Periodicity Detection in Time Series Databases

Mohamed G. Elfeky, Walid G. Aref, *Senior Member*, *IEEE*, and
Ahmed K. Elmagarmid, *Senior Member*, *IEEE*

Abstract—Periodicity mining is used for predicting trends in time series data. Discovering the rate at which the time series is periodic has always been an obstacle for fully automated periodicity mining. Existing periodicity mining algorithms assume that the periodicity rate (or simply the period) is user-specified. This assumption is a considerable limitation, especially in time series data where the period is not known a priori. In this paper, we address the problem of detecting the periodicity rate of a time series database. Two types of periodicities are defined, and a scalable, computationally efficient algorithm is proposed for each type. The algorithms perform in $O(n \log n)$ time for a time series of length n. Moreover, the proposed algorithms are extended in order to discover the periodic patterns of unknown periods at the same time without affecting the time complexity. Experimental results show that the proposed algorithms are highly accurate with respect to the discovered periodicity rates and periodic patterns. Real-data experiments demonstrate the practicality of the discovered periodic patterns.

Index Terms—Periodic patterns mining, temporal data mining, time series forecasting, time series analysis.

1 Introduction

Time series data captures the evolution of a data value over time. Life includes several examples of time series data. Examples are meteorological data containing several measurements, e.g., temperature and humidity, stock prices depicted in financial market, power consumption data reported in energy companies, and event logs monitored in computer networks. Periodicity mining is a tool that helps in predicting the behavior of time series data [23]. For example, periodicity mining allows an energy company to analyze power consumption patterns and predict periods of high and low usage so that proper planning may take place.

Research in time series data mining has concentrated on discovering different types of patterns: sequential patterns [3], [21], [12], [6], temporal patterns [8], periodic association rules [20], partial periodic patterns [14], [13], [25], [4], and surprising patterns [17] 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. They assume that users either know the value of the period beforehand or 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 [15], [7] or on devising special techniques for specific periodicity mining problems [24], [19]. Both approaches require multiple phases over the time series in order to output the periodic patterns themselves.

In this paper, we address the problem of discovering potential periods in time series databases, hereafter referred to as *periodicity detection*. We define two types of periodicities: *segment periodicity* and *symbol periodicity*. Whereas segment periodicity concerns the periodicity of the entire time series, symbol periodicity concerns the periodicities of the various symbols or values of the time series. For each periodicity type, a convolution-based algorithm is proposed and is analyzed, both theoretically and empirically. Furthermore, we extend the symbol periodicity detection algorithm so that it can discover the periodic patterns of unknown periods, termed *obscure periodic patterns*. Hence, we detect periodicity rates as well as frequent periodic patterns simultaneously.

The rest of the paper is structured as follows: In Section 2, we outline the related work with the main emphasis to distinguish the main contributions of this paper. In Section 3, we introduce the notation used throughout the paper, and we formally define the periodicity detection problem as well as the notion of the segment and symbol periodicity types. Sections 4 and 5 describe the two proposed algorithms for periodicity detection in time series databases. Moreover, Section 4 describes the proposed algorithm for mining obscure periodic patterns. In Section 6, the performance of the proposed algorithms is studied, extensively validating the algorithms' accuracy, examining their resilience to noise, and justifying their practicality. Finally, we summarize our findings in Section 7.

2 Background

Discovering the periodicity rate of time series data has drawn the attention of the data mining research community

- W.G. Aref and M.G. Elfeky are with the Department of Computer Sciences, Purdue University, 250 N. University St., West Lafayette, IN 47907-2066. E-mail: {aref, mgelfeky}@cs.purdue.edu.
- A.K. Elmagarmid is with Hewlett-Packard Company, 10955 Tantau Ave., Bldg. 45, ms 4143, Cupertino, CA 95014. E-mail: ahmed_elmagarmid@hp.com.

Manuscript received 3 May 2004; revised 18 Oct. 2004; accepted 4 Feb. 2005; published online 18 May 2005.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0129-0504.

very recently. Indyk et al. [15] 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, which is similar to our notion of segment periodicity (Section 3.3). However, our proposed algorithm for segment periodicity detection (Section 5) performs in $O(n \log n)$ time. We conduct a thorough performance study to compare our proposed segment periodicity detection algorithm to the periodic trends algorithm of [15]. In addition to the saving in time performance, our proposed segment periodicity detection algorithm is more resilient to noise and produces more accurate periods. The proposed segment periodicity detection algorithm favors the shorter periods rather than the longer ones that are favored by the periodic trends algorithm of [15]. The shorter periods are more accurate than the longer ones since they are more informative. For example, if the daily power consumption of a specific customer has a weekly pattern, it is more informative to report a period, say, of length 7, than to report the periods 14, 21, or other multiples of 7.

Specific to partial periodic patterns, Ma and Hellerstein [19] have developed a linear distance-based algorithm for discovering the potential periods regarding the symbols of the time series. In [24], a similar algorithm has been proposed with some pruning techniques. However, both algorithms miss some valid periods since they only consider the adjacent interarrivals. For example, consider a symbol that occurs in a time series in positions 0, 4, 5, 7, and 10. Since that symbol occurs in positions 0, 5, and 10, one of the underlying periods for that symbol should be 5. However, a distance-based algorithm only considers the adjacent inter-arrival times 4, 1, 2, and 3 as candidate periods, which clearly do not include the value 5. Should it be extended to include all possible interarrivals, the complexity of a distance-based algorithm [24], [19] would increase to $O(n^2)$. Although Berberidis et al. [7] have proposed an algorithm that considers all possible potential periods, their algorithm is inefficient as it considers one symbol at a time. Moreover, the algorithms of [24], [19], [7] require additional phase over the time series in order to output the periodic patterns. Not only does our proposed symbol periodicity detection algorithm (Section 4) perform in $O(n \log n)$, it also discovers all possible potential periods as well as their corresponding periodic patterns simultaneously.

Hence, we distinguish this paper by the following contributions:

1. We introduce a new notion of time series periodicity in terms of the symbols of the time series, which is termed symbol periodicity.
2. We propose a convolution-based algorithm for symbol periodicity detection. Convolution allows the algorithm to consider all symbols and periods at the same time and to discover the potential periods and the obscure periodic patterns simultaneously.
3. We propose a new algorithm for segment periodicity detection that outperforms the periodic trends algorithm of [15] with respect to time performance, resilience to noise, and accuracy of output periods.

3 Periodicity Detection Problem

3.1 Notation

Assume that a sequence of n time-stamped feature values is collected in a time series. For a given feature e, let e_i be the value of the feature at time-stamp i. The time series of feature e is represented as $T = e_0, e_1, \ldots, e_{n-1}$. For example, the feature in a time series for power consumption might be the hourly power consumption rate of a certain customer, while the feature in a time series for stock prices might be the final daily stock price of a specific company. If we discretize[1] the time series feature values into nominal discrete levels[2] 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 as $\Sigma = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \cdots\}$. Hence, we can view T as a sequence of n symbols drawn from a finite alphabet Σ.

A time series may also be a sequence of n time-stamped events drawn from a finite set of nominal event types. An example is the event log in a computer network that monitors the various events that 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.

3.2 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 time-stamps. For example, in the time series $T = \mathtt{abcabbabcb}$, the symbol b is periodic with period 4 since b exists every four time-stamps (in positions 1, 5, and 9). Moreover, the symbol a is periodic with period 3 since a exists almost every three time-stamps (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,

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

where $0 \leq l < p$, $m = \lceil (n-l)/p \rceil$, and n is the length of T. For example, if $T = \mathtt{abcabbabcb}$, then $\pi_{4,1}(T) = \mathtt{bbb}$ and $\pi_{3,0}(T) = \mathtt{aaab}$. 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 time-stamps. 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, $\pi_{p,l}(T) = \mathtt{abcbac}$, this means that the symbol changes every p time-stamp 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 time-stamps from the previous appearance, which means that p is a potential period for s. Let $\mathcal{F}_2(s,T)$ denote the number of

1. The problem of discretizing time series into a finite alphabet is orthogonal to our problem and is beyond the scope of this dissertation. See [10], [16] for an exhaustive overview of discretizing and segmentation techniques.

2. Nominal values are distinguished by name only and do not have an inherent order or a distance measure.

times the symbol s occurs in two consecutive positions in the time series T. For example, if $T = \texttt{abbaaabaa}$, then $\mathcal{F}_2(\texttt{a}, T) = 3$ and $\mathcal{F}_2(\texttt{b}, T) = 1$.

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

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

3.2.1 Obscure Periodic Patterns

The main advantage of Definition 1 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 periodic single-symbol pattern of length p is formed by inserting the symbol s in position l and inserting the "don't care" symbol $*$ in all other positions.*

The support of a periodic single-symbol pattern, formed according to Definition 2, is estimated by $\frac{\mathcal{F}_2(s, \pi_{p,l}(T))}{\lceil (n-l)/p \rceil - 1}$. For example, in the time series $T = \texttt{abcabbabcb}$, the pattern $\texttt{a}**$ is a periodic single-symbol pattern of length 3 with a support value of 2/3, and so is the single-symbol pattern $*\texttt{b}*$ with a support value of 1. However, we cannot deduce that the pattern $\texttt{ab}*$ is also periodic since we cannot estimate its support.[3] 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,*

$$S^p = (S_{p,0} \cup \{*\}) \times (S_{p,1} \cup \{*\}) \times \ldots \times (S_{p,p-1} \cup \{*\}).$$

Every ordered p-tuple $(s_0, s_1, \ldots, s_{p-1})$ that belongs to S^p corresponds to a candidate periodic pattern of the form $s_0 s_1 \ldots s_{p-1}$, where $s_i \in S_{p,i} \cup \{*\}$.

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

3.3 Segment Periodicity

Unlike symbol periodicity that focuses on the symbols (where different symbols may have different periods), segment periodicity focuses on the entire time series. A time series T is said to be periodic with a period p if it can be divided into equal-length segments, each of length p, that are "almost" similar. For example, the time series $T = \texttt{abcabcabc}$ is clearly periodic with a period 3. Likewise, the time series $T = \texttt{abcabdabc}$ is periodic with a period 3 despite the fact that its second segment is not identical to the other segments. Since the symbols are considered nominal, i.e., no inherent order is assumed, we can simply use Hamming distance to measure the similarity between two segments:

$$H(u, v) = \sum_{j=0}^{m-1} \begin{cases} 1 & u_j \ne v_j \\ 0 & u_j = v_j, \end{cases} \quad S(u, v) = 1 - H(u, v)/m,$$

where u and v are two segments of equal length m; u_j and v_j are the symbols at position j of the two segments u and v, respectively; H is the Hamming distance; and S is the similarity measure. The similarity function is defined in such a way that the higher its value, the more similar the two segments are, and $u = v \Leftrightarrow S(u, v) = 1$. Segment periodicity is therefore defined as follows.

Definition 4. *If a time series T of length n can be sliced into equal-length segments $T_0, T_1, \ldots, T_i, \ldots, T_N$, each of length p, where $T_i = e_{ip}, \ldots, e_{ip+p-1}$, $N = \lfloor n/p \rfloor - 1$, $S(T_i, T_j) \ge \tau \quad \forall i, j = 0, 1, \ldots, N$, and $0 \le \tau \le 1$, then T is said to be periodic with a period p with respect to periodicity threshold τ.*

Definition 5. *A period p is said to be perfect if $S(T_i, T_j) = 1$ $\forall i, j = 0, 1, \ldots, N$.*

For example, the time series $T = \texttt{abcabcabc}$ has a perfect period 3 and the time series $T = \texttt{abcabdabc}$ is periodic with a period 3 with respect to any periodicity threshold $\tau \le 2/3$.

4 Symbol Periodicity Detection

Assume first that the period p is known for some symbols of a specific time series T. Then, the problem is reduced to detect those 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 $T = \texttt{abcabbabcb}$, then shifting T three positions results in $T^{(3)} = ***\texttt{abcabba}$. Comparing T to $T^{(3)}$ results in four symbol matches. If the symbols are mapped in a particular way, we can deduce 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 symbol periodicity detection 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. Going 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 at the same time. The remaining part of this section describes those ideas in detail starting by defining the concept of convolution (Section 4.1) as it derives our mapping scheme (Section 4.2).

3. 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.

4.1 Convolution

A convolution [9] 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,[4] 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:

$$\begin{aligned}(X' \otimes Y)'_i = (X' \otimes Y)_{n-1-i} &= \sum_{j=0}^{n-1-i} x'_j y_{n-1-i-j} \\ &= \sum_{j=0}^{n-1-i} x_{n-1-j} y_{n-1-i-j},\end{aligned}$$

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 shifting one of the input sequences i positions and comparing it to the other input sequence.

Therefore, the symbol periodicity detection algorithm performs the following steps:

1. Converts the time series T into two finite sequences of numbers $\Phi(T)$ and $\Phi(T)'$, where $\Phi(T)'$ is the reverse of $\Phi(T)$ (based on the mapping scheme Φ described in Section 4.2);
2. Performs the convolution between the two sequences $\Phi(T)' \otimes \Phi(T)$;
3. Reverses the output $\Big(\Phi(T)' \otimes \Phi(T)\Big)'$;
4. Analyzes the component values of the resulting sequence to get the periodic symbols and their corresponding periods and positions (Section 4.2).

It is well-known that convolution can be computed by the fast Fourier transform (FFT) [18] as follows:

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

This computation reduces the time complexity of the convolution to $O(n \log n)$. The brute force approach of shifting and comparing the time series for all possible values of the period has the time complexity $O(n^2)$. Moreover, an external FFT algorithm [22] can be used for large sizes of databases mined while on disk.

4. The general definition of convolution does not assume equal-length sequences. We adapt the general definition to conform to our problem, in which convolutions only take place between equal-length sequences.

$\bar{T}$:	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						

Fig. 1. A clarifying example for the mapping scheme.

4.2 Mapping Scheme

Let $T = e_0, e_1, \ldots, e_{n-1}$ be a time series of length n, where e_is are symbols from a finite alphabet Σ of size σ. Let Φ be a mapping for the symbols of T such that $\Phi(T) = \Phi(e_0), \Phi(e_1), \ldots, \Phi(e_{n-1})$. Let $C(T) = (\Phi(T)' \otimes \Phi(T))'$ and $c_i(T)$ be the ith component of $C(T)$. The challenge to our algorithm is to obtain a mapping Φ of the symbols, which satisfies two conditions: 1) When the symbols match, this should contribute a nonzero value in the product $\Phi(e_j) \cdot \Phi(e_{i-j})$; otherwise, it should contribute 0 and 2) the value of each component of $C(T)$ and $c_i(T) = \sum_{j=0}^{i} \Phi(e_j) \cdot \Phi(e_{i-j})$ 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 three 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 3.2.

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 $C(T)$'s 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, $c_1(T) = 2^1 + 2^{11} + 2^{14} = 18{,}434$. Fig. 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.

Fig. 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 $c_i(\bar{T}) = \sum_{j=0}^{i} 2^j \bar{e}_j \cdot \bar{e}_{i-j}$ for $i = 0, 1, \ldots, \sigma n - 1$. Thus, $C(T) = \pi_{\sigma,0}(C(\bar{T}))$. Assume that $c_p(T)$ is a nonzero 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 4 where $T = \texttt{abcabbabcb}$, $n = 10$, and $\sigma = 3$, let $s_0, s_1, s_2 = \texttt{a}, \texttt{b}, \texttt{c}$, respectively. Then, for $p = 3$, $W_3 = \{18, 16, 9, 7\}$, $W_{3,0} = \{18, 9\}$, and

$$W_{3,0,0} = \{18, 9\} \Rightarrow \mathcal{F}_2(\texttt{a}, \pi_{3,0}(T)) = 2,$$

which conforms to the results obtained previously. As another example, if $T = \texttt{cabccbacd}$, where $n = 9$, $\sigma = 4$, and $s_0, s_1, s_2, s_3 = \texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}$, respectively, then, for $p = 4$, $W_4 = \{18, 6\}$, $W_{4,2} = \{18, 6\}$,

$$W_{4,2,0} = \{18\} \Rightarrow \mathcal{F}_2(\texttt{c}, \pi_{4,0}(T)) = 1,$$

and $W_{4,2,3} = \{6\} \Rightarrow \mathcal{F}_2(\texttt{c}, \pi_{4,3}(T)) = 1$, which are correct since $\pi_{4,0}(T) = \texttt{ccd}$ and $\pi_{4,3}(T) = \texttt{cc}$.

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 neither a single-symbol pattern nor the *"don't care"* pattern, i.e., at least two s_{j_i}s are not $*$. The set $W_{p,j_i,i}$ contains the values responsible for the symbol $s_{j_i} \neq *$. Let W^p be a subset of the Cartesian product of the sets $W_{p,j_i,i}$ for all i where $s_{j_i} \neq *$ 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 $T = \texttt{abcabbabcb}$, $W_{3,0,0} = \{18, 9\}$ corresponds to the symbol a, and $W_{3,1,1} = \{16, 7\}$ corresponds to the symbol b, then for the candidate periodic pattern ab$*$, $W^p = \{(18, 16), (9, 7)\}$, 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 periodic single-symbol 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 that is performed over a binary vector of length σn. Hence, the complexity is $O(\sigma n \log \sigma n)$ when FFT is used to compute the convolution. Given that, in practice, $\sigma << n$, the complexity of the symbol periodicity detection algorithm is, in fact, $O(n \log n)$. Note that adding the coefficient 2^j to the convolution definition still preserves that

$$X \otimes Y = \text{FFT}^{-1}\Big(\text{FFT}(X) \cdot \text{FFT}(Y)\Big).$$

The complete algorithm is sketched in Fig. 2.

5 Segment Periodicity Detection

The idea behind our new algorithm for segment periodicity detection follows the same idea of shifting and comparing the time series for all possible values. However, the interest is turned to the total number of matches rather than the specific symbols that match. For example, if $T = \texttt{abcabdabc}$, then shifting T three positions results in $T^{(3)} = ***\texttt{abcabd}$. Comparing T to $T^{(3)}$ results in four matches out of six possible matches. We argue that such an occurrence of a large number of matches corresponds to a candidate period for the time series in hand. To ascertain this argument, assume that comparing T with $T^{(p)}$ results in $n - p$ matches for a certain index p and a time series T of length n. Then, $e_p = e_0, e_{p+1} = e_1, \ldots, e_{n-1} = e_{n-1-p}$ and, also, $e_{2p} = e_p, e_{2p+1} = e_{p+1}, \ldots$, etc., which means that the segment of length p is periodic and p is a *perfect* period for T. If, for another index q, comparing T with $T^{(q)}$ results in a number of matches slightly less than $n - q$ due to a few mismatches, then q can be considered a candidate period for T.

Similar to the proposed algorithm for symbol periodicity detection, our proposed algorithm for segment periodicity detection uses the concept of convolution in order to shift and compare the time series for all possible values of the period. However, segment periodicity detection needs a mapping scheme simpler than that of symbol periodicity detection since what matters is the total number of matches rather than the symbols that match. In other words, the mapping scheme Φ for segment periodicity detection should only satisfy the following condition: When the symbols match, this should contribute a nonzero value in the product $\Phi(e_j) \cdot \Phi(e_{i-j})$; otherwise, it should contribute 0.

```
Input:  A time series $T = e_0, e_1, \ldots, e_{n-1}$, and the periodicity threshold $\tau$.
Output:  Candidate periods and 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 $c_i(\bar{T}) = \sum_{j=0}^{i} 2^j \bar{e}_j \cdot \bar{e}_{i-j}$, 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 $\frac{\mathcal{F}_2(s_k, \pi_{p,l}(T))}{\lceil (n-l)/p \rceil - 1} \geq \tau$, then $p$ is a candidate period for the symbol $s_k$
         at position $l$.

     (d) Form and output the periodic single-symbol 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.
```

Fig. 2. The symbol periodicity detection algorithm.

We map the symbols to the σth roots of unity[5] [5], where σ is the size of the symbols alphabet. For example, if the time series contains only three symbols, then they are mapped to ω^0, ω^1, ω^2. Moreover, we modify the regular convolution definition to be

$$(X \otimes Y)_i = \sum_{j=0}^{i} x_j y_{i-j}^{-1}.$$

Without loss of generality, assume that a symbol s_k is mapped to ω^k. Adopting the modified convolution, a symbol match (say s_k) contributes 1 to the product $\omega^k \omega^{-k}$. A mismatch, i.e., $s_k \neq s_l$, contributes a perturbative term $\omega^{k-l} \neq 1$. Since the sum of the σth roots of unity is equal to zero, i.e., $\sum_{i=0}^{\sigma-1} \omega^i = 0$, then $E(\omega^Z) = 0$ when Z is a uniformly distributed random variable over $\{0, 1, \ldots, \sigma-1\}$. In other words, if the convolution computation is repeated for all possible mappings over $\{\omega^0, \omega^1, \ldots, \omega^{\sigma-1}\}$, then the mean value at a mismatch position will be equal to 0 and the mean value at a match position will be equal to 1, which satisfies the aforementioned condition. Therefore, we can obtain the convolution of the time series $C(T)$ as the mean of the inner product $\sum_{j=0}^{i} \Phi(e_j)\Phi^{-1}(e_{i-j})$ over all the possible mappings. Experiments show that accurate estimates can be achieved with few iterations rather than iterating over all possible mappings.

Similar to the symbol periodicity detection algorithm, the complexity of the segment periodicity detection algorithm is the complexity of the convolution step that is performed over a vector of length n, yet is repeated number of times ℓ. Hence, the complexity is $O(\ell n \log n)$ when FFT is used to compute the convolution. The maximum value for ℓ is $\sigma!$, which is the number of all possible mappings of the symbols. Clearly, this value is too large and, hence, is not negligible. However, in practice, few iterations are enough to get accurate estimates, i.e., $\ell << \sigma!$. Therefore, the practical complexity of the segment periodicity detection algorithm is $O(n \log n)$. Note that the modified convolution definition used here still preserves that

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

The complete algorithm is sketched in Fig. 3.

6 Experimental Study

This section contains the results of an extensive experimental study that examines various aspects of the proposed algorithms.[6] The most important aspect is the accuracy with respect to the discovered periods, which is the subject of

5. The nth roots of unity are the n solutions to the equation $x^n = 1$. The kth root has the form $\omega^k = e^{2\pi ik/n}$.

6. The source code of the proposed algorithms is available at http://www.cs.purdue.edu/~mgelfeky/Source.

```
Input:  A time series $T = e_0, e_1, \ldots, e_{n-1}$, and the periodicity threshold $\tau$.
Output:  Candidate periods for $T$.
Algorithm:

  1. For $\ell = 1, 2, \ldots, k$,

     (a) Randomly and uniformly select an ordering for the symbols
         $s_0, s_1, \ldots, s_{\sigma-1}$.
     (b) From $T$, obtain a complex sequence $T_\ell$ by replacing every symbol $s_k$
         by $\omega^k$, i.e., $\Phi_\ell(s_k) = \omega^k$,
     (c) Compute the sequence $C(T_\ell)$ where $c_i(T_\ell) = \sum_{j=0}^{i} \Phi_\ell(e_j)\Phi_\ell^{-1}(e_{i-j})$.

  2. Compute the vector $\hat{C}(T) = \sum_{\ell=1}^{k} C(T_\ell)/k$ and consider it as an estimate
     for $C(T)$.

  3. For $p = 1, 2, \ldots, n/2$, if $\frac{c_p(T)}{n-p} \geq \tau$, output $p$ as a candidate period for $T$.
```

Fig. 3. The segment periodicity detection algorithm.

Section 6.1. As noisy data is inevitable, Section 6.2 scrutinizes the resilience of the proposed algorithms to various types of noise that can occur in time series data. The estimation accuracy of the segment periodicity detection algorithm is studied in Section 6.3. Then, the time performance of the proposed algorithms is studied in Section 6.4. The practicality and usefulness of the results are explored using real data experiments shown in Section 6.5. The periodic trends algorithm of [15] is compared with our proposed algorithms throughout the experiments. As discussed earlier in Section 2, the periodic trends algorithm of [15] is the fastest in the literature for detecting all 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. Some types of noise include replacement, insertion, deletion, or any mixture 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 until 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.

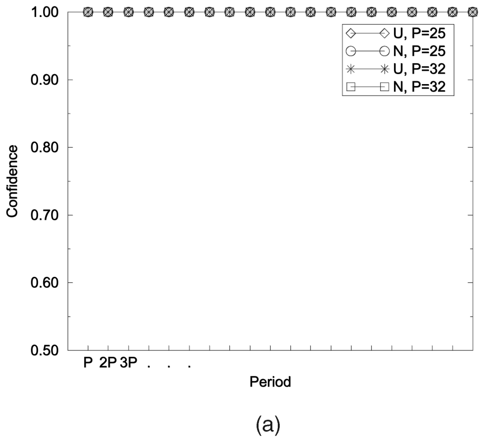
Two databases serve the purpose of real data experiments. The first one is a relatively small database that contains the daily power consumption rates of some customers over a period of one year. It is made available through the CIMEG[7] project. The database size is approximately 5 Megabytes. The second 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 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).

6.1 Accuracy

Synthetic data, both inerrant and noisy, are used in this experiment in order to inspect the accuracy of the proposed algorithms. The accuracy measure that we use is the ability of the algorithms to detect the periodicities that are artificially embedded into the synthetic data. To accurately discover a period, it is not enough to discover it at any periodicity threshold value. In other words, the periods discovered with a high periodicity threshold value are better candidates than those discovered with a lower periodicity threshold value. Therefore, we define the confidence of a discovered period to be the minimum periodicity threshold value required to detect this period. The accuracy is measured by the average confidence of all the periods that were artificially embedded into the synthetic data.

Figs. 4 and 5 give 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 periodicities embedded are: $P, 2P, \ldots$. If the data is perfectly periodic, the confidence of all the periodicities should be 1. Time series lengths of 1M symbols are used

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



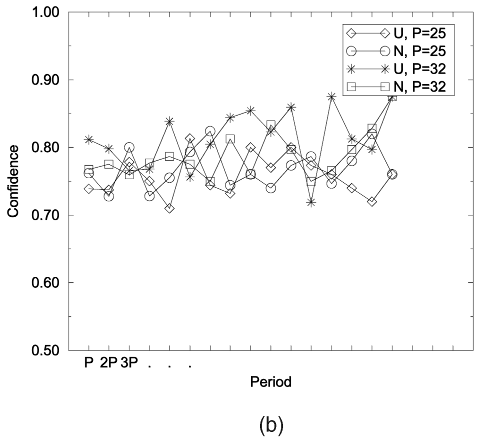
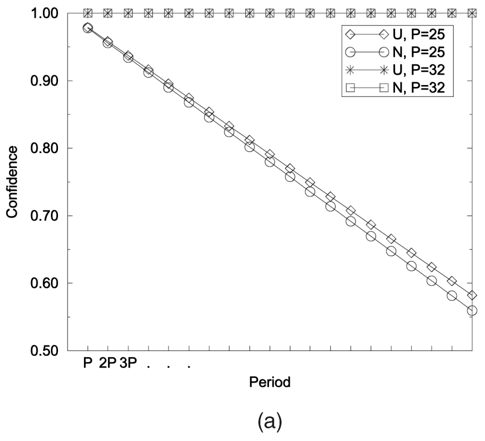


Fig. 4. Accuracy of the symbol periodicity detection algorithm. (a) Inerrant data. (b) Noisy data.



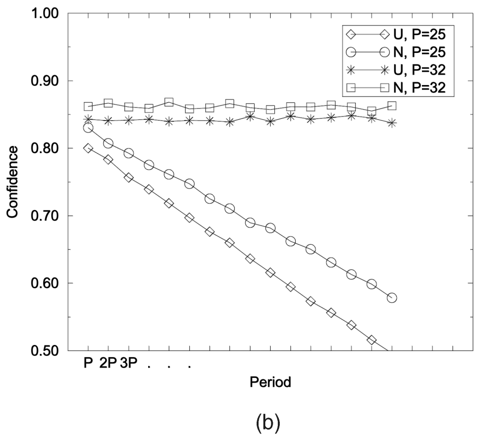
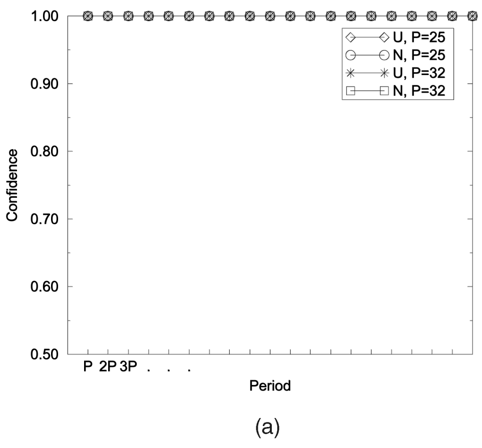


Fig. 5. Accuracy of the segment periodicity detection algorithm. (a) Inerrant data. (b) Noisy data.

with alphabet size of 10. The values collected are averaged over 100 runs. Fig. 4a shows that the symbol periodicity detection algorithm is able to detect all the embedded periodicities in the inerrant time series data with the highest possible confidence. Fig. 5a shows a similar behavior only when the period divides the time series length. In general, Fig. 5 illustrates that the behavior of the segment periodicity detection algorithm depends on the period. When the period divides the time series length, the algorithm detects all the periods at the highest confidence. When the period does not divide the time series length, the algorithm favors the lower values. This behavior is due to the misalignment of the last portion of the time series that does not form a complete periodic segment. Fig. 4 shows an unbiased behavior of the symbol periodicity detection algorithm with respect to the period since what matters are the symbols rather than the whole segment. Both figures show an expected decrease in the confidence values due to the presence of noise.

Fig. 6 gives the results of the same experiment for the periodic trends algorithm of [15]. However, in order to inspect the accuracy 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 best candidate periods. In other words, if the absolute values are sorted in ascending order, the corresponding periods will be ordered from the best to the worst candidate. Therefore, it is the rank of the period in this 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 best candidate period that has rank 1 will have normalized rank value of 1, and worse 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. Fig. 6b shows a biased behavior of the periodic trends algorithm with respect to the period, as it favors the longer periods. However, we believe that the shorter periods are more accurate than the longer 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.



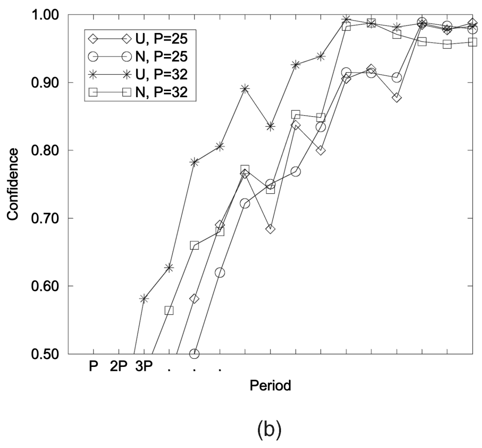
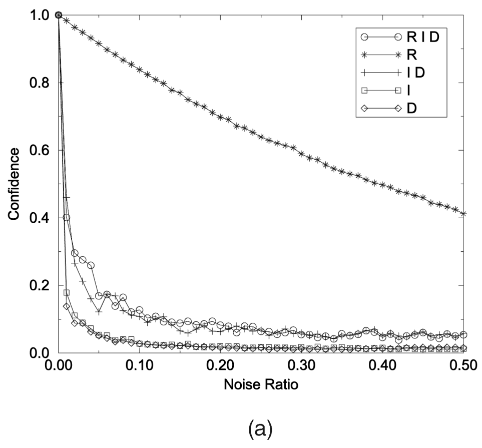


Fig. 6. Accuracy of the periodic trends algorithm. (a) Inerrant data. (b) Noisy data.



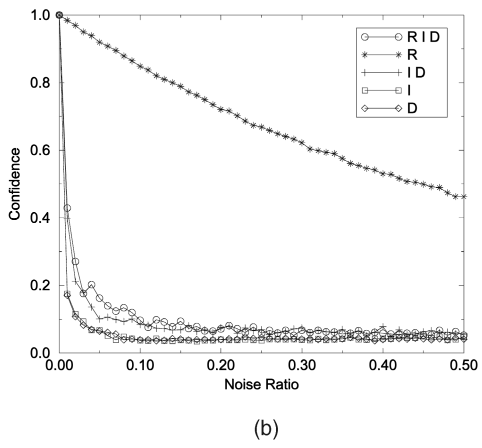
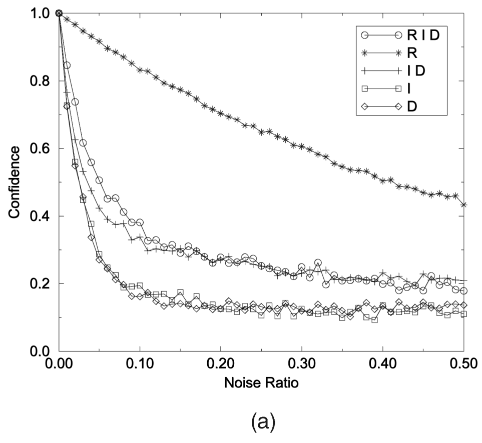


Fig. 7. Resilience to noise of the symbol periodicity detection algorithm. (a) Uniform, Period = 25. (b) Normal, Period = 32.



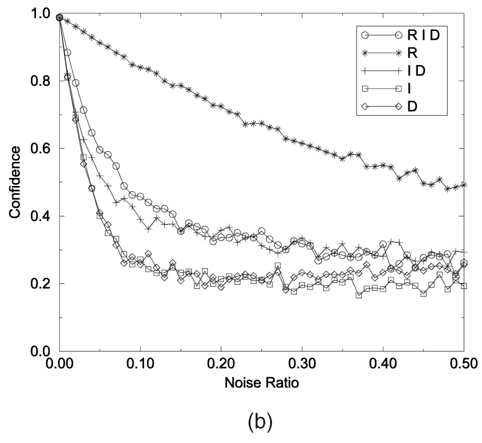


Fig. 8. Resilience to noise of the segment periodicity detection algorithm. (a) Uniform, Period = 32. (b) Normal, Period = 25.

6.2 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 periodicity detection algorithms toward these types of noise as well as different mixtures of them. Results are given in Figs. 7 and 8 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 mixed, e.g., "R I D" means that the noise ratio is distributed equally among replacement, insertion, and deletion, while "I D" means that the noise ratio is distributed equally among insertion and deletion only. Time series lengths of 1M symbols are used with an 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 arbitrary combination of

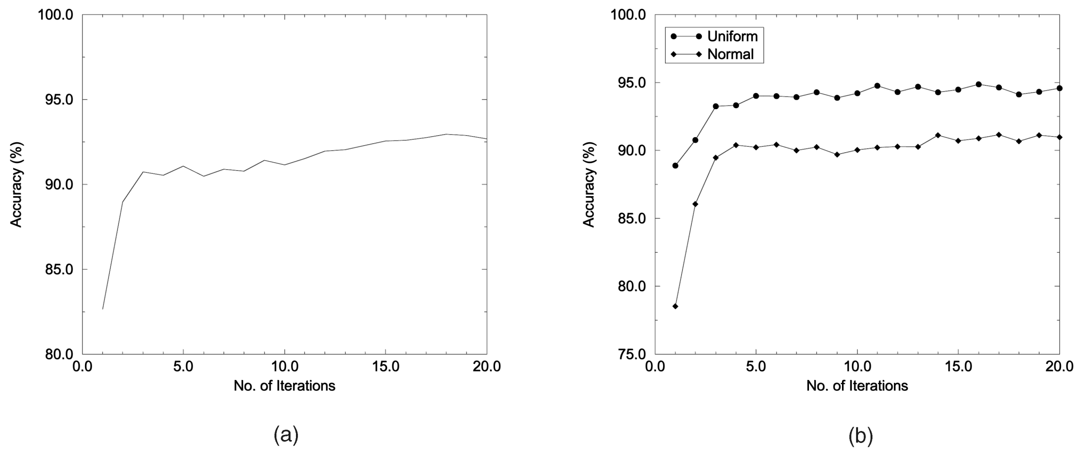


Fig. 9. Estimation accuracy of the segment periodicity detection algorithm. (a) Wal-Mart data. (b) Synthetic data.

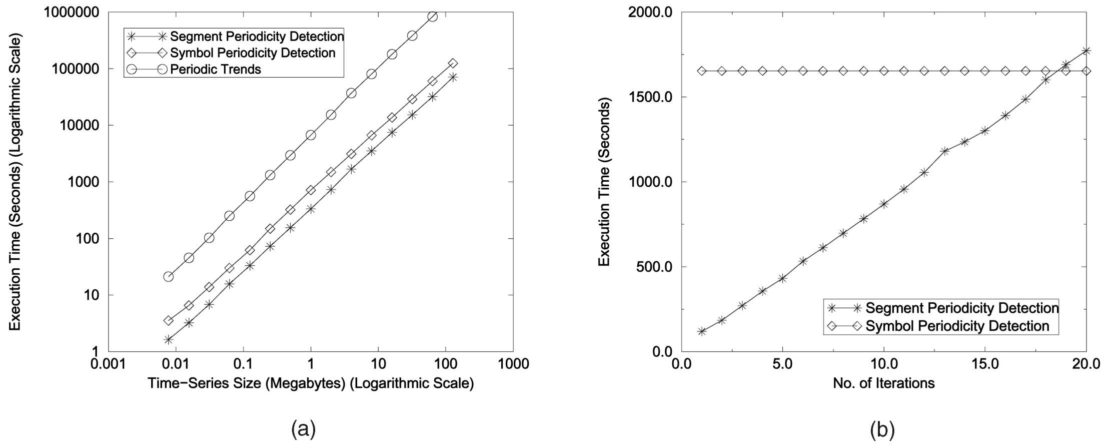


Fig. 10. Time behavior of the periodicity detection algorithms. (a) Wal-Mart data. (b) Synthetic data.

a period and a data distribution is selected for each figure. The figures show an expected decrease in the confidence with the increase in noise. Both algorithms are well resilient to replacement noise. At 40 percent periodicity thresholds, both algorithms can tolerate 50 percent replacement noise in the data. When the other types of noise get involved separately or mixed with replacement noise, the algorithms perform poorly. However, segment periodicity detection can be considered roughly resilient to those other types since periodicity thresholds in the range 10 percent to 20 percent are not uncommon.

6.3 Estimation Accuracy of Segment Periodicity

The next experiment studies the estimation accuracy of the proposed segment periodicity detection algorithm. We examine the accuracy by measuring the closeness of the convolution values estimated by the algorithm to the exact values. Fig. 9 gives the results for both real data (Wal-Mart timed sales transactions) and synthetic data (uniform and normal data). Fig. 9 shows that the estimation accuracy increases when more iterations are carried out. This is expected as the mean values converge to the exact values over all iterations. More importantly, Fig. 9 shows that an estimation accuracy of 90 percent is achieved after only two to four iterations, which supports our claim that few iterations are enough to obtain accurate estimates.

6.4 Time Performance

To evaluate the time performance of the proposed periodicity detection algorithms, Fig. 10a 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. As it has been shown in the previous experiment (Fig. 9), only three iterations of the segment periodicity detection algorithm are carried out in this experiment. Fig. 10a shows that the execution time is linearly proportional to the time series length. More importantly, the figure shows that segment periodicity detection takes less execution time than symbol periodicity detection. Fig. 10b supports this latest observation using synthetic data (1M symbols with alphabet size of 10) and increasing the number of iterations for the segment periodicity detection. Note that there are no iterations in the symbol periodicity detection algorithm. The straight line in the figure represents a fixed value for the execution time of the symbol periodicity detection algorithm. Fig. 10b shows that, up to 18 iterations, segment periodicity detection still takes less execution time than symbol periodicity detection.

TABLE 1
Segment Periodicity Detection Output

Periodicity Threshold (%)	Wal-Mart Data		CIMEG Data	
	# Output Periods	Some Output Periods	# Output Periods	Some Output Periods
50	1054	22	42	41
55	816	170	27	14
60	532	481	23	7, 63
65	286	841	10	42, 116
70	101	24, 144	3	119, 123
75	19	504, 672	1	126
80	4	168, 336, 3961, 4063	0	
85	0		0	

TABLE 2
Symbol Periodicity Detection Output

Periodicity Threshold (%)	Wal-Mart Data		CIMEG Data	
	# Output Periods	Some Output Periods	# Output Periods	Some Output 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

Furthermore, Fig. 10a shows that the proposed segment periodicity detection algorithm outperforms the periodic trends algorithm of [15] with respect to the execution time. This experimental result agrees with the theoretical results as the periodic trends algorithm [15] performs in $O(n \log^2 n)$ time whereas our proposed segment periodicity detection algorithm performs in $O(n \log n)$ time.

6.5 Real Data Experiments

Tables 1 and 2 display the output of both algorithms for the Wal-Mart and CIMEG data for different values of the periodicity thresholds. Clearly, the algorithms output fewer periods for higher periodicity threshold values and the periods detected with respect to a certain value of the periodicity threshold are enclosed within those detected with respect to a lower value. To verify their accuracy, the algorithms should at least output the periods that are expected in the time series. From Fig. 11, Wal-Mart data has an expected period of 24 that corresponds to the daily pattern of number of transactions per hour. CIMEG data has an expected period of 7 that corresponds to the weekly pattern of power consumption rates per day.

For the segment periodicity detection algorithm, Table 1 shows that, for Wal-Mart data, a period of 24 hours is detected when the periodicity threshold is 70 percent or less. In addition, the algorithm detects many more periods, some of which are quite interesting. A period of 168 hours

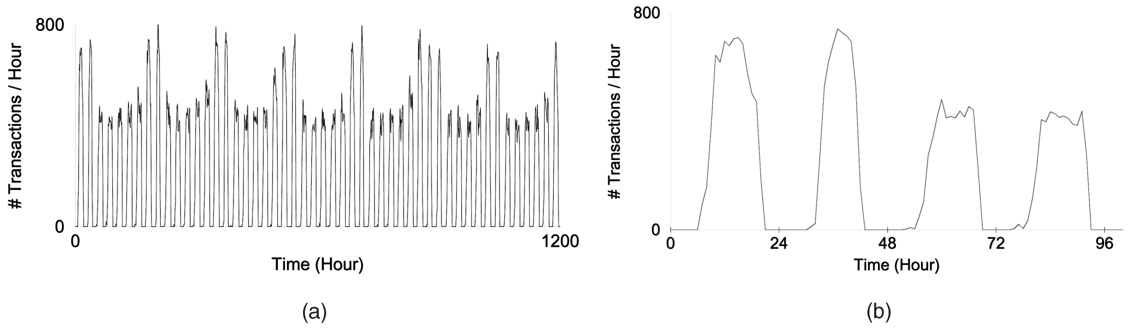


Fig. 11. Plot of Wal-Mart time series. (a) Zoomed out. (b) Zoomed in.

TABLE 3
Symbol Periodicity Detection Output (Cont'd)

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	

(24×7) can be explained as the weekly pattern of the number of transactions per hour. This period value is not easily seen in Fig. 11, yet it is there (every seven cycles). A period of 3,961 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 periodicity threshold is 60 percent or less. Other clear periods are those that are multiples of 7. However, a period of 123 days is difficult to explain. A similar behavior is shown in Table 2 for the symbol periodicity detection algorithm with a higher number of output periods than that of segment periodicity detection. This is expected since symbol periodicity detects periods for individual symbols rather than for the entire time series. Further analysis of the output of the symbol periodicity detection algorithm is shown in Table 3.

Exploring the period of 24 hours for Wal-Mart and that of 7 days for CIMEG data produces the results given in Table 3. Note that periodic single-symbol 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 percent or less represents the periodic single-symbol pattern $*******\mathtt{b}****************$. 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 follows:

In 80 percent of the days, less than 200 transactions per hour occur in the seventh hour of the day (between 7:00 a.m. and 8:00 a.m.).

As another example, $(\mathtt{a}, 3)$ for CIMEG data with respect to a periodicity threshold of 50 percent or less represents the periodic single-symbol pattern $***\mathtt{a}***$. Knowing that the symbol a represents the *very low* level for CIMEG data (less than 6,000 Watts/Day), this periodic pattern can be interpreted as follows:

In 50 percent of the weeks, less than 6,000 Watts/Day are consumed in the fourth day of the week.

Finally, Table 4 gives the final output of periodic patterns of Wal-Mart data for the period of 24 hours for periodicity threshold of 35 percent. Each pattern can be interpreted in a similar way to the above.

Experimenting the periodic trends algorithm of [15] with the real data gives a worse behavior than that of the proposed segment periodicity detection. The known a priori periods (24 for Wal-Mart data and 7 for CIMEG data) are discovered only after very low periodicity threshold values (<30 percent) are considered. The reason is that the periodic trends algorithm favors the longer periods and, so, the shorter periods appear late in the candidate order. For example, for Wal-Mart data, the periodic trends algorithm detects that a period of 2,808 hours (a multiple of 24) is the best candidate period; and, for CIMEG data, it detects that a period of 3,708 days (not a multiple of 7) is the best candidate period.

7 Conclusions

In this paper, we have defined two types of periodicities for time series databases. Whereas symbol periodicity addresses the periodicity of the symbols in the time series, segment periodicity addresses the periodicity of the entire time series regarding its segments. We have proposed a scalable, computationally efficient algorithm for detecting each type of periodicity in $O(n \log n)$ time, for a time series of length n. An empirical study of the algorithms using real-world and synthetic data sets proves the practicality of the problem, validates the accuracy of the algorithms, and validates the usefulness of their outputs. Moreover, segment periodicity detection takes less execution time

TABLE 4
Obscure Periodic Patterns for Wal-Mart Data

Obscure Periodic Pattern	Support (%)
$\mathtt{aaaa}*****************\mathtt{aaa}$	49.78166
$\mathtt{aaaa}***\mathtt{b}*************\mathtt{aaa}$	42.57642
$\mathtt{aaaa}***\mathtt{b}*********\mathtt{d}***\mathtt{aaa}$	38.56768
$*****\mathtt{bbbbc}************\mathtt{aa}$	35.80786

whereas symbol periodicity detects more periods. We can conclude that in practice, segment periodicity detection could be applied first and, if the results are not sufficient, or not appealing, symbol periodicity detection can be applied afterwards. Finally, we have extended the proposed symbol periodicity detection algorithm to discover the obscure periodic patterns.

REFERENCES

[1] K. Abrahamson, "Generalized String Matching," *SIAM J. Computing,* vol. 16, no. 6, pp. 1039-1051, 1987.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," *Proc. 20th Int'l Conf. Very Large Data Bases,* Sept. 1994.

[3] R. Agrawal and R. Srikant, "Mining Sequential Patterns," *Proc. 11th Int'l Conf. Data Eng.,* Mar. 1995.

[4] W. Aref, M. Elfeky, and A. Elmagarmid, "Incremental, Online, and Merge Mining of Partial Periodic Patterns in Time-Series Databases," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 3, pp. 332-342, 2004.

[5] M. Atallah, F. Chyzak, and P. Dumas, "A Randomized Algorithm for Approximate String Matching," *Algorithmica,* vol. 29, no. 3, pp. 468-486, 2001.

[6] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, "Sequential Pattern Mining Using A Bitmap Representation," *Proc. Eighth Int'l Conf. Knowledge Discovery and Data Mining,* July 2002.

[7] C. Berberidis, W. Aref, M. Atallah, I. Vlahavas, and A. Elmagarmid, "Multiple and Partial Periodicity Mining in Time Series Databases," *Proc. 15th European Conf. Artificial Intelligence,* July 2002.

[8] C. Bettini, X. Wang, S. Jajodia, and J. Lin, "Discovering Frequent Event Patterns with Multiple Granularities in Time Sequences," *IEEE Trans. Knowledge and Data Eng.,* vol. 10, no. 2, pp. 222-237, 1998.

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

[10] C. Daw, C. Finney, and E. Tracy, "A Review of Symbolic Analysis of Experimental Data," *Rev. Scientific Instruments,* vol. 74, no. 2, pp. 915-930, 2003.

[11] M. Elfeky, W. Aref, and A. Elmagarmid, "Using Convolution to Mine Obscure Periodic Patterns in One Pass," *Proc. Ninth Int'l Conf. Extending Data Base Technology,* Mar. 2004.

[12] M. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints," *Proc. 25th Int'l Conf. Very Large Data Bases,* Sept. 1999.

[13] J. Han, G. Dong, and Y. Yin, "Efficient Mining of Partial Periodic Patterns in Time Series Databases," *Proc. 15th Int'l Conf. Data Eng.,* Mar. 1999.

[14] J. Han, W. Gong, and Y. Yin, "Mining Segment-Wise Periodic Patterns in Time Related Databases," *Proc. Fourth Int'l Conf. Knowledge Discovery and Data Mining,* Aug. 1998.

[15] P. Indyk, N. Koudas, and S. Muthukrishnan, "Identifying Representative Trends in Massive Time Series Data Sets Using Sketches," *Proc. 26th Int'l Conf. Very Large Data Bases,* Sept. 2000.

[16] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting Time Series: A Survey and Novel Approach," *Data Mining in Time Series Databases,* M. Last, A. Kandel, and H. Bunke, eds., World Scientific Publishing, June 2004.

[17] E. Keogh, S. Lonardi, and B. Chiu, "Finding Surprising Patterns in a Time Series Database in Linear Time and Space," *Proc. Eighth Int'l Conf. Knowledge Discovery and Data Mining,* July 2002.

[18] D. Knuth, *The Art of Computer Programming,* vol. 2, second ed., series in computer science and information processing, Reading, Mass.: Addison-Wesley, 1981.

[19] S. Ma and J. Hellerstein, "Mining Partially Periodic Event Patterns with Unknown Periods," *Proc. 17th Int'l Conf. Data Eng.,* Apr. 2001.

[20] B. Ozden, S. Ramaswamy, and A. Silberschatz, "Cyclic Association Rules," *Proc. 14th Int'l Conf. Data Eng.,* Feb. 1998.

[21] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proc. Fifth Int'l Conf. Extending Data Base Technology,* Mar. 1996.

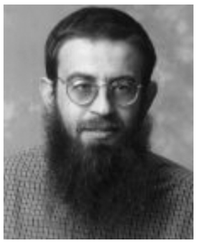
[22] J. Vitter, "External Memory Algorithms and Data Structures: Dealing with Massive Data," *ACM Computing Surveys,* vol. 33, no. 2, pp. 209-271, June 2001.

[23] A. Weigend and N. Gershenfeld, *Time Series Prediction: Forecasting the Future and Understanding the Past.* Reading, Mass.: Addison-Wesley, Reading, 1994.

[24] J. Yang, W. Wang, and P. Yu, "Mining Asynchronous Periodic Patterns in Time Series Data," *Proc. Sixth Int'l Conf. Knowledge Discovery and Data Mining,* Aug. 2000.

[25] J. Yang, W. Wang, and P. Yu, "InfoMiner +: Mining Partial Periodic Patterns with Gap Penalties," *Proc. Second Int'l Conf. Data Mining,* Dec. 2002.

Mohamed G. Elfeky received the BSc and MSc degrees in computer science from Alexandria University, Egypt, in 1996 and 1999, respectively. He received the PhD degree in computer science from Purdue University in 2005. His current research interests include data mining, data quality, and time series databases.



Walid G. Aref is an associate professor of computer science at Purdue. His research interests are in developing database technologies for emerging applications, e.g., spatial, multimedia, genomics, and sensor-based databases. He is also interested in indexing, data mining, scalable media servers, and geographic information systems (GIS). Professor Aref's research has been supported by the US National Science Foundation (NSF), the Purdue Research Foundation, CERIAS, Panasonic, and Microsoft Corp. In 2001, he received the CAREER Award from NSF, and in 2004, he was selected as a Purdue University Faculty Scholar. Professor Aref is a member of the ACM, a member of the IEEE Computer Society, and a senior member of the IEEE.

Ahmed K. Elmagarmid received the BS degree in computer science from the University of Dayton and the MS and PhD degrees from The Ohio State University in 1977, 1981, and 1985, respectively. He received a Presidential Young Investigator award from the National Science Foundation and distinguished alumni awards from the Ohio State University and the University of Dayton in 1988, 1993, and 1995, respectively. Professor Elmagarmid is the editor-in-chief of *Distributed and Parallel Databases: An International Journal* and of the book series on advances in database systems, and serves on the editorial boards of the *IEEE Transactions on Knowledge and Data Engineering*, *Information Sciences*, and the *Journal of Communications Systems*. He has served on the editorial boards of the *IEEE Transactions on Computers* and the *IEEE Data Engineering Bulletin*. He is on the steering committees for the IEEE International Conference on Data Engineering and the IEEE Symposium on Research Issues in Data Engineering, and has served on the organization committees of several international conferences. Professor Elmagarmid is the director of the Indiana Center for Database Systems (ICDS) and the newly formed Indiana Telemedicine Incubator. His research interests are in the areas of video databases, multidatabases, data quality, and their applications in telemedicine and digital government. He is the author of several books in databases and multimedia. He was chief scientist for Hewlett-Packard from 2001 to 2003 while on leave from Purdue. He has served widely as an industry consultant and/or adviser to Telcordia, Harris, IBM, MCC, UniSql, MDL, BNR, etc. He served as a faculty member at the Pennsylvania State University from 1985-1988 and has been with the Department of Computer Science at Purdue University since 1988. He is a senior member of the IEEE and a member of the IEEE Computer Society.