to measure this kind of *surprising patterns* because it treats all patterns equally in the sense that every occurrence carries the same weight towards the assessment of the significance of a pattern regardless of the probability of occurrence. A more suitable measurement, *information*, is introduced to naturally value the degree of surprise of each occurrence of a pattern as a continuous and monotonically decreasing function of its probability of occurrence. This would allow patterns with vastly different occurrence probabilities to be handled seamlessly. As the accumulated degree of surprise of all repetitions of a pattern, the concept of *information gain* is proposed to measure the overall degree of surprise of the pattern within a data sequence. The *bounded information gain* property is identified to tackle the predicament caused by the violation of the *downward closure* property by the information gain measure and in turn provides an efficient solution to this problem. Empirical tests demonstrate the efficiency and the usefulness of the proposed model.

1. INTRODUCTION

Periodic pattern discovery is an important problem in mining time series data and has wide applications. A periodic pattern is an ordered list of events, which repeats itself in the event sequence. It is useful in characterizing the cyclic behavior of the time series. As a newly developed research area, most previous work on mining time series data addresses the issue by utilizing the support (number of occurrences) as the metric to identify the important patterns from the rest [4, 13]. A qualified pattern in the support model must occur sufficient number of times. In some applications, e.g., market basket, such a model is proved to be very meaningful and important. However, in some other applications, the number of occurrences may not represent the significance of a pattern. Consider the following examples.

- *Web server load.* Consider a web server cluster consisting of 5 servers. The workload on each server is measured by 4 ranks: *low*, *relatively low*, *relatively high*, and *high*. Then there are $4^5 = 1024$ different events, one for each possible combination of server states. Some preliminary examination of the cluster states over time might show that the state fluctuation complies with some periodic behavior. Although the high workload on all servers may occur at a much lower frequency than other states, pattern(s) involving it may be of more interests to some system administrators if the occurrence of such pattern(s) contradicts the prior expectation.
- *Earthquake.* Earthquakes occur very often in California. It can be classified by its magnitude and type. Scientist may be interested in knowing whether there exists any inherent seismic period so that prediction can be made. Note that any unidentified seismic periodicity involving big earthquake is much more valuable even though it occurs at a much lower frequency than smaller ones.

In the above examples, we can see that users may be interested in not only the frequently occurred patterns, but also the surprising patterns (i.e., beyond prior expectation) as well. A large number of occurrences of an "expected" frequent pattern sometimes may not be as interesting as a few occurrences of an "expected" rare pattern. The support model is not ideal for these applications because, in the support model, the occurrence of a pattern carries the same weight (i.e., 1) towards its significance, regardless of its likelihood of occurrence. Intuitively, the assessment of significance of a pattern in a sequence should take into account the expectation of pattern occurrence (according to some prior knowledge). Recently, many research has been proposed [1, 3, 5, 6, 8, 9, 10, 11, 12, 15] towards this objective. We will furnish an overview in the next section. In this paper, a new model is proposed to characterize the class of so-called *surprising* patterns (instead of frequent patterns). We will see that our model not only has a solid theoretical foundation but also allows an efficient mining algorithm.

The measure of *surprise* should have the following properties. (1) The surprise of a pattern occurrence is anti-monotonic with respect to the likelihood that the pattern may occur by chance (or by prior knowledge). (2) The metric should have some physical meaning, i.e., not arbitrary created. It is fortunate that the *information* metric [2] which is widely studied and used in the communication field can fulfill both requirements. Intuitively, information is a measurement of how likely a pattern will occur or the amount of "surprise" when a pattern actually occurs. If a pattern is expected to occur frequently based on some prior knowledge or by chance, then an occurrence of that pattern carries less information. Thus, we use information to measure the surprise of an occurrence of a pattern. The *information gain* metric is introduced to represent the accumulated information of a pattern in an event sequence and is used to assess the degree of surprise of the pattern. In the remainder of this paper, we refer to this model as the *information model*.

The information model is different from the support model. For a given minimum information gain threshold, let Ψ be the set of patterns that satisfy this threshold. Under the support model, in order to find all patterns in Ψ when event occurrence frequencies are vastly different, the minimum support threshold has to be set very low. A major problem could rise from this: too many patterns discovered. This means that the patterns with most information gain are buried in a sea of patterns with relatively low information gain. This could be a large burden for the end user to distinguish the surprising patterns (i.e., patterns with high information gain) from the rest. In addition, since a large number of patterns have to be discovered, the support model may yield an inefficient algorithm.

Although the information gain is a more meaningful metric for the problems addressed previously, it does not preserve the *downward closure* property (as the *support* does). For example, the pattern (a_1, a_2) may have enough information gain while both $(a_1, *)$ and $(*, a_2)$ do not. The $*$ symbol represents the "don't care" position. We cannot take advantage of the standard pruning technique (e.g., Apriori algorithm) developed for mining association rules and temporal patterns [4]. Fortunately, we are able to identify the *bounded information gain property* where patterns with inextensible prefixes could not be surprising (given some information gain threshold) and can be excluded from consideration at a very early stage. This motivates us to devise a recursive algorithm as the core of our pattern discovery tool, InfoMiner. In summary, this paper has the following contributions.

- We propose a new mining problem that is to find surprising periodic patterns in a sequence of data.
- The concepts of information and information gain are proposed to measure the degree of surprise of the pattern exhibited in a sequence of data.
- Since the downward closure does not hold with information gain, we devise an efficient algorithm to mine the surprising patterns and associated subsequences based on the *bounded information gain property* that is preserved by the information gain metric.

The remainder of this paper is organized as follows. Some related work is discussed in Section 2. We present the information model in Section 3. Section 4 discusses the detailed algorithm of finding patterns whose information gain is above a certain threshold. Experimental results are shown in Section 5. Finally, we draw the conclusion in Section 6.

2. RELATED WORK

In this section, we provide a brief overview of recent advances that is closely related to our work presented in this paper. There is much work in mining periodic patterns[4, 7, 13]. Despite the difference in problem formulation, most work adopted the support as the measure of interestingness (or significance) and aimed at discovering frequent patterns. Recently, many efforts have been carried out to address the potential disadvantages associated with the support model and to propose alternative solutions. Due to space limitations, we are focusing on the related work of models of interestingness.

Multiple supports scheme was introduced by Liu et. al. [6] and later extended by Wang et al. [12] to find itemsets which do not occur frequently overall, but have high correlation to occur with some other items. The support threshold to qualify a frequent itemset can be specified as a fraction of the minimum support of all items [6] or subsets of items [12] in the itemset. This variable support has similar effect as the information gain introduced in this paper. However, there exists some fundamental difference between these two concepts. For example, if the support of item A, B, and C is 0.01, 0.02, 0.8, respectively, then the support threshold to qualify itemset AB and AC is the same. Nevertheless, the itemset AC is expected to occur more frequently than AB because the support of C is much larger than that of B. This aspect was not fully taken into account by the multiple support model. In contrast, the information gain metric proposed in this paper would capture the difference of occurrences between B and C.

Mining patterns that are statistically significant (rather than frequent) becomes a popular topic. Brin et al. [3] first introduced the concept of correlation and it was shown that in many applications the correlation measurement can reveal some very important patterns. The Chi-squared test was used to test the correlation among items. Instead of explicitly enumerating all correlated itemsets, the border comprising the set of minimal correlated itemsets[1] is identified, and no further distinction is made on the degree of correlation of itemsets above the border (i.e., supersets of some itemset on the border). This model sometimes becomes sub-optimal. As shown in Figure 1, itemsets A and B are highly correlated but C is independent of them[2]. In addition, $\{A, B, C, D\}$ is also highly correlated. We can view that the degree of correlation of $\{A, B, C\}$ is not as strong as that of $\{A, B\}$ and $\{A, B, C, D\}$. This observation can also be confirmed by the Chi-squared test[3]. In many applications, users are only interested in the itemsets such as $\{A, B\}$ and $\{A, B, C, D\}$, but not $\{A, B, C\}$. However, [3] cannot distinguish between $\{A, B, C\}$ and $\{A, B, C, D\}$ once $\{A, B\}$ is identified as a correlated itemset. Another potential drawback of this model is the expensive computation required by this model. The running time of all patterns with i-correlated items is $O(n \times |CAND| \times \min\{n, 2^i\})$ where n and $|CAND|$ are the number of transactions and the number of candidates at ith level, respectively. To overcome these drawbacks, Oates et al. [9] proposed models for statistical dependencies using G statistic and devised randomized algorithms to produce approximate results. In contrast, our model not only can successfully identify $\{A, B\}$ and $\{A, B, C, D\}$ without including $\{A, B, C\}$ but also leads to a much more efficient deterministic algorithm.

Another important advance is accomplished in mining so-called *unexpected patterns*. Berger et al. [1] proposed a probabilistic measure of interestingness based on unexpectedness

[1] A minimal correlated itemset is a correlated itemset whose subsets are all independent.

[2] $Prob(AB) \times Prob(C) = \frac{1}{2} \times \frac{2}{3} = Prob(ABC)$.

[3] In general, the chi-squared test requires a large sample. For the demonstration purpose only, we assume that the chi-squared test is valid in this example.

Transaction ID	Items
1	ABCD
2	ABFG
3	CEGF
4	ABCD
5	CEGH
6	CEFH

Figure 1: An example of transaction set

in the context of temporal logic, whereby a pattern is deemed interesting if the ratio of the actual number of occurrences of the pattern exceeds the expected one by some user defined threshold. Solving the problem in its general frame is in nature NP-hard and hence some heuristics are proposed to produce an approximate answer. Our model presented in this paper can in fact achieve a similar objective but enables an efficient solution without sacrificing the accuracy.

3. MODEL OF SURPRISING PATTERNS

In this paper, we adopt the general model for periodic patterns proposed in [4] with one exception: Instead of finding *frequent* patterns[4], our goal is to discover **surprising** patterns in an event sequence. Let $E = \{a_1, a_2, \dots\}$ be a set of distinct events. The event sequence is a sequence of events in E. A periodic pattern is a list of l events that may occur recurrently in the sequence with period length l. The **information** carried by an event a_i ($a_i \in E$) is defined to be $I(a_i) = -\log_{|E|} Prob(a_i)$ where $\mid E \mid$ and $Prob(a_i)$ are the number of events in E and the probability that a_i occurs, respectively. The probability $Prob(a_i)$ can be assessed in many ways. Without loss of generality, we adopt the following way to assess the information carried by an event. $Prob(a_i) = \frac{Num_D(a_i)}{N}$ for all $a_i \in E$ where $Num_D(a_i)$ and N are the number of occurrences of the event a_i in an event sequence D and the length of D, respectively. This means that an occurrence of a frequent event carries less information/surprise than a rare event. Note that this also coincides with the original intention of *information* in the data communication community. We shall show later that this gives us the opportunity to handle patterns with divergent probabilities seamlessly. Theoretically speaking, the base of the logarithm function can be any real number that is greater than 1. Typically, $\mid E \mid$ is chosen to be the base to play a normalization role in the computation (i.e., $I(a_i) = 1$ if $Prob(a_i) = \frac{1}{|E|}$). For example, the sequence in Figure 2 contains 6 different events a_1, a_2, a_3, a_4, a_5, and a_6. Their probabilities of occurrence and information are shown in Table 1.

A **pattern** of **length** l is a tuple of l events, each of which is either an event in E, or the **eternal event** (represented by symbol $*$). An eternal event is a virtual event that matches any event in E and is used to represent the "don't care" position in a pattern. By definition, the information of the eternal event * is $I(*) = -log_{|E|} Prob(*) = 0$ since $Prob(*) = 1$. An intuitive interpretation is that the occurrence of an event that is known to be always true does not provide any "new information" or "surprise". A pattern P with length l is in the form

[4]For a pattern s in a sequence $d_1, d_2, \dots, d_N$, the frequency count is defined as $\mid \{i \mid 0 \le i \le \frac{N}{|s|}$, and the string s is true in $d_{i|s|+1}, \dots, d_{i|s|+|s|}\} \mid$.

Table 1: Probability of Occurrence and Information

Event	Probability	Information
a_1	$\frac{6}{40} = 0.15$	$-\log_6(0.15) = 1.06$
a_2	$\frac{8}{40} = 0.20$	$-\log_6(0.20) = 0.90$
a_3	$\frac{12}{40} = 0.30$	$-\log_6(0.30) = 0.67$
a_4	$\frac{5}{40} = 0.125$	$-\log_6(0.125) = 1.16$
a_5	$\frac{6}{40} = 0.15$	$-\log_6(0.15) = 1.06$
a_6	$\frac{3}{40} = 0.075$	$-\log_6(0.075) = 1.45$

of $(p_1, p_2, \dots, p_l)$ where $p_i \in E \cup \{*\}$ $(1 \le i \le l)$ and at least one position has to be filled by an event in E[5]. P is called a **singular pattern** if only one position in P is filled by an event in E and the rest positions are filled by *. Otherwise, P is referred to as a **complex pattern**. For example, $(*, a_3, *)$ is a singular pattern of length 3 and $(a_2, a_6, *, a_2)$ is a complex pattern of length 4. Note that an event may have multiple occurrences in a pattern. As a permutation of a list of events, a pattern $P = (p_1, p_2, \dots, p_l)$ will occur with a probability $Prob(P) = Prob(p_1) \times Prob(p_2) \times \cdots \times Prob(p_l)$ in a random event sequence if no advanced knowledge on correlation among these events is assumed. Then the information carried by P is $I(P) = -\log_{|E|} Prob(P) = I(p_1) + I(p_2) + \cdots + I(p_l)$. It follows directly that the information of a singular pattern always equals to the information of the event specified in the pattern. This property provides a natural bridge between events and patterns. For example, $I((*, a_6, *, *)) = I(a_6) = 1.45$ and $I((a_2, a_6, *, *)) = I(a_2) + I(a_6) = 0.90 + 1.45 = 2.35$ according to Table 1.

Given a pattern $P = (p_1, p_2, \dots, p_l)$ and a segment S of l events $s_1, s_2, \dots, s_l$, we say that S **supports** P if, for each event p_i $(1 \le i \le l)$ specified in P, either $p_i = *$ or $p_i = s_i$ is true. The segment a_2, a_6, a_3, a_2 supports the pattern $(a_2, a_6, *, *)$ while the segment a_1, a_6, a_4, a_5 does not. To assess whether a pattern of length l is surprising in an event sequence D, D is viewed as a list of disjoint contiguous segments, each of which consists of l events. The number of segments that support P is also called the **support** of P (denoted by $Support(P)$). The event subsequence[6] consisting of the list of segments that support P is called the **projected subsequence** of D on P. In Figure 2, the event sequence consists of 40 events. When mining periodic patterns with $l = 4$, it can be viewed as a list of 10 segments, each of which contains 4 events. The support of $(a_2, a_6, *, *)$ is 3 in the sequence and the projected subsequence on $(a_2, a_6, *, *)$ is a_2, a_6, a_3, a_2 a_2, a_6, a_5, a_2, a_2, a_6, a_2, a_2. As a measurement of the degree of surprise of a pattern P in an event sequence D, the **information gain** of P in D is defined as $G(P) = I(P) \times (Support(P) - 1)$. Since our objective is to mine surprising periodic patterns, an event combination appearing in the event sequence which never recurs is of little value in this problem domain. Therefore, in the proposed model, only recurrences of a pattern will have positive contribution to the information gain. $Support(P) - 1$ is indeed the number of recurrences of P. In the rest of the paper, we will use $Repetition(P)$ to denote it. For example, $Repetition((a_2, a_6, *, *)) = 3 - 1 = 2$ and $G((a_2, a_6, *, *)) = 2.35 \times 2 = 4.70$ in Figure 2.

Similar to the support model, an information gain thresh-

[5]This requirement is employed to exclude the trivial pattern $(*, *, \dots, *)$ from being considered.

[6]Given two sequences D and D', D is a **subsequence** of D' if D can be obtained by removing some events in D'.

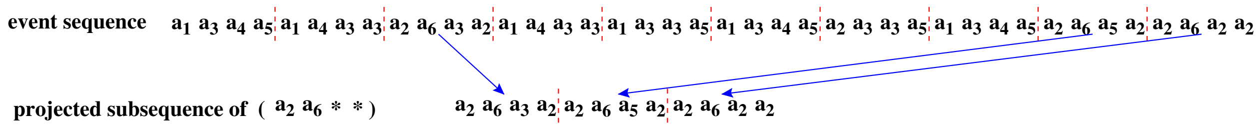

Figure 2: Event sequence and projected subsequence

old, min_gain, is specified by the user to define the minimum information gain to qualify a surprising pattern. Given an event sequence and an information gain threshold, the goal is to discover all patterns whose information gains in the event sequence exceed the threshold. Obviously, the proper value of this threshold is application dependent and may be specified by a domain expert. A user may use the following heuristic to choose the value of min_gain. If a pattern with probability p is regarded as a surprising pattern when it repeats itself by at least n times in the sequence. Then the min_gain can be set to $(-\log_{|E|} p) \times n$ where $-\log_{|E|} p$ is the information of the pattern.

To facilitate the explanation in the rest of the paper, we refer to a pattern, say P, as a **subpattern** of another pattern, say P', if P can be generated by replacing some event(s) in P' by the eternal event *. P' is called a **superpattern** of P. For example, $(a_2, a_6, *, *)$ and $(*, a_6, *, *)$ are subpatterns of $(a_2, a_6, *, a_2)$. The pattern-subpattern relationship essentially defines a partial order among patterns of the same length.

4. PROJECTION-BASED ALGORITHM

Previous work on pattern discovery usually utilizes the Apriori property that can be stated as "if a pattern P is significant, then any subpattern of P is also significant". This property holds for the support model but is not true in the information model. For example, in Figure 2, the information gain $G((a_2, *, *, *)) = 0.90 \times 3 = 2.70$ which is less than the information gain of pattern $(a_2, a_6, *, *)$ (i.e., 4.70). If the threshold is set to be 4.5, then $(a_2, a_6, *, *)$ qualifies while $(a_2, *, *, *)$ does not. (Note that a_6 is an infrequent event which occurs only three times in the event sequence.) This implies that the algorithms developed for the support model are not applicable. The pruning power of the support model essentially comes from the fact that if we know a pattern is not valid, we do not need to examine its superpatterns. Can we achieve a similar pruning power under the information model? To answer this question, we first introduce a concept called *extensible prefix*.

DEFINITION 4.1. *For a pattern $P = (p_1, p_2, \ldots, p_l)$, the tuple $(p_1, p_2, \ldots, p_i)$ is called a* **prefix** *of P where $1 \leq i \leq l$.*

A prefix is part of a pattern. A pattern can be generated by appending more events to the prefix. For instance, $(a_1, *, a_4)$ is a prefix of patterns $(a_1, *, a_4, a_3)$ $(a_1, *, a_4, a_2)$, $(a_1, *, a_4, a_4)$, and $(a_1, *, a_4, *)$, etc.

DEFINITION 4.2. *Given an information gain threshold, a prefix is* **extensible** *in a given sequence if at least one pattern with this prefix is surprising (i.e., whose information gain meets the threshold), and is* **inextensible** *otherwise.*

It follows from the definition that, all prefixes of a surprising pattern are extensible, and any pattern with an inextensible prefix cannot be a surprising pattern. In order to find all surprising patterns, we only need to examine extensible prefixes. Fortunately, we have the following theorem.

THEOREM 4.1. **(Bounded information gain)** *Given an information gain threshold min_gain and a period length l, a prefix $P_i = (p_1, p_2, \ldots, p_i)$, is not extensible iff $Repetition(P_i) < \frac{min\_gain}{max\_info}$ where $max\_info = I(P_i) + \sum_{k=i+1}^{l} f_k$ is the maximum information that can be carried by any pattern with prefix P_i and f_k is the highest information that can be carried by any potential event at the (unspecified) position k of the pattern for a given sequence.*

The proof of this theorem is in [14]. Once a prefix is deemed to be inextensible, we will immediately eliminate it from any further examination. Only extensible prefixes will be used to extend to longer (extensible) prefixes and to construct candidate patterns. Furthermore, given a period length l, for any prefix $P_i = (p_1, p_2, \ldots, p_i)$, consider an unspecified position k where $i < k \leq l$. Not every event can potentially serve on position k of a surprising pattern with prefix P_i. An event $e \in E$ can possibly be a candidate for position k only if e recurs on position k sufficient number of times. In particular, the minimum required number of repetitions is $min\_rep = \frac{min\_gain}{I(P_i)+\sum_{j=i+1}^{l} f_j}$. This indicates that only a (small) subset of events may serve as the candidates for each unspecified position and we can limit our search to these candidate events, and also leads to the following remark..

REMARK 4.1. **(Candidate pruning)** *For any two prefixes $P_{i1} = (p_1, p_2, \ldots, p_{i1})$ and $P_{i2} = (p_1, p_2, \ldots, p_{i2})$ where $i1 < i2$, any candidate event e on position k ($i2 < k \leq l$) for prefix P_{i2} must also be a candidate on position k for prefix P_{i1}, where l is the period length.*

The proof of this remark is in [14]. Remark 4.1 states that, as the prefix grows longer by filling some unspecified positions, the candidate set of a still open position will only shrink monotonically. This provides us with the opportunity to mine surprising patterns by only proceeding with candidate event for each position. Powered with this pruning technique, we develop a progressive approach by starting from extensible prefixes that contain only one filled position (the remaining positions are unspecified) and then proceeding to extensible prefixes with more filled positions gradually to achieve the maximum pruning effect. A candidate event list for each open (i.e., unspecified) position is maintained and continuously pruned when more positions are filled. This process continues until all surprising patterns have been identified by examining and extending extensible prefixes. A depth first algorithm is then developed to generate all qualified patterns in a recursive manner.

Another observation we made is that a segment shall not support a pattern P, if it does not support one of P's prefixes. To expedite the process, when we examine a prefix Q, we may screen out those segments that do not support Q and only retain the projected subsequence of Q so that the evaluation of any prefix containing Q would not have to resort to the original sequence. Note that the projected subsequence will also be further pruned every time the algorithm proceeds to a prefix with more specified positions. For a given period l, starting with a pattern frame of l slots (without any specific event assigned), potential patterns (or prefixes of potential patterns) are generated progressively by every time assigning a candidate event to a still-open position. Such assignment may lead to both a refinement of event candidates for the remaining position(s) by applying the above property and a further projection of the projected subsequence by specifying some of the remaining open positions.

The main procedure of mining patterns for a given pattern period is described in the procedure *InfoMiner*. *InfoMiner* is a recursive function. At the kth level of recursion, the patterns with k non-eternal events are examined. For example, all singular patterns (e.g., $(a_1, *, *, *)$) are examined at the initial invocation of *InfoMiner*; at the second level of invocations of *InfoMiner*, all candidate patterns with two non-eternal events (e.g., $(a_1, *, *, a_5)$) are evaluated; an so on. This is achieved by extending the extensible prefixes to include an additional event during each invocation of *InfoMiner* and passing the new prefixes to the next level of recursion. Notice that at most l levels of recursion may be invoked to mine patterns of period l.

Being more specific, at each level of the recursion, we evaluate patterns with certain prefixes in a projected subsequence S. Starting from a null prefix and the sequence in Figure 2, the number of repetitions for each candidate event of each open position is collected from the projected subsequence S. Then the bounded information gain property is employed to refine the candidate list for each remaining open position. Finally, for each open position i and each event e in the refined candidate list, a new prefix is created by extending the original one to include the event e on the ith open position. Note that this newly created prefix is guaranteed to be extensible and would have the same number of repetitions as the event e at position i. A candidate pattern $P = (prefix, \overbrace{* \cdots *}^{l-i})$ is constructed by filling all remaining open positions following the prefix with the eternal event *. We then verify whether P has sufficient information gain. The projected subsequence on each new prefix is also generated. The pseudo code of *InfoMiner* can be found in [14].

Figure 3(a) shows the projected subsequence of prefix (a_1). Next, we compute the number of repetitions of each event at every position (Figure 3(b)). Then we refine the set of candidate events by the minimum repetitions (Figure 3(c)). The new set of prefixes are generated in Figure 3(d). Finally, we append eternal event(s) to each new prefix to generate a set of patterns with l events. Figure 3(e) shows the set of patterns that satisfy the minimum information gain threshold. The *InfoMiner* algorithm continues until l levels.

$a_1\ a_3\ a_4\ a_5 \mid a_1\ a_4\ a_3\ a_3 \mid a_1\ a_4\ a_3\ a_3 \mid a_1\ a_3\ a_3\ a_5 \mid a_1\ a_3\ a_4\ a_5 \mid a_1\ a_3\ a_4\ a_5$

(a) projected subsequence of prefix (a_1)

$(a_1, \ , \ , \ )$

Event	Repetition
a_3	3
a_4	1
a_6	0

Event	Repetition
a_3	2
a_4	2

Event	Repetition
a_2	0
a_3	1
a_5	3

(b) repetitions of each candidte in the projected subsequence

$(a_1, \ , \ , \ )$

Event	Repetition
a_3	3

Event	Repetition
a_3	2
a_4	2

Event	Repetition
a_5	3

(c) refined candidates by bounded information gain property where min_rep = 2

(a_1, a_3) $(a_1, *, a_3)$ $(a_1, *, a_4)$ $(a_1, *, *, a_5)$

(d) generated new prefixes

Pattern	Information Gain
$(a_1, a_3, *, *)$	5.19
$(a_1, *, *, a_5)$	6.36

(e) qualified pattern with min_gain = 4.5

Figure 3: Invocation of *InfoMiner* for Prefix (a_1)

5. EXPERIMENTAL RESULTS

We implemented *InfoMiner* in C programming language on an IBM RS-6000 (300 MHz CPU) with 128MB running AIX operating system. To analyze the benefits of the information model and the performance of the *InfoMiner* algorithm, we employ one real trace and four synthetic traces.

5.1 IBM Intranet Trace

The IBM Intranet traces consist of 160 critical nodes, e.g., file servers, routers, etc., in the IBM T. J. Watson Intranet. Each node issues a message in response of certain situation, e.g., CPU saturation, router interface down, etc. There are total 20 types of messages. We treat a certain message from a particular node as a distinct event, thus there are total 500 distinct events in the trace because a certain type of node may only send out 4 or 5 types of messages. The IBM Intranet trace consists of 10,000 occurrences of the events. By applying *InfoMiner* on this trace, we found some surprising patterns that are also interesting. For example, the pattern $(node_a\_fail, *, node_b\_saturated, *)$ has the eighth highest information gain. This pattern means that a short time after a router ($node_a$) fails, the CPU on another node ($node_b$) is saturated. Under a thorough investigation, we found that $node_b$ is a file server and after $node_a$ fails, all requests to some files are sent to $node_b$, thus causes the bottleneck. Further experimental results can be found in [14].

5.2 Synthetic Sequences

To analyze the performance of the *InfoMiner* algorithm, four sequences are synthetically generated. Each sequence consists of 1024 distinct events and 20M occurrences of events. The synthetic sequence is generated as follows. First, at the beginning of the sequence, the period length l of the next pattern is determined, which is geometrical distributed with mean μ_l. The number of events involved in a pattern is randomly chosen between 1 and l. The number of repetitions m of this pattern is geometrical distributed with mean μ_m. The events that are involved in the pattern are chosen according to a normal distribution. (This means that some events occurs more frequently than other.) However, the pattern may not perfectly repeat itself for m times. To simulate the imperfectness of the subsequence, we employ a parameter δ to control the noise. δ is uniformly distributed between 0.5 and 1. With probability δ, the next l events match the pattern. Otherwise, the next

l events do not support the pattern. The replacement events are chosen from the event set with the same normal distribution.This subsequence ends when there are m matches, and a new subsequence for a new pattern starts. This process repeats until it reaches the end of the sequence. Four sequences are generated based on values of μ_l and μ_m in Table 2.

Table 2: Parameters of Synthetic Data Sets

Data Set	μ_l	μ_m	Distinct events	Total Events
$l3m20$	3	20	1024	20M
$l100m20$	100	20	1024	20M
$l3m1000$	3	1000	1024	20M
$l100m1000$	100	1000	1024	20M

The main pruning power of the *InfoMiner* algorithm is provided by the bounded information gain pruning technique. In our *InfoMiner* algorithm, for each prefix, we prune the candidate events on each remaining open position. Although the number of candidate events on each open position can be $|E|$ theoretically, in practice the average number of candidates in each open position decreases dramatically with the increase of the number of specified positions (i.e., the length of the prefix). This is due to the fact that the value of max_info is estimated from the candidate event with the highest information on each open position. Thus, with more positions specified, the max_info value decreases. In turn, the min_rep threshold ($min\_rep = \lceil \frac{min\_gain}{max\_info} \rceil$) increases and more candidate events are pruned. We conduct experiments with our *InfoMiner* algorithm on the four synthetic sequences. Figure 4(a) shows the average number of remaining candidate events on each open position as a function of the number of specified positions (i.e., the length of the prefix). The number of candidate events decreases dramatically with the number of specified positions. With data set $l3m20$ and $l3m1000$, since the average pattern length is 3, there is no candidate after 5 or 6 positions are specified. In addition, with all four data sets, when the number of specified positions is greater than 3, the average number of events on each open position is very small (i.e., less than 0.4). This leads to the overall efficiency of the *InfoMiner* algorithm.

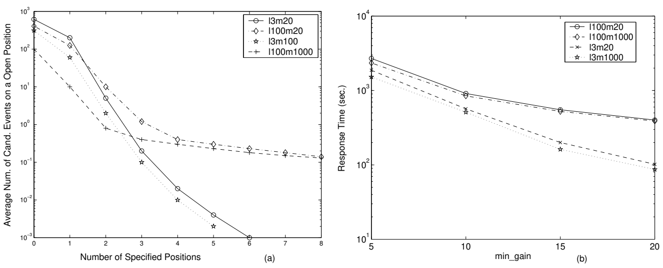

Figure 4: Pruning Effects and Response Time

The overall response time of the *InfoMiner* algorithm largely depends on the min_gain threshold. We ran several tests on the above data sets with different min_gain thresholds. Figure 4(b) shows the response time for each data set. Our bounded information gain pruning can reduce a large amount of patterns since more than 99% of the patterns are pruned. With the increase of the min_gain threshold, the pruning effects become more dominant because more patterns can be eliminated by the bounded information gain pruning. Thus, the response time improves with increasing min_gain threshold on all four data sets. (Note that the Y-axis is in log scale in Figure 4(a) and (b).) To explore the scalability of the *InfoMiner* algorithm, we also experiment with event sequences of different lengths, from 1 million to 100 million. We found that the response time of the *InfoMiner* is linear with respect to both the length of the event sequence and the period length.

6. CONCLUSION

In this paper, we study the problem of surprising periodic pattern discovery in a data sequence that consists of events with vastly different occurrence frequencies. Our goal is not to find the patterns that occur often, but rather to discover patterns that are surprising. Instead of using the support metric, we propose a new metric: information gain. Although the new metric does not possess the Apriori property as the support metric does, we identify the bounded information gain property in the information model. Based on the bounded information gain property, we devise a recursive algorithm for mining surprising periodic patterns by only exploring extensible prefixes.

7. REFERENCES

[1] G. Berger and A. Tuzhilin. Discovering unexpected patterns in temporal data using temporal logic. *Temporal Databases - Research and Practice, Lecture Notes on Computer Sciences*, (1399) 281-309, 1998.

[2] R. Blahut. *Principles and Practice of Information Theory*, Addison-Wesley Publishing Company, 1987.

[3] S. Brin, R. Motwani, C. Silverstein. Beyond market baskets: generalizing association rules to correlations. *Proc. ACM SIGMOD Conf. on Management of Data*, 265-276, 1997.

[4] J. Han, G. Dong, and Y. Yin. Efficient mining partial periodic patterns in time series database. *Proc. Int. Conf. on Data Engineering*, 106-115, 1999.

[5] M. Klemetinen, H. Mannila, P. Ronkainen, H. Toivonen, and A. Verkamo. Finding interesting rules from large sets of discovered association rules. *Proc. CIKM*, 1994.

[6] B. Liu, W. Hsu, and Y. Ma. Mining association Rules with multiple minimum supports. *Proc. ACM SIGKDD*, 337-341, 1999.

[7] S. Ma and J. Hellerstein. Mining partially periodic event patterns with unknown periods. *Proc. Int. Conf. on Data Engineering*, 205-214, 2001.

[8] H. Mannila, D. Pavlov, and P. Smyth. Prediction with local patterns using cross-entropy. *Proc. ACM SIGKDD*, 357-361, 1999.

[9] T. Oates, M. D. Schmill, P. R. Cohen. Efficient mining of statistical dependencies. *Proc. 16th Int. Joint Conf. on Artificial Intelligence*, 794-799, 1999.

[10] B. Padmanabhan and A. Tuzhilin. Small is beautiful: discovering the minimal set of unexpected patterns. *Proc. ACM KDD*, 54-63, 2000.

[11] A. Silberschatz and A. Tuzhilin. What makes patterns interesting in knowledge discover systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* vol. 8 no. 6, pp. 970-974, 1996.

[12] K. Wang, Y. He, and J. Han. Mining frequent itemsets using support constraints. *Proc. Int. Conf. on Very Large Data Bases*, 2000.

[13] J. Yang, W. Wang, and P. Yu. Mining asynchronous periodic patterns in time series data. *Proc. ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, pp. 275-279, 2000.

[14] J. Yang, W. Wang, and P. Yu. InfoMiner: mining surprising periodic patterns. *IBM Research Report*, 2001.

[15] M. J. Zaki. Generating non-redundant association rules. *Proc. ACM SIGKDD*, 34-43, 2000.