Using Convolution to Mine Obscure Periodic Patterns in One Pass [*]

Mohamed G. Elfeky   Walid G. Aref   Ahmed K. Elmagarmid

Department of Computer Sciences, Purdue University
{mgelfeky, aref, ake}@cs.purdue.edu

Abstract

The mining of periodic patterns in time series databases is an interesting data mining problem that can be envisioned as a tool for forecasting and predicting the future behavior of time series data. Existing periodic patterns mining algorithms either assume that the periodic rate (or simply the period) is user-specified, or try to detect potential values for the period in a separate phase. The former assumption is a considerable disadvantage, especially in time series databases where the period is not known a priori. The latter approach results in a multi-pass algorithm, which on the other hand is to be avoided in online environments (e.g., data streams). In this paper, we develop an algorithm that mines periodic patterns in time series databases with unknown or obscure periods such that discovering the period is part of the mining process. Based on convolution, our algorithm requires only one pass over a time series of length n, with $O(n \log n)$ time complexity.

1 Introduction

A time series database is one that abounds with data evolving over time. Life embraces several examples of time series databases such as meteorological data containing several measurements, e.g., temperature and humidity, stock prices depicted in financial market, and power consumption data reported in energy corporations. Data mining is the process of discovering patterns and trends by sifting through large amounts of data using technology that employs statistical and mathematical techniques.

Research in time series data mining has concentrated on discovering different types of patterns: sequential patterns [3, 18, 10, 5], temporal patterns [7], periodic association rules [17], partial periodic patterns [12, 11, 4], surprising patterns [14] to name a few. These periodicity mining techniques require the user to specify a period that determines the rate at which the time series is periodic. Hence, they assume that users either know the value of the period beforehand or that they are willing to try various period values until satisfactory periodic patterns emerge. Since the mining process must be executed repeatedly to obtain good results, this trial-and-error scheme is clearly not efficient. Even in the case of time series data with a priori known periods, there may be obscure periods, and consequently interesting periodic patterns that will not be discovered. The solution to these problems is to devise techniques for discovering potential periods in time series data. Research in this direction has focused either on devising general techniques for discovering potential periods [13, 6], or on devising special

[*]This work has been supported in part by the National Science Foundation under grants IIS-0093116, EIA-9972883, IIS-0209120, and by grants from NCR and Wal-Mart.

techniques for specific periodicity mining problems [20, 16]. Both approaches turn out to require multiple passes over the time series in order to output the periodic patterns themselves. However, real-time systems, which draw the attention of database researchers recently (e.g., as in data streams), cannot abide the time nor the storage needed for multiple passes over the data.

In this paper, we address the problem of mining periodic patterns in time series databases of unknown or obscure periods, hereafter referred to as *obscure periodic patterns*. We define the periodicity of the time series in terms of its symbols, and subsequently define the obscure periodic patterns where the period is a variable rather than an input parameter (Section 2). We develop a convolution-based algorithm for mining the obscure periodic patterns in one pass (Section 3). To the best of our knowledge, our proposed algorithm is the first algorithm in the literature (Section 1.1) that mines the periodic patterns with unknown period in one pass. In Section 4, the performance of our proposed algorithm is extensively studied verifying its correctness, examining its resilience to noise, and justifying its practicality. We summarize our findings in Section 5.

1.1 Related Work

Discovering the period of time series data has drawn the attention of the data mining research community very recently. Indyk et al. [13] have addressed this problem under the name *periodic trends*, and have developed an $O(n \log^2 n)$ time algorithm, where n is the length of the time series. Their notion of a periodic trend is the relaxed period of the entire time series, and their output is a set of candidate period values. In order to output the periodic patterns of the time series, a periodic patterns mining algorithm should be incorporated using each candidate period value, resulting in a multi-pass periodicity mining process.

Specific to partial periodic patterns, Ma and Hellerstein [16] have developed a linear distance-based algorithm for discovering the potential periods regarding the symbols of the time series. However, their algorithm misses some valid periods since it only considers the adjacent inter-arrivals. For example, consider a symbol that occurs in a time series in positions 0, 4, 5, 7, and 10. Although the underlying period should be 5, the algorithm only considers the periods 4, 1, 2, and 3. Should it be extended to include all possible inter-arrivals, the complexity of the algorithm of [16] will increase to $O(n^2)$. In [20], a similar algorithm has been proposed with some pruning techniques. Yet, both algorithms of [20, 16] require at least two passes over the time series in order to output the periodic patterns.

Berberidis et al. [6] have solved the problem of the distance-based algorithms by developing a multi-pass algorithm for discovering the potential periods regarding the symbols of the time series, one symbol at a time. Their algorithm suffers from the need for incorporating a periodic patterns mining algorithm to output the periodic patterns of the time series.

2 Problem Definition

2.1 Notation

Assume that a sequence of n timestamped feature values is collected in a time series database. For a given feature t, let t_i be the value of the feature at timestamp i. The time series of feature t is represented as $T = t_0, t_1, \ldots, t_{n-1}$. For example, the feature in a time series database for power consumption might be the hourly power consumption rate of a certain customer, while the feature in a time series database for stock prices might be the final daily stock price of a specific company. If we discretize [14] the time series feature values into nominal discrete levels and denote each level (e.g., high, medium, low, etc.) by a symbol (e.g., a, b, c, etc.), then the set of collected feature values can be denoted $\Sigma = \{a, b, c, \cdots\}$, where T is a string of length n over Σ.

A time series database may also be a sequence of n timestamped events drawn from a finite set of nominal event types, e.g., the event log in a computer network monitoring the various events that can occur. Each event type can be denoted by a symbol (e.g., a, b, c, etc.), and hence we can use the same notation above.

2.2 Symbol Periodicity

In a time series T, a symbol s is said to be periodic with a period p if s exists "almost" every p timestamps. For example, in the time series $T = abcabbabcb$, the symbol b is periodic with period 4 since b exists every four timestamps (in positions 1, 5 and 9). Moreover, the symbol a is periodic with period 3 since a exists almost every three timestamps (in positions 0, 3, and 6 but not 9). We define symbol periodicity as follows.

Let $\pi_{p,l}(T)$ denote the projection of a time series T according to a period p starting from position l; that is

$$\pi_{p,l}(T) = t_l, t_{l+p}, t_{l+2p}, \ldots, t_{l+(m-1)p},$$

where $l < p$, $m = \lceil (n-l)/p \rceil$, and n is the length of T. For example, if $T = abcabbabcb$, then $\pi_{4,1}(T) = bbb$, and $\pi_{3,0}(T) = aaab$. Intuitively, the ratio of the number of occurrences of a symbol s in a certain projection $\pi_{p,l}(T)$ to the length of this projection indicates how often this symbol occurs every p timestamps. However, this ratio is not quite accurate since it captures all the occurrences even the outliers. In the example above, the symbol b will be considered periodic with period 3 with a frequency of $1/4$, which is not quite true. As another example, if for a certain T, $\pi_{p,l}(T) = abcbac$, this means that the symbol changes every p timestamp and so no symbol should be periodic with a period p. We remedy this problem by considering only the consecutive occurrences. A consecutive occurrence of a symbol s in a certain projection $\pi_{p,l}(T)$ indicates that the symbol s reappeared in T after p timestamps from the previous appearance, which means that p is a potential period for s. Let $\mathcal{F}_2(s, T)$ denote the number of times the symbol s occurs in two consecutive positions in the time series T. For example, if $T = abbaaabaa$, then $\mathcal{F}_2(a, T) = 3$ and $\mathcal{F}_2(b, T) = 1$.

Definition 1 *If a time series T of length n contains a symbol s such that $\exists l, p$ where $l < p$, and $\frac{\mathcal{F}_2(s, \pi_{p,l}(T))}{\lceil (n-l)/p \rceil - 1} \geq \psi$ where $0 < \psi \leq 1$; then s is said to be periodic in T with period p at position l with respect to a periodicity threshold equal to ψ.*

For example, in the time series $T = abcabbabcb$, $\frac{\mathcal{F}_2(a, \pi_{3,0}(T))}{\lceil 10/3 \rceil - 1} = 2/3$, thus the symbol a is periodic with period 3 at position 0 with respect to a periodicity threshold $\psi \leq 2/3$. Similarly, the symbol b is periodic with period 3 at position 1 with respect to a periodicity threshold $\psi \leq 1$.

2.3 Obscure Periodic Patterns

The main advantage of the definition of symbol periodicity is that not only does it determine the candidate periodic symbols, but it also determines their corresponding periods and locates their corresponding positions. Thus, there are no presumptions of the period value, and so obscure periodic patterns can be defined as follows.

Definition 2 *If a time series T of length n contains a symbol s that is periodic with period p at position l with respect to an arbitrary periodicity threshold, then a single-symbol periodic* *pattern of length p is formed by putting the symbol s in position l and putting the "don't care" symbol § in all other positions. The support of such single-symbol periodic pattern is estimated by $\frac{\mathcal{F}_2(s,\pi_{p,l}(T))}{\lceil (n-l)/p \rceil - 1}$.*

For example, in the time series $T = abcabbabcb$, the pattern a§§ is a periodic pattern of length 3 with a support value of 2/3, and also the pattern §b§ is a periodic pattern of length 3 with a support value of 1. However, we cannot deduce that the pattern ab§ is also periodic since we cannot estimate its support[1]. The only thing we know for sure is that its support value will not exceed 2/3.

Definition 3 *In a time series T of length n, let $S_{p,l}$ be the set of all the symbols that are periodic with period p at position l with respect to an arbitrary periodicity threshold. Let S^p be the Cartesian product of all $S_{p,l}$ in an ascending order of l, that is $S^p = (S_{p,0} \cup \{\S\}) \times (S_{p,1} \cup \{\S\}) \times \ldots \times (S_{p,p-1} \cup \{\S\})$. Every ordered pair $(s_0, s_1, \ldots, s_{p-1})$ that belongs to S^p corresponds to a candidate periodic pattern of the form $s_0 s_1 \ldots s_{p-1}$ where $s_i \in S_{p,i} \cup \{\S\}$.*

For example, in the time series $T = abcabbabcb$, we have $S_{3,0} = \{a\}$, $S_{3,1} = \{b\}$, and $S_{3,2} = \{\}$, then the candidate periodic patterns are a§§, §b§, and ab§, ignoring the "*don't care*" pattern §§§.

3 Mining Obscure Periodic Patterns

Let us assume first that for a specific time series T, the period p is known. Then, the problem is reduced to mining the periodic patterns of length p, or in other words to detect the symbols that are periodic with period p. A way to solve this simpler problem is to shift the time series p positions, denoted as $T^{(p)}$, and compare this shifted version $T^{(p)}$ to the original version T. For example, if $T = abcabbabcb$, then shifting T 3 positions results in $T^{(3)} = $ §§§$abcabba$. Comparing T to $T^{(3)}$ results in four symbol matches that, if the symbols are mapped in a particular way, can reveal that those four matches are actually two for the symbol a both at position 0, and two for the symbol b both at position 1.

Therefore, our proposed algorithm for mining obscure periodic patterns relies on two main ideas. The first is to obtain a mapping scheme for the symbols, which reveals, upon comparison, the symbols that match and their corresponding positions. Turning back to the original problem where the period is unknown, the second idea is to use the concept of convolution in order to shift and compare the time series for all possible values of the period, and hence detect all the symbol periodicities for all the periods in one pass. The remaining part of this section describes those ideas in detail starting by defining the concept of convolution (Section 3.1) as it derives our mapping scheme (Section 3.2).

3.1 Convolution

A convolution [8] is defined as follows. Let $x = [x_0, x_1, \ldots, x_{n-1}]$ and $y = [y_0, y_1, \ldots, y_{n-1}]$ be two finite length sequences of numbers[2], each of length n. The convolution of x and y is defined as another finite length sequence $x \otimes y$ of length n such that $(x \otimes y)_i = \sum_{j=0}^{i} x_j y_{i-j}$ for $i = 0, 1, \ldots, n-1$. Let $x' = [x'_0, x'_1, \ldots, x'_{n-1}]$ denote the reverse of the vector x, i.e., $x'_i = x_{n-1-i}$. Taking the convolution of x' and y, and obtaining its reverse leads to the following:

[1]This is similar to the Apriori property of the association rules [2], that is if A and B are two frequent itemsets, then AB is a candidate frequent itemset that may turn out to be infrequent.

[2]The general definition of convolution does not assume equal length sequences. We adapted the general definition to conform to our problem, in which convolutions only take place between equal length sequences.

$(x' \otimes y)'_i = (x' \otimes y)_{n-1-i} = \sum_{j=0}^{n-1-i} x'_j y_{n-1-i-j} = \sum_{j=0}^{n-1-i} x_{n-1-j} y_{n-1-i-j}$, i.e.,

$$\begin{aligned}
&(x' \otimes y)'_0 = x_0 y_0 + x_1 y_1 + \cdots + x_{n-1} y_{n-1}, \\
&(x' \otimes y)'_1 = x_1 y_0 + x_2 y_1 + \cdots + x_{n-1} y_{n-2}, \\
&\vdots \\
&(x' \otimes y)'_{n-1} = x_{n-1} y_0
\end{aligned} \; .$$

In other words, the component of the resulting sequence at position i corresponds to positioning one of the input sequences in front of position i of the other input sequence.

Based on the mapping scheme described in the next section, our proposed algorithm converts the time series into two identical finite sequences of numbers, reverses one of them, performs the convolution between them, and then reverses the output. The component values of the resulting sequence will be analyzed exhaustively to get the periodic symbols and their corresponding periods and positions (Section 3.2).

It is well known that convolution can be calculated by fast Fourier transform (FFT) [15] as follows:

$$x \otimes y = \mathrm{FFT}^{-1}\Big(\mathrm{FFT}(x) \cdot \mathrm{FFT}(y)\Big).$$

Therefore, this allows us to achieve two key benefits. First, the time complexity is reduced to $O(n \log n)$, where an arithmetic operation takes a constant time. Second, an external FFT algorithm [19] can be used for large sizes of databases mined while on disk.

3.2 Mapping Scheme

Let $T = t_0, t_1, \ldots, t_{n-1}$ be a time series of length n, where t_i's are symbols from a finite alphabet Σ of size σ. Let Φ be a mapping for the symbols of T such that $\Phi(T) = \Phi(t_0), \Phi(t_1), \ldots, \Phi(t_{n-1})$. Let $C^T = (\Phi(T)' \otimes \Phi(T))'$, and c_i^T be the ith component of C^T. The challenge to our one pass algorithm is to obtain a mapping Φ of the symbols, which satisfies two conditions: (i) when the symbols match, this should contribute a non-zero value in the product $\Phi(t_j) \cdot \Phi(t_{i-j})$, otherwise it should contribute 0, and (ii) the value of each component of C^T, $c_i^T = \sum_{j=0}^{i} \Phi(t_j) \cdot \Phi(t_{i-j})$, should identify the symbols that cause the occurrence of this value and their corresponding positions.

We map the symbols to the binary representation of increasing powers of two [1]. For example, if the time series contains only the 3 symbols a, b, and c, then a possible mapping could be $a : 001$, $b : 010$, and $c : 100$, corresponding to power values of 0, 1, and 2, respectively. Hence, a time series of length n is converted to a binary vector of length σn. For example, let $T = acccabb$, then T is converted to the binary vector $\bar{T} = 001100100100001010010$. Adopting regular convolution, defined previously, results in a sequence $C^{\bar{T}}$ of length σn. Considering only the n positions $0, \sigma, 2\sigma, \ldots, (n-1)\sigma$, which are the exact start positions of the symbols, gives back the sequence C^T. The latter derivation of C^T can be written as $C^T = \pi_{\sigma,0}(C^{\bar{T}})$ using the projection notation defined in Section 2.2.

The first condition is satisfied since the only way to obtain a value of 1 contributing to a component of C^T is that this 1 comes from the same symbol. For example, for $T = acccabb$, although $c_1^{\bar{T}} = 1$, this is not considered one of C^T components. However, $c_3^{\bar{T}} = 3$ and so $c_1^T = 3$, which corresponds to three matches when T is compared to $T^{(1)}$. Those matches are seen from the manual inspection of T to be two c's and one b. Nevertheless, it is not possible to determine those symbols only by examining the value of c_1^T, i.e., the second condition is not yet satisfied. Therefore, we modify the convolution definition to be $(x \otimes y)_i = \sum_{j=0}^{i} 2^j x_j y_{i-j}$. The reason for adding the coefficient 2^j is to get a different contribution for each match, rather than an unvarying contribution of 1. For example, when the new definition of convolution is used for the previous example, $c_1^T = 2^1 + 2^{11} + 2^{14} = 18434$. Figure 1 illustrates this calculation. The powers of 2 for this value are 1, 11, and 14. Examining those powers modulo 3, which is the size of the alphabet in this particular example, results in 1, 2 and 2, respectively, which correspond to the symbols b, c, and c, respectively.

$\bar{T}$:	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0
$\bar{T}^{(3)}$:				0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	1	0
$c_3^{\bar{T}} = c_1^T =$							2^{14}		+	2^{11}		+								2^1	
$\bar{T}$:	0	0	1	1	0	0	1	0	0	1	0	0	0	0	1	0	1	0	0	1	0
$\bar{T}^{(12)}$:													0	0	1	1	0	0	1	0	0
$c_{12}^{\bar{T}} = c_4^T =$															2^6						

Figure 1: A Clarifying Example for the Mapping Scheme.

Figure 1 gives another example for c_4^T containing only one power of 2, which is 6, that corresponds to the symbol a since $6 \bmod 3 = 0$ and a was originally mapped to the binary representation of 2^0. This means that comparing T to $T^{(4)}$ results in only one match of the symbol a. Moreover, the power value of 6 reveals that the symbol a is at position 0 in $T^{(4)}$. Note that in the binary vector, the most significant bit is the leftmost one, whereas the most significant position of the time series T is the rightmost one. Therefore, not only can the power values reveal the number of matches of each symbol at each period, they also reveal their corresponding starting positions. This latter observation complies with the definition of symbol periodicity.

Formally, let $s_0, s_1, \ldots, s_{\sigma-1}$ be the symbols of the alphabet of a time series T of length n. Assume that each symbol s_k is mapped to the σ-bit binary representation of 2^k to form $\bar{T}$. The sequence $C^{\bar{T}}$ is computed such that $c_i^{\bar{T}} = \sum_{j=0}^{i} 2^j \bar{t}_j \cdot \bar{t}_{i-j}$ for $i = 0, 1, \ldots, \sigma n - 1$. Thus, $C^T = \pi_{\sigma,0}(C^{\bar{T}})$. Assume that c_p^T is a non-zero component of C^T. Let W_p denote the set of powers of 2 contained in c_p^T, i.e.,

$$W_p = \{w_{p,1}, w_{p,2}, \ldots\}$$

where $c_p^T = \sum_h 2^{w_{p,h}}$, and let

$$W_{p,k} = \{w_{p,h} : w_{p,h} \in W_p \wedge w_{p,h} \bmod \sigma = k\}.$$

As shown in the previous example, the cardinality of each $W_{p,k}$ represents the number of matches of the symbol s_k when T is compared to $T^{(p)}$. Moreover, let

$$W_{p,k,l} = \{w_{p,h} : w_{p,h} \in W_{p,k} \wedge (n - p - 1 - \lfloor w_{p,h}/\sigma \rfloor) \bmod p = l\}.$$

Revisiting the definition of symbol periodicity, we observe that the cardinality of each $W_{p,k,l}$ is equal to the desired value of $\mathcal{F}_2(s_k, \pi_{p,l}(T))$. Working out the example of Section 2.2 where $T = abcabbabcb$, $n = 10$, and $\sigma = 3$, let $s_0, s_1, s_2 = a, b, c$, respectively. Then, for $p = 3$, $W_3 = \{18, 16, 9, 7\}$, $W_{3,0} = \{18, 9\}$, $W_{3,0,0} = \{18, 9\} \Rightarrow \mathcal{F}_2(a, \pi_{3,0}(T)) = 2$ which conforms to the results obtained previously. As another example, if $T = cabccbacd$ where $n = 9$, $\sigma = 4$, and $s_0, s_1, s_2, s_3 = a, b, c, d$, respectively, then for $p = 4$, $W_4 = \{18, 6\}$, $W_{4,2} = \{18, 6\}$, $W_{4,2,0} = \{18\} \Rightarrow \mathcal{F}_2(c, \pi_{4,0}(T)) = 1$, and $W_{4,2,3} = \{6\} \Rightarrow \mathcal{F}_2(c, \pi_{4,3}(T)) = 1$ which are correct since $\pi_{4,0}(T) = ccd$ and $\pi_{4,3}(T) = cc$.

One final detail about our algorithm is the use of the values $w_{p,h}$ to estimate the support of the candidate periodic patterns formed according to Definition 3. Let $s_{j_0} s_{j_1} \ldots s_{j_{p-1}}$ be a candidate periodic pattern that is not a single-symbol pattern nor the "*don't care*" pattern, i.e., at least 2 s_{j_i}'s are not §. The set $W_{p,j_i,i}$ contains the values responsible for the symbol

```
Input:  a time series $T = t_0, t_1, \ldots, t_{n-1}$, and the periodicity threshold $\psi$.

Output:  candidate periodic patterns for $T$ (a pattern and its support value).

Algorithm:

  1. Select an arbitrary ordering for the symbols $s_0, s_1, \ldots, s_{\sigma-1}$.
  2. From $T$, obtain a binary vector $\bar{T}$ by replacing every symbol $s_k$ by the $\sigma$-bit
     binary representation of $2^k$.
  3. Compute the sequence $C^{\bar{T}}$ where $c_i^{\bar{T}} = \sum_{j=0}^{i} 2^j \bar{t}_j \cdot \bar{t}_{i-j}$, and consequently $C^T =$
     $\pi_{\sigma,0}(C^{\bar{T}})$.
  4. For $p = 1, 2, \ldots, n/2$,
     (a) compute the set $W_p = \{w_{p,1}, w_{p,2}, \ldots\}$ where $c_p^T = \sum_h 2^{w_{p,h}}$,
     (b) for $k = 0, 1, \ldots, \sigma - 1$, and $l = 0, 1, \ldots, p - 1$, compute the sets $W_{p,k}$ and $W_{p,k,l}$
         as defined, and consequently the values of $\mathcal{F}_2(s_k, \pi_{p,l}(T))$,
     (c) if $\frac{\mathcal{F}_2(s_k, \pi_{p,l}(T))}{\lceil (n-l)/p \rceil - 1} \geq \psi$, then $p$ is a candidate period for the symbol $s_k$ at
         position $l$.
     (d) form and output the single-symbol periodic patterns according to
         Definition 2.
     (e) form the set of candidate periodic patterns according to Definition 3,
         calculate the support of each as defined, and output those whose support
         values are not less than the periodicity threshold.
```

Figure 2: Obscure Periodic Patterns Mining Algorithm

$s_{j_i} \neq \S$. Let W^p be a subset of the Cartesian product of the sets $W_{p,j_i,i}$ for all i where $s_{j_i} \neq \S$ such that all the values in an ordered pair should have the same value of $\lfloor \frac{n-p-1-\lfloor w_{p,h}/\sigma \rfloor}{p} \rfloor$. The support estimate of that candidate periodic pattern is $\frac{|W^p|}{\lfloor n/p \rfloor}$. For example, if $T = abcabbabcb$, $W_{3,0,0} = \{18, 9\}$ corresponds to the symbol a, and $W_{3,1,1} = \{16, 7\}$ corresponds to the symbol b, then for the candidate periodic pattern $ab\S$, $W^p = \{(18, 16), (9, 7)\}$, and the support of this pattern is $2/3$.

Therefore, our algorithm scans the time series once to convert it into a binary vector according to the proposed mapping, performs the modified convolution on the binary vector, and analyzes the resulting values to determine the symbol periodicities and consequently the single-symbol periodic patterns. Then, the set of candidate periodic patterns is formed and the support of each pattern is estimated.

The complexity of our algorithm is the complexity of the convolution step, which is $O(n \log n)$ when performed using FFT. Note that adding the coefficient 2^j to the convolution definition still preserves that

$$x \otimes y = \mathrm{FFT}^{-1}\Big(\mathrm{FFT}(x) \cdot \mathrm{FFT}(y)\Big).$$

Although one might assume that the algorithm requires handling of large numbers $O(2^{\sigma n})$, careful implementation using bit structures and bit operations avoids such complexity. The complete algorithm is sketched in Figure 2.

4 Experimental Study

This section contains the results of an extensive experimental study that examines the proposed obscure periodic patterns mining algorithm for different aspects, the most important of which is the correctness of the symbol periodicity detection phase that is studied solely in Section 4.1, followed by studying the time performance in Section 4.2. As data inerrancy is inevitable, Section 4.3 scrutinizes the resilience of the proposed algorithm to various kinds of noise that can occur in time series data. The practicality and usefulness of the results are explored using real data experiments shown in Section 4.4. Moreover, the periodic trends algorithm of [13] is chosen to be compared versus our proposed algorithm throughout the experiments. As discussed earlier in Section 1.1, the periodic trends algorithm of [13] is the fastest among the ones in the literature that detects all the valid candidate periods.

In our experiments, we exploit synthetic data as well as real data. We generate controlled synthetic time series data by tuning some parameters, namely, data distribution, period, alphabet size, type, and amount of noise. Both uniform and normal data distributions are considered. Types of noise include replacement, insertion, deletion, or any combination of them. Inerrant data is generated by repeating a pattern, of length equal to the period, that is randomly generated from the specified data distribution. The pattern is repeated till it spans the specified time series length. Noise is introduced randomly and uniformly over the whole time series. Replacement noise is introduced by altering the symbol at a randomly selected position in the time series by another. Insertion or deletion noise is introduced by inserting a new symbol or deleting the current symbol at a randomly selected position in the time series.

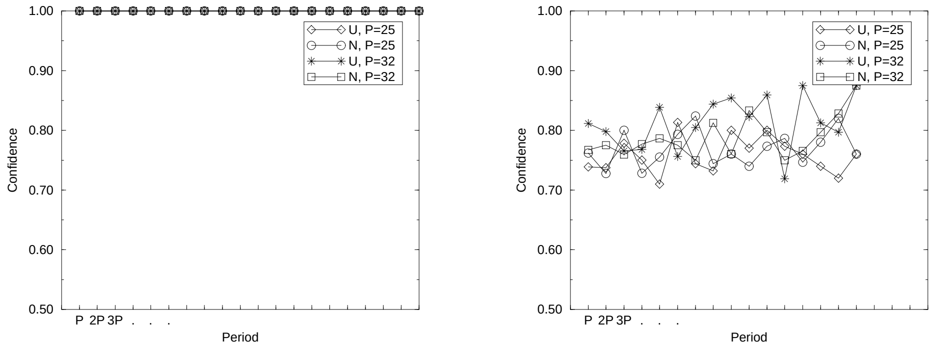
Two databases serve the purpose of real data experiments. The first one is a relatively small database that contains the daily power consumption rates of some customers over a period of one year. It is made available through the CIMEG[3] project. The database size is approximately 5 Megabytes. The second database is a Wal-Mart database of 70 Gigabytes, which resides on an NCR Teradata Server running the NCR Teradata Database System. It contains sanitized data of timed sales transactions for some Wal-Mart stores over a period of 15 months. The timed sales transactions data has a size of 130 Megabytes. In both databases, the numeric data values are discretized[4] into five levels, i.e., the alphabet size equals to 5. The levels are *very low*, *low*, *medium*, *high*, and *very high*. For the power consumption data, discretizing is based on discussions with domain experts (*very low* corresponds to less than 6000 Watts/Day, and each level has a 2000 Watts range). For the timed sales transactions data, discretizing is based on manual inspection of the values (*very low* corresponds to zero transactions per hour, *low* corresponds to less than 200 transactions per hour, and each level has a 200 transactions range).

4.1 Verification of Correctness

Synthetic data, both inerrant and noisy, are used in this experiment in order to inspect the correctness of the proposed algorithm. The correctness measure will be the ability of the algorithm to detect the symbol periodicities that are artificially embedded into the synthetic data. Figure 3 gives the results of this experiment. We use the symbols "U" and "N" to denote the uniform and the normal distributions, respectively; and the symbol "P" to denote the period. Recall that inerrant synthetic data is generated in such a way that it is perfectly periodic, i.e., the symbol periodicities are embedded at periods $P, 2P, \ldots$. The confidence is

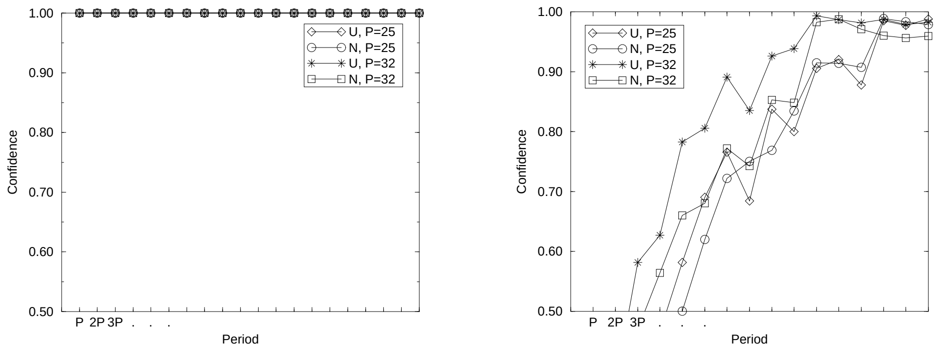
[3]CIMEG: Consortium for the Intelligent Management of the Electric Power Grid. http://helios.ecn.purdue.edu/~cimeg.

[4]The problem of discretizing time series into a finite alphabet is orthogonal to our problem and is beyond the scope of this paper (see [9] for an exhaustive overview of discretizing techniques).



(a) Inerrant Data (b) Noisy Data

Figure 3: Verification of the Correctness of the Obscure Periodic Patterns Mining Algorithm.



(a) Inerrant Data (b) Noisy Data

Figure 4: Correctness of Periodic Trends Algorithm

the minimum periodicity threshold value required to detect a specific period. If the data is perfectly periodic, the confidence of all the periodicities should be 1. Time series lengths of 1M symbols are used with alphabet size of 10. The values collected are averaged over 100 runs. Figure 3(a) shows that the algorithm is able to detect all the embedded periodicities in the inerrant time series data with the highest possible confidence. Although Figure 3(b) shows an expected decrease in the confidence values due to the presence of noise, the values are still high enough (above 70%) to consider the algorithm as correct. Figure 3(b) shows also an unbiased behavior of the algorithm with respect to the period unlike the algorithm of [13] as shown in Figure 4.

Figure 4 gives the results of the same experiment for the periodic trends algorithm of [13]. However, in order to inspect the correctness of that algorithm, we will briefly discuss its output. The algorithm computes an absolute value for each possible period, and then it outputs the periods that correspond to the minimum absolute values as the most candidate periods. In other words, if the absolute values are sorted in ascending order, the corresponding periods will be ordered from the most to the least candidate. Therefore, it is the rank of the period in this

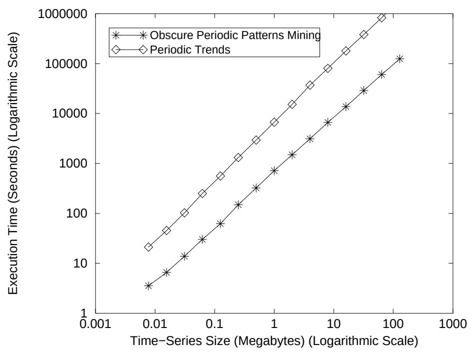


Figure 5: Time Behavior of the Obscure Periodic Patterns Mining Algorithm.

candidacy order that favors a period over another rather than its corresponding absolute value. Normalizing the ranks to be real-valued ranging from 0 to 1 is trivial. The normalized rank can be considered as the confidence value of each period (the most candidate period that has rank 1 will have normalized rank value of 1, and less candidate periods that have lower ranks will have lower normalized rank values). This means that if the data is perfectly periodic, the embedded periodicities should have the highest ranks and so confidence values close to 1. Figure 4(b) shows a biased behavior of the periodic trends algorithm with respect to the period, as it favors the higher period values. However, we believe that the smaller periods are more accurate than the larger ones since they are more informative. For example, if the power consumption of a specific customer has a weekly pattern, it is more informative to report the period of 7 days than to report the periods of 14, 21, or other multiples of 7.

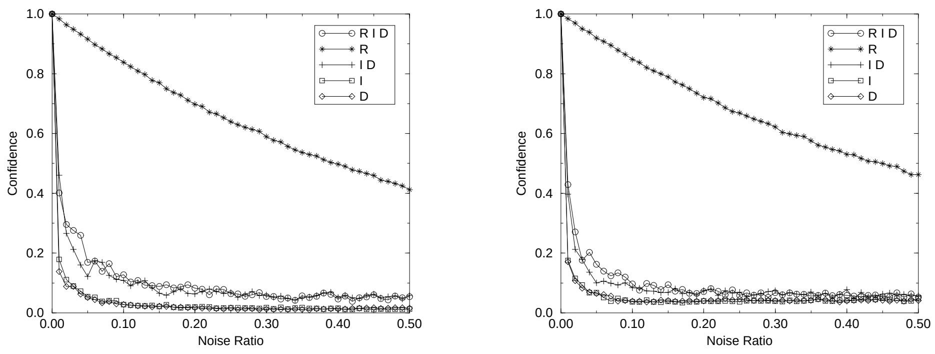
4.2 Time Performance

As mentioned earlier, our proposed algorithm is the only one that mines periodic patterns with unknown period in one pass. Having said that, there is no direct time performance comparison between our proposed algorithm and the ones in the literature. However, we compare the time behavior of the periodic trends algorithm of [13] to that of the periodicity detection phase of our proposed algorithm.

Figure 5 exhibits the time behavior of both algorithms with respect to the time series length. Wal-Mart timed sales transactions data is used in different portion lengths of powers of 2 up to 128 Megabytes. Figure 5 shows that the execution time of our proposed algorithm is linearly proportional to the time series length. More importantly, the figure shows that our proposed algorithm outperforms the periodic trends algorithm of [13]. This experimental result agrees with the theoretical results as the periodic trends algorithm [13] performs in $O(n \log^2 n)$ time whereas our proposed algorithm performs in $O(n \log n)$ time.

4.3 Resilience to Noise

As mentioned before, there are three types of noise: replacement, insertion and deletion noise. This set of experiments studies the behavior of the proposed obscure periodic patterns mining algorithm towards these types of noise as well as different combinations of them. Results are given in Figure 6 in which we use the symbols “R”, “I”, and “D” to denote the three types of noise, respectively. Two or more types of noise can be combined, e.g., “R I D” means that the noise ratio is distributed equally among replacement, insertion, and deletion, while “I D”



(a) Uniform, Period=25 (b) Normal, Period=32

Figure 6: Resilience to Noise of the Obscure Periodic Patterns Mining Algorithm.

means that the noise ratio is distributed equally among insertion and deletion only. Time series lengths of 1M symbols are used with alphabet size of 10. The values collected are averaged over 100 runs. Since the behaviors were similar regardless of the period or the data distribution, an arbitrarily combination of period and data distribution is chosen. Figure 6 shows that the algorithm is very resilient to replacement noise. At 40% periodicity threshold, the algorithm can tolerate 50% replacement noise in the data. When the other types of noise get involved separately or with replacement noise, the algorithm performs poorly. However, the algorithm can still be considered roughly resilient to those other types of noise since periodicity thresholds in the range 5% to 10% are not uncommon.

4.4 Real Data Experiments

Tables 1, 2 and 3 display the output of the obscure periodic patterns mining algorithm for the Wal-Mart and CIMEG data for different values of the periodicity threshold. We examine first the symbol periodicities in Table 1 and then the periodic patterns in Tables 2 and 3. Clearly, the algorithm outputs fewer periodic patterns for higher threshold values, and the periodic patterns detected with respect to a certain value of the periodicity threshold are enclosed within those detected with respect to a lower value. To verify its correctness, the algorithm should at least output the periodic patterns of periods that are expected in the time series. Wal-Mart data has an expected period value of 24 that corresponds to the daily pattern of number of transactions per hour. CIMEG data has an expected period value of 7 that corresponds to the weekly pattern of power consumption rates per day.

Table 1 shows that for Wal-Mart data, a period of 24 hours is detected when the periodicity threshold is 70% or less. In addition, the algorithm detects many more periods, some of which are quite interesting. A period of 168 hours (24×7) can be explained as the weekly pattern of number of transactions per hour. A period of 3961 hours shows a periodicity of exactly 5.5 months plus one hour, which can be explained as the daylight savings hour. One may argue against the clarity of this explanation, yet this proves that there may be obscure periods, unknown a priori, that the algorithm can detect. Similarly, for CIMEG data, the period of 7 days is detected when the threshold is 60% or less. Other clear periods are those that are multiples of 7. However, a period of 123 days is difficult to explain.

Exploring the period of 24 hours for Wal-Mart and that of 7 days for CIMEG data produces

Periodicity Threshold (%)	Wal-Mart Data		CIMEG Data	
	# Periods	Some Periods	# Periods	Some Periods
50	3164	263, 409	103	20, 34
55	2777	337, 385	95	128
60	2728	481	95	7, 46
65	2612	503	87	14, 54
70	2460	505	80	32
75	2447	577	79	28, 52
80	2328	647	74	38
85	2289	791	72	116
90	2285	721	72	21
95	2281	24, 168	71	35, 73

Table 1: Periods

Periodicity Threshold (%)	Wal-Mart Data Period=24		CIMEG Data Period=7	
	# Patterns	Patterns	# Patterns	Patterns
50	13	(b,5), (d,17)	2	(a,3)
55	11	(b,8)	1	(b,2)
60	10	(b,6), (c,9)	1	
65	8	(a,21)	0	
70	7	(a,3)	0	
75	6	(b,7)	0	
80	6		0	
85	5	(a,0), (a,1), (a,2), (a,22) (a,23)	0	
90	5		0	
95	5		0	

Table 2: Single-Symbol Periodic Patterns

the results given in Table 2. Note that single-symbol periodic pattern is reported as a pair, consisting of a symbol and a starting position for a certain period. For example, (b,7) for Wal-Mart data with respect to a periodicity threshold of 80% or less represents the single-symbol periodic pattern §§§§§§§b§§§§§§§§§§§§§§§§. Knowing that the symbol b represents the *low* level for Wal-Mart data (less than 200 transactions per hour), this periodic pattern can be interpreted as *less than 200 transactions per hour occur in the 7th hour of the day (between 7:00am and 8:00am) for* 80% *of the days.* As another example, (a,3) for CIMEG data with respect to a periodicity threshold of 50% or less represents the single-symbol periodic pattern §§§a§§§. Knowing that the symbol a represents the *very low* level for CIMEG data (less than 6000 Watts/Day), this periodic pattern can be interpreted as *less than 6000 Watts/Day occur in the 4th day of the week for* 50% *of the days.* Finally, Table 3 gives the final output of periodic patterns of Wal-Mart data for the period of 24 hours for periodicity threshold of 35%. Each pattern can be interpreted in a similar way to the above.

5 Conclusions

In this paper, we have developed a one pass algorithm for mining periodic patterns in time series of unknown or obscure periods, where discovering the periods is part of the mining algorithm. Based on an adapted definition of convolution, the algorithm is computationally efficient as it scan the data only once and takes $O(n \log n)$ time, for a time series of length n. We have

Periodic Pattern	**Support (%)**
aaaa§§§§§§§§§§§§§§§§§*aaa*	49.78166
aaaa§§§*b*§§§§§§§§§§§§§*aaa*	42.57642
aaaa§§§*b*§§§§§§§§§*d*§§§*aaa*	38.56768
§§§§§*bbbbc*§§§§§§§§§§§§*aa*	35.80786

Table 3: Periodic Patterns for Wal-Mart Data

defined the periodic pattern in a novel way that the period is a variable rather than an input parameter. An empirical study of the algorithm using real-world and synthetic data proves the practicality of the problem, validates the correctness of the algorithm, and the usefulness of its output patterns.

References

[1] K. Abrahamson. Generalized String Matching. *SIAM Journal on Computing*, Vol. 16, No. 6, pages 1039-1051, 1987.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *Proc. of the 20th Int. Conf. on Very Large Databases*, Santiago, Chile, September 1994.

[3] R. Agrawal and R. Srikant. Mining Sequential Patterns. In *Proc. of the 11th Int. Conf. on Data Engineering*, Taipei, Taiwan, March 1995.

[4] W. Aref, M. Elfeky, and A. Elmagarmid. Incremental, Online, and Merge Mining of Partial Periodic Patterns in Time-Series Databases. To appear in *IEEE Transactions on Knowledge and Data Engineering.*

[5] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. Sequential Pattern Mining using A Bitmap Representation. In *Proc. of the 8th Int. Conf. on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 2002.

[6] C. Berberidis, W. Aref, M. Atallah, I. Vlahavas, and A. Elmagarmid. Multiple and Partial Periodicity Mining in Time Series Databases. In *Proc. of the 15th Euro. Conf. on Artificial Intelligence*, Lyon, France, July 2002.

[7] C. Bettini, X. Wang, S. Jajodia, and J. Lin. Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, No. 2, pages 222-237, 1998.

[8] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, MA, 1990.

[9] C. Daw, C. Finney, and E. Tracy. A Review of Symbolic Analysis of Experimental Data. *Review of Scientific Instruments*, Vol. 74, No. 2, pages 915-930, 2003.

[10] M. Garofalakis, R. Rastogi, and K. Shim. SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In *Proc. of the 25th Int. Conf. on Very Large Databases*, Edinburgh, Scotland, UK, September 1999.

[11] J. Han, G. Dong, and Y. Yin. Efficient Mining of Partial Periodic Patterns in Time Series Databases. In *Proc. of the 15th Int. Conf. on Data Engineering*, Sydney, Australia, March 1999.

[12] J. Han, W. Gong, and Y. Yin. Mining Segment-Wise Periodic Patterns in Time Related Databases. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, New York City, New York, August 1998.

[13] P. Indyk, N. Koudas, and S. Muthukrishnan. Identifying Representative Trends in Massive Time Series Data Sets Using Sketches. In *Proc. of the 26th Int. Conf. on Very Large Data Bases*, Cairo, Egypt, September 2000.

[14] E. Keogh, S. Lonardi, and B. Chiu. Finding Surprising Patterns in a Time Series Database in Linear Time and Space. In *Proc. of the 8th Int. Conf. on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 2002.

[15] D. Knuth. *The Art of Computer Programming, Vol. 2.* Addison-Wesley, Reading, MA, 1981.

[16] S. Ma and J. Hellerstein. Mining Partially Periodic Event Patterns with Unknown Periods. In *Proc. of the 17th Int. Conf. on Data Engineering*, Heidelberg, Germany, April 2001.

[17] B. Ozden, S. Ramaswamy, and A. Silberschatz. Cyclic Association Rules. In *Proc. of the 14th Int. Conf. on Data Engineering*, Orlando, Florida, February 1998.

[18] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. In *Proc. of the 5th Int. Conf. on Extending Database Technology*, Avignon, France, March 1996.

[19] J. Vitter. External Memory Algorithms and Data Structures: Dealing with Massive Data. *ACM Computing Surveys*, Vol. 33, No. 2, pages 209-271, June 2001.

[20] J. Yang, W. Wang, and P. Yu Mining Asynchronous Periodic Patterns in Time Series Data. In *Proc. of the 6th Int. Conf. on Knowledge Discovery and Data Mining*, Boston, Massachusetts, August 2000.