Meta-Patterns: Revealing Hidden Periodic Patterns

Wei Wang, Jiong Yang, and Philip S. Yu
IBM T. J. Watson Research Center
{ww1, jiyang, psyu}@us.ibm.com

Abstract

Discovery of periodic patterns in time series data has become an active research area with many applications. These patterns can be hierarchical in nature, where a higher level pattern may consist of repetitions of lower level patterns. Unfortunately, the presence of noise may prevent these higher level patterns from being recognized in the sense that two portions (of a data sequence) that support the same (high level) pattern may have different layouts of occurrences of basic symbols. There may not exist any common representation in terms of raw symbol combinations; and hence such (high level) pattern may not be expressed by any previous model (defined on raw symbols or symbol combinations) and would not be properly recognized by any existing method. In this paper, we propose a novel model, namely meta-pattern, to capture these high level patterns. As a more flexible model, the number of potential meta-patterns could be very large. A substantial difficulty lies on how to identify the proper pattern candidates. However, the well-known Apriori property is not able to provide sufficient pruning power. A new property, namely component location property, is identified and used to conduct the candidate generation so that an efficient **computation-based** *mining algorithm can be developed. Last but not least, we apply our algorithm to some real and synthetic sequences and some interesting patterns are discovered.*

1 Introduction

Periodicity detection on time series data is a challenging problem of great importance in many real applications. The periodicity is usually represented as repeated occurrences of a list of symbols in a certain order at some frequency [3, 6, 11, 12]. Due to the changes of system behavior, some pattern may be only notable within a portion of the entire data sequence and different patterns may present at different places and be of different durations. The evolution among patterns may also follow some regularity. Such regularity, if any, would be of great value in understanding the nature of the system and building prediction models. Consider the application of *inventory replenishment*. The history of inventory refill orders can be regarded as a symbol sequence. For brevity, let's only consider the replenishment of the flu medicine. Figure 1(a) shows the history of refill orders of a pharmacy during 1999 and 2000 on a weekly basis. The symbol "r" stands for a refill order of flu medicine was placed in the corresponding week while "-" represents that no flu medicine replenishment was made in that week. It is easy to see that the replenishment follows a biweekly pattern during the first half of each year and falls into a triweekly cycle during the second half of each year. This seasonal fluctuation also forms a high level periodic pattern (The period length is one year). However, such high level patterns may not be expressible by any previous model (defined in terms of raw symbols) due to the presence of noise, even when the noise is very limited. In the above example, a major outbreak of flu caused a provisional replenishment at the 4th week of 1999 (Figure 1(a)). Afterwards, even though the replenishment frequency is back to once every other week, the occurrences of all subsequent replenishments become misaligned. Even though the biweekly replenishment cycle was notable in the first two quarters of both 1999 and 2000, the corresponding portions in the data sequence have different layout of replenishment occurrences. This characteristic determines that the representation of the above two-level periodicity is beyond the expressive power of any traditional model of periodic patterns that only takes raw symbols as components. In any traditional model, each symbol specified in a pattern uniquely matches its counterpart in the data sequence, and all occurrences of a pattern have to share a unique common layout. This inherent limitation would prevent many interesting high level patterns from being captured. Note that, even if the period length (i.e., 52 weeks) is given[1], the only annual pattern that is able to be generated under the traditional model via pairwise comparison of symbols corresponding to each week is shown in Figure 1(b). The symbol "*" denotes the don't care position[2] and can match any symbol on the corresponding position. Clearly, little information is conveyed in this pattern as the important two-level periodicity is completely concealed.

To tackle the problem, we propose a so-called **meta-pattern** model to capture high level periodicities. A meta-pattern may take occurrences of patterns/meta-patterns (of lower granularity) as components. In contrast, we refer to the patterns that contain

[1] Note that in many applications, e.g., seismic periodicity analysis, the period length is usually unknown in advance and is part of the mining objective.

[2] It is introduced to represent the position(s) in a pattern where no strong periodicity exhibits.

only raw symbol(s) as the **basic patterns**, which may be viewed as special cases of meta-patterns. In general, the noise could occur anywhere, be of various duration, and even occur multiple times within the portion where a pattern is notable as long as the noise is below some threshold. Even though the allowance of noise plays a positive role in characterizing system behavior in a noisy environment, it prevents such a meta-pattern from being represented in the form of an (equivalent) basic pattern. The model of meta-pattern provides a more powerful means to periodicity representation. The recursive nature of meta-pattern not only can tolerate a greater degree of noises/distortion, but also can capture the (hidden) hierarchies of pattern evolutions, which may not be expressible by previous models. In the previous example, the biweekly and the triweekly replenishment cycles can be easily represented by $P_1 = (r : [1, 1], * : [2, 2])$ and $P_2 = (r : [1, 1], * : [2, 3])$, respectively, where the numbers in the brackets indicate the offset of the component within the pattern. The two-level periodicity can be easily represented as $(P_1 : [1, 24], * : [25, 25], P_2 : [26, 52])$ which can be interpreted as the pattern P_1 repeats at the first 24 weeks and the pattern P_2 repeats from week 26 to week 52. As shown in Figure 1 (c), each rectangle box denotes the portion where the corresponding low level pattern (i.e., either $(r : [1, 1], * : [2, 2])$ or $(r : [1, 1], * : [2, 3])$) is notable.

Unfortunately, the flexibility of meta-pattern poses serious challenges in the discovery process, which may not be encountered in mining basic patterns.

- While a basic pattern has two degrees of freedom: the period (i.e., the number of positions in the pattern) and the choice of symbol for each single position, a meta-pattern has an additional degree of freedom: the length of each component in the pattern. It is incurred by the fact that a component may occupy multiple positions. This extra degree of freedom would increase the number of potential meta-pattern candidates dramatically.
- Many patterns/meta-patterns may collocate or overlap for any given portion of a sequence. As a result, during the meta-pattern mining process, there could be a large number of candidates for each component of a (higher level) meta-pattern. This also aggravates the mining complexities.

Therefore, how to identify the "proper" candidate meta-patterns is very crucial to the overall efficiency of the mining process, and will be the focus of the algorithm part of the paper. To tackle this problem, we employ a so called *component location property*, in addition to the traditionally used Apriori property, to prune the search space. This is inspired by the observation that a pattern may participate in a meta-pattern only if its notable portions exhibit a certain cyclic behavior. A *computation-based* algorithm is devised to identify the potential period of a meta-pattern and for each candidate period, the potential components and their lengths within the meta-pattern. The set of all meta-patterns can be categorized according to their structures and are evaluated in a designed order so that the pruning power provided by both properties can be fully utilized.

In summary, we claim the following contributions in this paper.

- A model of meta-pattern is identified to capture the cyclic relationship among discovered periodic patterns and to enable a recursive construction of exhibited cyclic regularities.
- The *component location property* is proposed to provide further pruning power, in addition to the traditional Apriori property.
- A *computation-based* algorithm is designed to identify and to verify potential meta-pattern candidates.

The remainder of this paper is organized as follows. Section 2 gives a brief overview of recent related research. The general model is presented in Section 3. Section 4 outlines the major steps of our algorithm. Section 5 presents experimental results. The conclusion is drawn in Section 6.

2 Related Work

Most previous work on mining sequence data fell into two categories: discovering sequential patterns [1, 2, 4, 7, 8, 9, 10, 14] and mining periodic patterns[3, 6, 11, 12]. The primary difference between them is that the models of sequential pattern purely take into account the number of occurrences of the pattern while the frameworks for periodic patterns focus on characterizing cyclic behaviors. Due to space limitations, we will give a brief survey of the recent work on periodic patterns.

In [3], Han et. al. presented algorithms for efficiently mining partial periodic patterns. In practice, not every portion in the time series may contribute to the periodicity. For example, a company's stock may often gain a couple of points at the beginning of each trading session but it may not have much regularity at later time. This type of looser periodicity is often referred to as *partial periodicity*. We will see later that our model also allows partial periodicity.

To accommodate the phenomenon that the system behavior may change over time, a flexible model of asynchronous periodic pattern was proposed in [11]. In this model, a qualified pattern may be present only within a subsequence and whose occurrences may be shifted due to disturbance. Two parameters min_rep and max_dis are employed to specify the minimum number of repetitions that is required within each segment of non-disrupted pattern occurrences and the maximum allowed disturbance between any two successive valid segments. The intuition behind this is that a pattern needs to repeat itself at least a certain number of times to demonstrate its significance and periodicity. On the other hand, the disturbance between two valid segments has to be within some reasonable bound. Otherwise, it would be more appropriate to treat such disturbance as a signal of "change of system behavior" instead of random noise injected into some persistent behavior. The parameter

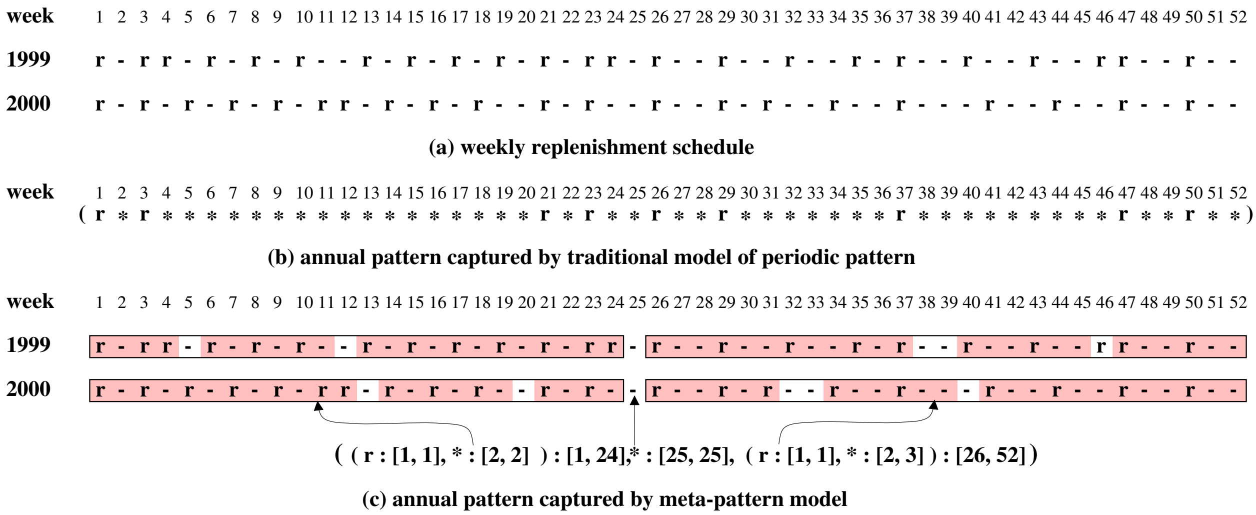

Figure 1. Meta Pattern

max_dis acts as the boundary to separate these two phenomena. For example, Figure 1(c) shows 6 valid segments of pattern $(r : [1,1], * : [2,2])$ (for both 1999 and 2000) if we set $min\_rep = 2$. (Each valid segment is indicated by a shaded region). Moreover, if we set $max\_dis = 3$, the three valid segments in 1999 (separated by a disturbance of 1 symbol) form a valid symbol subsequence of $(r : [1,1], * : [2,2])$ while the three valid segments 2000 form another valid symbol subsequence. Obviously, the appropriate values of these two parameters are application dependent and need to be specified by the user. Note that, due to the presence of disturbance, some subsequent valid segment may not be well synchronized with the previous ones. (Some position shifting occurs.) Upon satisfying these two requirements, the longest valid subsequence of a pattern is returned. A two phase algorithm is devised to first generate potential periods by distance-based pruning followed by an iterative procedure to derive and validate candidate patterns and locate the longest valid subsequence. Still, this paper only concerns patterns constructed from raw symbols and does not address the problem of meta-patterns. As we pointed out earlier, due to the extra degree of freedom possessed by meta-pattern and the massive amount of potential candidate components, no direct generalization of existing algorithm (designed for mining patterns consisting of only raw symbols) can be applied to mine meta-pattern efficiently. In fact, serious challenges exist in how to quickly identify the "proper" candidate meta-patterns so that unnecessary computation can be avoided. Thus, we will focus on candidate meta-pattern generation (rather than the candidate meta-pattern validation) in this paper.

Many patterns may occur frequently by nature and they are of no importance in many applications. In [12], instead of frequently occurred periodic patterns, the statistically significant patterns are mined. There is an occurrence probability associated with each pattern. If a pattern occurrence much more than the expectation, then this pattern is called statistically significant or surprising. An efficient algorithm is also proposed in [12] for mining such patterns.

3 Model of Meta-Patterns

Let $\Im = \{a, b, c, \ldots\}$ be a set of literals. A traditional periodic pattern [3, 11] consists of a tuple of k components, each of which is either a literal or "*". k is usually referred to as the **period** of the pattern. "*" can be substituted for any literal and is used to enable the representation of partial periodicity. For example, $\ldots, a, b, c, a, b, d, a, b, b, \ldots$ is a sequence of literals and $(a, b, *)$ [3] represents that the incident "b following a" occurs for every 3 time instances in the data sequence. The period of this pattern is 3 by definition. Note that the third component in the pattern is filled by a "*" since there is no strong periodicity presents in the data sequence with respect to this component. Because a pattern may start anywhere in the data sequence, only patterns whose first component is a literal in $\Im$ need to be considered. In this paper, we refer to this type of patterns as **basic** patterns as each component in the pattern is restricted to be either a literal or a "*". In contrast, a **meta-pattern** may have pattern(s)/meta-pattern(s) as its component(s). This enables us to represent complicated basic patterns in a more concise way and to possibly reveal some hidden patterns among discovered ones. Formally, a **meta-pattern** is a tuple consisting of k components $(x_1, x_2, \ldots, x_k)$ where each x_i $(1 \leq i \leq k)$ can be one of the following choices augmented by the offsets of the starting and ending positions of the component with respect to the beginning of the meta-pattern: (1) a symbol in $\Im$; (2) "don't care" *; and (3) a pattern/meta-pattern. We also require that at least one position of a meta-pattern has to correspond to a non "*" component to ensure a non-trivial pattern. For example, $((r : [1,1], * : [2,2]) : [1,24], * : [25,25], (r : [1,1], * : [2,3]) : [26,52])$ is

[3] Since each component corresponds to exactly one symbol, we do not have to explicitly record the offset of a component within the pattern as this information can be easily derived.

a meta-pattern with three components: $(r:[1,1],*:[2,2])$, *, and $(r:[1,1],*:[2,3])$. The **length** of a component is the number of positions that the component occupies in the meta-pattern. In the previous example, the component length of $(r:[1,1],*:[2,2])$ is 24. We also say that 52 is the **span** of this meta-pattern, which is equal to the sum of the length of all components in the meta-pattern. This pattern can be interpreted as "*the pattern* $(r:[1,1],*:[2,2])$ *is true for 24 positions (or weeks in previous example) followed by the pattern* $(r:[1,1],*:[2,3])$ *for 27 positions with a gap of one position in between, and such a behavior repeats for every 52 positions*". For brevity, we sometimes omit the augmenting offset of a component if the length of the component is only one position. For example, $(r,*)$ is the abbreviation of $(r:[1,1],*:[2,2])$ and $((r,*):[1,24],*,(r,*:[2,3]):[26,52])$ is equivalent to $((r:[1,1],*:[2,2]):[1,24],*:[25,25],(r:[1,1],*:[2,3]):[26,52])$. It is obvious that the meta-pattern is a more flexible model than the basic pattern and the basic pattern can be viewed as a special (and simpler) case of the meta-pattern. Because of the hierarchical nature of the meta-pattern, the concept of *level* is introduced to represent the "depth" of a meta-pattern. By setting the level of basic pattern to be 1, the **level** of a meta-pattern is defined as the maximum level of its components plus 1. According to this definition, the level of $(r,*:[2,3])$ is 1 and the level of $P_1=((r,*):[1,24],*,(r,*:[2,3]):[26,52])$ is 2. Note that the components of a meta-pattern do not have to be of the same level. For instance, $(P_1:[1,260],*:[261,300])$ is a meta-pattern (of level 3) which has a level-2 component and a level-1 component.

All terminologies associated with the basic patterns [11] (i.e., level-1 patterns) can be generalized to the case of meta-patterns (i.e., higher level patterns). We now give a brief overview of terms defined in [11] for basic patterns. Given a symbol sequence $D'=d_1,d_2,\ldots,d_s$ and a basic pattern $P=(p_1,p_2,\ldots,p_s)$, we say D' **supports** P iff, for each $i(1\le i\le s)$, either $p_i=*$ or $p_i=d_i$. D' is also called a **match** of P. Given a pattern P and a symbol sequence D, a list of j disjoint matches of P in D is called a **segment** with respect to P iff they form a contiguous portion of D. j is referred to as the **number of repetitions** of this segment. Such a segment is said to be a **valid segment** iff j is greater than or equal to the required minimum repetition threshold min_rep. A **valid subsequence** in D (with respect to P) is a set of disjoint valid segments where the distance between any two consecutive valid segments does not exceed the required maximum disturbance threshold max_dis. P is said to be a **valid pattern** in D if there exists a valid subsequence in D with respect to P. The parameters min_rep and max_dis, in essence, define the significance of the periodicity and the boundary to separate noise and change of system behavior. The appropriate values of min_rep and max_dis are application dependent and are specified by the user.

Similarly, given a symbol sequence $D'=d_1,d_2,\ldots,d_s$, for any meta-pattern $X=(x_1:[1,t_1],x_2:[t_1+1,t_2],\ldots,x_l:[t_{l-1}+1,s])$, D' **supports** X iff, for each component x_i, either (1) x_i is "*" or (2) x_i is a symbol and $d_{t_{i-1}+1}=\ldots=d_{t_i}=x_i$ or (3) x_i is a (mata-)pattern and $d_{t_{i-1}+1},\ldots,d_{t_i}$ is a valid subsequence with respect to x_i. D' is in turn called a **match** of P. We can define *segment*, *subsequence*, and *validation* in a similar manner to that of a basic pattern. Given a meta-pattern X and a symbol sequence D, a list of j disjoint matches of X in D is called a **segment** with respect to X iff they form a contiguous portion of D. j is referred to as the **number of repetitions** of this segment. Such segment is said to be a **valid segment** iff j is greater than or equal to the required minimum repetitions min_rep. A **valid subsequence** in D (with respect to X) is a set of disjoint valid segments where the distance between any two consecutive valid segments does not exceed the required maximum disturbance max_dis. P is said to be a **valid pattern** in D if there exists a valid subsequence in D with respect to X.

In this paper, given a symbol sequence and two parameters min_rep and max_dis, we aim at mining valid meta-patterns together with their longest valid subsequences (i.e., the valid subsequence which has the most overall repetitions of the corresponding meta-pattern). Since a meta-pattern can start anywhere in a sequence, we only need to consider those starting with a non "*" component.

4 Algorithm Outline

The great flexibility of the model poses considerable difficulties to the generation of candidate meta-patterns. Therefore, we will focus on the efficient candidate generation of meta-patterns in the remainder of this paper. The well-known Apriori property holds on the set of meta-patterns of the same span, which can be stated as follows: *for any valid meta-pattern* $P=(P_1:[1,t_1],P_2:[t_1+1,t_2],\ldots,P_s:[t_{s-1}+1,t_s])$*, the meta-pattern constructed by replacing any component* P_i *with "*" in* P *is also valid*. For example, let $X_1=((a,b,*):[1,19],*:[20,21])$ and $X_2=((a,b,*):[1,19],*:[20,21],(b,c):[22,27],*:[28,30],X_1:[31,150])$. If X_2 is a valid meta-pattern, then the pattern $X_3=((a,b,*):[1,19],*:[20,21],(b,c):[22,27],*:[28,150])$ (generated by replacing X_1 with "*") must be valid as well. Note that X_2 is a level-3 meta-pattern which has three non "*" components: $(a,b,*)$, (b,c), and X_1; whereas X_3 is a level-2 meta-pattern that has two non "*" components: $(a,b,*)$ and (b,c). Intuitively, X_3 should be examined before X_2 so that the result can be used to prune the search space.

Nevertheless, because of the hierarchical characteristic of the meta-pattern, the Apriori property does not render sufficient pruning power as we proceed to high level patterns from discovered low level patterns. After identifying valid meta-patterns of level l, the brute force method (powered by the Apriori property) to mine patterns of level $l+1$ is to first generate all possible candidates of level $l+1$ by taking valid lower level patterns as component(s); and then, verify them against the symbol sequence. While the verification of a base pattern can be performed efficiently (e.g., in linear time with respect to the length of the symbol sequence [11]), the verification for a candidate meta-pattern may be a cumbersome process because of the typically complicated structure of the candidate meta-pattern. In fact, considerable difficulty lies on determining whether a certain portion of the raw symbol sequence corresponds to a valid subsequence of a component of the candidate pattern, especially when the component itself is also meta-pattern. One strategy to speed up the process is to store all valid subsequences of each valid low level pattern when the pattern is verified. Then the procedure of determining whether a portion of the sequence is a valid subsequence of a given component can be accomplished via table look-up operations. Even though this strategy requires additional storage space, it can usually lead to at least an order of magnitude of performance improvement. We will refer this method as the **match-based approach** in the remainder of this paper.

However, this match-based approach is still very cumbersome, and more specifically, suffers from two major drawbacks. (1) The number of candidate patterns of a certain level (say level l) is typically an exponential function of the number of discovered lower level meta-patterns. While a basic pattern has two degrees of freedom: the period and the choice of symbol at each position/component, a meta-pattern has an additional degree of freedom: the length of each component. This additional degree of freedom dramatically increases the number of candidate patterns generated. If there are v valid lower level patterns, the number of candidate patterns of span s and with exactly k components for level l is in the order of $\Theta(v^k \times (2k)^s)$. (2) There are typically a huge number of valid subsequences associated with each valid pattern even though only a few of them may eventually be relevant. Generating and storing all of them would consume a significant amount of computing resources and storage space, which in turn leads to unnecessary inefficiency. To overcome these drawbacks, we made the following observation.

Property 4.1 (Component Location Property) *A valid low level meta-pattern may serve as a component of a higher level meta-pattern only if its presence in the symbol sequence exhibits some cyclic behavior and such cyclic behavior has to follow the same periodicity as the higher level meta-pattern by sufficient number of times (i.e., at least min_rep times).*

In the above example, the meta-pattern X_1 can serve as a component of a higher level meta-pattern (e.g., X_2) only if the locations of valid subsequences of X_1 exhibits a cyclic behavior with a period equal to the span of X_2 (i.e., 150). Otherwise, X_1 could not serve as a component of X_2. This property suggests that we can avoid the generation of a huge number of unnecessary candidate meta-patterns by deriving candidates from qualified span-component combinations according to the component location property. To identify qualified span-component combinations, we need to detect the periodicities exhibited by the locations of valid subsequences of each low level meta-pattern. This can be achieved without generating all valid subsequences for a meta-pattern. In fact, only the set of **maximum valid segments** are sufficient. For a given pattern, a valid segment is a maximum valid segment if it is not a portion of another valid segment. For example, if $min\_rep = 3$ and $max\_dis = 6$, $\{S_1, S_2, S_3, S_4, S_5$ $S_6\}$ is the set of maximum valid segments of basic pattern $(a, *)$ for the symbol sequence in Figure 2(a). Usually, the number of maximum valid segments is much smaller than the number of valid subsequences. The total number of distinct valid subsequences of $(a, *)$ in the symbol sequence given in Figure 2(a) would be in the order of hundreds. It is in essence an exponential function of the number of maximum valid segments. Furthermore, for each maximum valid segment, we only need to store a pair of location indexes indicating its starting and ending positions. In the above example, the segment S_1 occupies 8 positions (positions 1 to 8) in Figure 2(a) and its location indexes is the pair $(1, 8)$. The location indexes of maximum valid segments indeed provide a compact and ingenious representation of all necessary knowledge of a valid low level meta-pattern and is motivated by the following observations.

- Given the set of location indexes of maximum valid segments of a pattern, it is easy to compute all possible starting positions and ending positions of valid subsequences. Any starting position of a valid segment is also a starting position of a valid subsequence because a valid subsequence is essentially a list of valid segments. Given a maximum valid segment S containing r repetitions of the pattern, there are $r - min\_rep + 1$ distinct starting positions that can be derived from S. More specifically, they are the positions of the first $r - min\_rep + 1$ occurrences of the pattern in S, respectively. For instance, positions 1 and 3 are the two starting positions derived from S_1. Similarly, all possible ending positions can be computed as well.
- The starting positions of the valid subsequences that exhibit cyclic behavior also present the same periodicity and so do their ending positions. Figure 2(b) shows the set of possible starting positions and ending positions of valid subsequences of $(a, *)$. When $min\_rep = 3$, by careful examination, the potential periodicities of $(a, *)$ (i.e., the possible spans of meta-patterns that $(a, *)$ may participate in as a component) include 7, 9, 11, 18, and 20. The periodic behavior discovered on starting positions and ending positions for span = 18 is shown in Figure 2(c) and (d), respectively.

Thus, our strategy is to first compute the set of possible starting positions and identify, if any, the ones that exhibit some periodicity. The same procedure is also performed on the ending positions. If the same periodicity exists for both starting positions and ending positions, we then examine, for each pair of starting and ending positions, whether a valid subsequence exists and what is the possible format of the higher level meta-pattern (i.e., possible span of the meta-pattern and possible length of the component). Figure 2(e) shows some candidate components generated from $(a, *)$ and the valid subsequences that support them.

It is important to notice that the maintenance of the location indexes of maximum valid segments leads to a double-win situation. Besides its positive role in candidate generation, it also

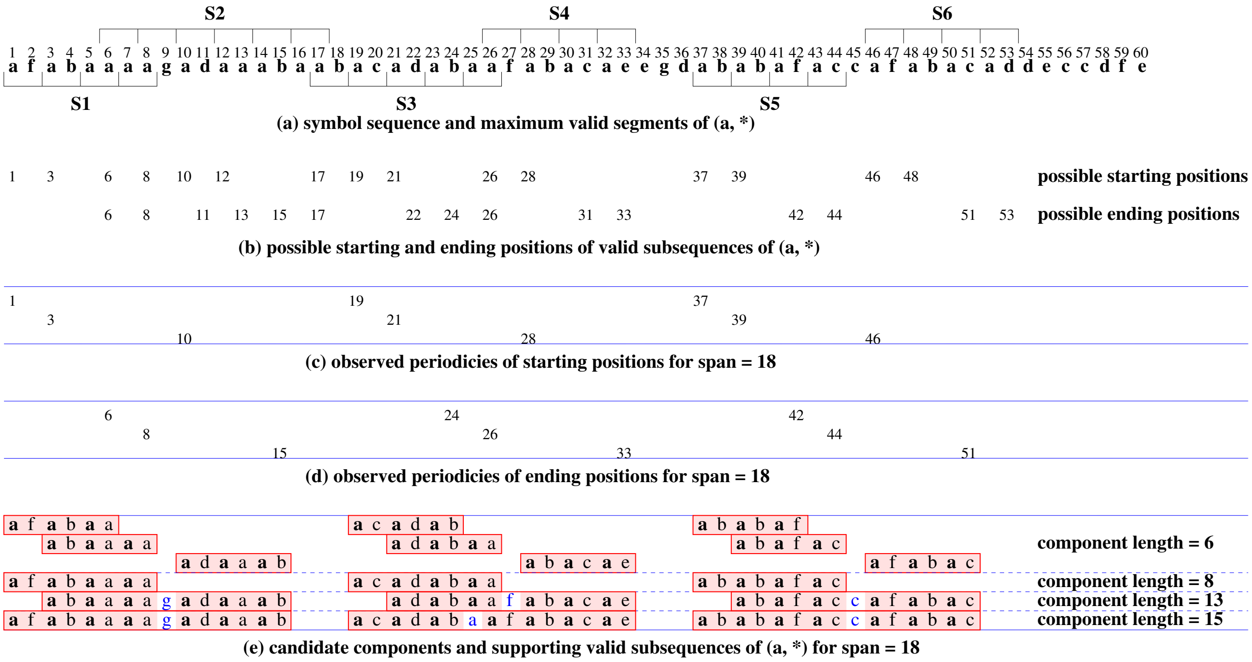

Figure 2. Computation Based Approach

enables the verification process to be accomplished efficiently without the expensive generation and maintenance of all valid subsequences nor the necessity of resort to the raw symbol sequence. As a result, we devise an efficient **computation-based** algorithm (as opposite to the traditional match-based approach) in the sense that the discovery of valid meta-patterns (other than base patterns) can be accomplished through pure computation (performed on the location indexes of maximum valid segments) without ever resort back to the raw symbol sequence. It has been demonstrated that this advantage offers at least two orders of magnitudes speed-up comparing to the match-based approach.

The component location property can provide substantial inter-level pruning effect during the generation of high level candidates from valid low level meta-patterns; whereas the traditional Apriori property can render some pruning power to conduct the mining process of meta-patterns of the same level. While all meta-patterns can be categorized according to their levels and the number of non "*" components in the pattern as shown in Figure 3, the pruning effects provided by the component location property and the Apriori property are indicated by dashed arrows and solid arrows, respectively. Consequently, the algorithm consists two level of iterations. The outer iteration exploits the component location property while the inner iteration utilizes the Apriori property. More specifically, each outer iteration discovers all meta-patterns of a certain level (say, level l) and consists of the following two phases.

1. *candidate component generation*. For each newly discovered valid pattern/meta-pattern of level l, generate candidate components for meta-patterns of level $l+1$. The component location property is employed in this phase.
2. *candidate pattern generation and verification*. This phase generates candidate meta-patterns of level $l+1$ based on the candidate components discovered in the previous step and validates them. This phase utilizes the Apriori property and contains an iteration loop. During each iteration, meta-patterns with a certain number (say k) of non "*" components are examined, which includes the following two steps.
   (a) If $k = 1$, the candidate singular meta-patterns[4] of level $l+1$ are generated from candidate components derived in the previous phase. Otherwise, the candidate meta-patterns of level $l+1$ with k non "*" components are generated based on the discovered level-$(l+1)$ meta-patterns with $(k-1)$ non "*" components.
   (b) The newly generated candidate patterns are validated.

   This inner iteration continues until no new candidate patterns of level $(l+1)$ can be generated.

The entire procedure terminates when no new candidate components can be generated. Due to space limitations, we would not elaborate on the details of the algorithm. Interested readers please refer to [13].

5 Experimental Results

The meta-pattern discovery algorithm is implemented in C on an AIX workstation with 300 MHz CPU and 128 MB main

[4] A singular meta-pattern is a meta-pattern that has only one non "*" component. Otherwise, it is called a complex meta-pattern.

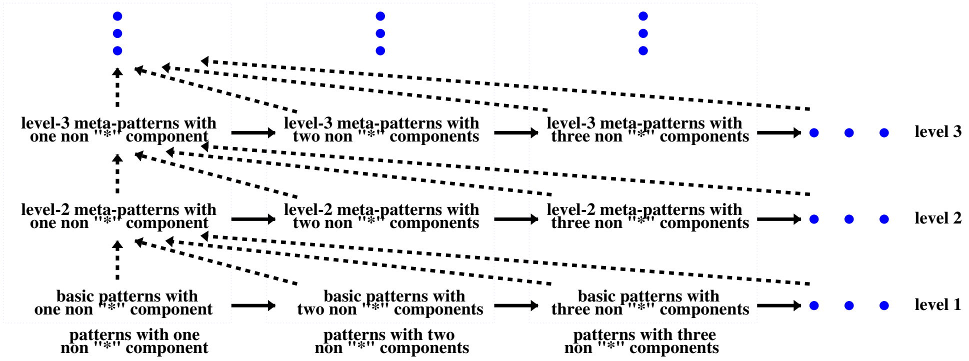

Figure 3. Pruning Directions

memory. A real trace log from the search engine *scour.net* is employed to evaluate the benefits of the meta-pattern model while four synthetically generated sequences are used to measure the performance of our algorithm.

5.1 Scour Traces

Scour is a web search engine specialized in multimedia content search whose URL is "http://www.scour.net". We apply the meta-pattern discovery algorithm to the traces of scour traces. Table 1 shows the number of patterns discovered from the Scour sequence with respective thresholds. There exist some interesting patterns. When min_rep and max_dis are set to 3 and 200, respectively, there is a level 3 meta-pattern. This level 3 pattern describes the following phenomenon. In a weekday between 4am and 12pm EST, there exists a pattern that the number of hits is between 10000 to 20000; and during 5pm to 1:30am EST, we found the pattern that the number of hits is between 40000 and 49999. Furthermore, this pattern repeated itself during each weekday within a week (i.e., level-2 pattern) and it also exhibits weekly trend (i.e., level-3 pattern). In addition, we also compute the meta-patterns that cannot be expressed in the form of basic patterns. We call these patterns *meta only* patterns and the number of these patterns is also shown in Table 1. From this table, we can see that most of the discovered level 2 and 3 patterns can not be expressed in the form of basic patterns, and thus can only be represented as meta-patterns.

To further understand the behavior of our proposed meta-pattern mining algorithm, we constructed four long synthetic sequences and the sensitive analysis of our algorithm on these sequences is presented in the following section.

5.2 Synthetic Sequences

The four synthetic sequences are generated as follows. Each sequence consists of 1024 distinct symbols and 20M occurrences of symbols. The synthetic sequence is generated as follows. First, at the beginning of the sequence, the level of pattern is determined randomly. There are four possible levels, i.e., 1, 2, and 3, 4. Next, the number of segments in this pattern is determined. The length l of each segment is selected based on a geometric distribution with mean μ_l. The number of repetitions of a lower level pattern in a segment is randomly chosen between min_rep and $\lfloor \frac{l}{p} \rfloor$ where p is the span of the lower level pattern. The number of symbols involved in a pattern is randomly chosen between 1 and the span p. The number of valid segments is chosen according to a geometrical distribution with mean μ_s. After each valid segment, the length of the disturbance is determined based on a geometrical distribution with mean μ_d. This process repeats until the length of the sequence reaches 20M. Four sequences are generated based on values of μ_l, μ_s, μ_r, and μ_d in Table 2.

Data Set	μ_l	μ_s	μ_r	μ_d
$DS1$	5	5	50	50
$DS2$	5	5	1000	1000
$DS3$	100	1000	50	50
$DS4$	100	1000	1000	1000

Table 2. Parameters of Synthetic Data Sets

In our proposed algorithm, we use the component location property pruning to reduce the candidate patterns. Figure 4(a) shows the effects of the component location property pruning method. The pruning effect is represented as a ratio of the candidate patterns (after pruning) and the total number of potential patterns at each level. We compute the pruning effects of each data set individually and the average is shown in the figure. It is evident that the pruning effects increase with the min_rep threshold because more patterns would be disqualified. In addition, in the higher level, the pruning effects are more evident because more constraints are involved.

In our approach, we only store the maximum valid segments for each pattern. We compare the computation-based approach with the match-based approach. In the match-based approach, for each pattern, all valid subsequences are stored and used to mine higher level meta-patterns. For each level of pattern, we track the CPU time consumed by the computation-based approach and the match-based approach. (We assume that all information can be stored in main memory. Since the number of possible subsequences is much larger than that of maximum valid segments, the match-based approach has more ad-

min_rep	max_dis	Level-1 Patterns	Level-2 Patterns	Level-3 Patterns	Meta only Patterns
3	200	107	12	1	10
10	200	31	5	0	5
20	200	15	2	0	2

Table 1. Patterns Discovered in Scour Trace

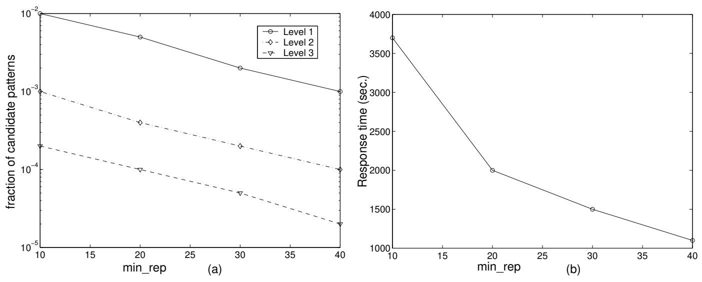

Figure 4. Performance of Meta-pattern Algorithm

vantages with this assumption.) The ratio of the CPU time of the computation-based approach over that of the match-based approach is calculated and presented in Table 3. It is obvious that the computation-based approach can save at least 95% of the CPU time comparing to the match-based approach. This is because the number of maximum valid segments is far less than that of valid subsequences as we explained in Section 5.

Level	$DS1$	$DS2$	$DS3$	$DS4$
2	0.02	0.02	0.04	0.01
3	0.05	0.03	0.01	0.02

Table 3. CPU Time Ratio

The overall response time is one of the most important criteria for evaluation of an algorithm. We mine the meta-patterns with different min_rep threshold. For a given min_rep, we mine the patterns on all four data sets and the average response time over the four data sets are taken and shown in Figure 4(b). The average response time decreases exponentially as min_rep increases. Although the average response time is a bit long when min_rep is small, but it is still tolerable (around 1 hour) due to the pruning effects of component location property and the computation-based candidate pattern generation. The meta-pattern mining algorithm is also applied to symbol sequences with different lengths. We found that the response time of our algorithm is linearly proportional to the length of the symbol sequence.

6 Conclusion

Meta-pattern is proposed to capture the hierarchical cyclic behavior exhibited in data sequence. A meta-pattern itself can serve as a component of some higher level meta-patterns. To accommodate noises in the data sequence, two parameters min_rep and max_dis are used to qualify a subsequence. The number of candidates of meta-patterns could be very large. To minimize the response time of the pattern mining process, a pruning algorithm based on the component location property and Apriori property is proposed which can greatly to reduce the number of candidate patterns. In addition, a computation-based algorithm is designed to identify potential meta-pattern candidates. We use the meta-pattern mining algorithm on some real traces and some very interesting results are discovered.

References

[1] R. Agrawal and R. Srikant. Mining Sequential Patterns. *Proc. ICDE*, 3-14, 1995.

[2] M. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: sequential pattern mining with regular expression constraints. *Proc. VLDB*, 223-234, 1999.

[3] J. Han, G. Dong, and Y. Yin. Efficient mining partial periodic patterns in time series database. *Proc. ICDE*, 106-115, 1999.

[4] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu. FreeSpan: frequent pattern-projected sequential pattern mining. *Proc. SIGKDD*, 2000.

[5] L. Lin and T. Risch. Querying continuous time sequences. *Proc. VLDB*, 170-181, 1998.

[6] S. Ma and J. Hellerstein. Mining partially periodic event patterns with unknown periods. *Proc. ICDE*, 205-214, 2001.

[7] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, vol. 1, no. 3, 259-289, 1997.

[8] B. Padmanabhan and A. Tuzhilin. Pattern discovery in temporal databases: a temporal logic approach. *Proc. SIGKDD*, 351-354, 1996.

[9] R. Srikant and R. Agrawal. Mining sequential patterns: generalizations and performance improvements. *Proc. EDBT*, 3-17, 1996.

[10] W. Wang, J. Yang, and R. Muntz. TAR: temporal association rules on evolving numerical attributes. *Proc. ICDE*, 283-292, 2001.

[11] J. Yang, W. Wang, and P. Yu. Mining asynchronous periodic patterns in time series data. *Proc. SIGKDD*, 275-279, 2000.

[12] J. Yang, W. Wang, and P. Yu. Mining surprising periodic patterns. To appear in *Proc. SIGKDD*, 2001.

[13] J. Yang, W. Wang, and P. Yu. Meta-Patterns: revealing hidden periodic patterns. *IBM Research Report*, 2001.

[14] M. Zaki. Sequence mining in categorical domains: incorporating constraints. *Proc. CIKM*, 422-429, 2000.