2002

Periodicity Detection in Time Series Databases

Mohamed G. Elfeky

Walid G. Aref
Purdue University, aref@cs.purdue.edu

Mikhail J. Atallah
Purdue University, mja@cs.purdue.edu

Report Number:

02-029

PERODICITY DETECTION IN TIME SERIES DATABASES

Mohamed G, Ekfet
Walid G. Aref
Mikhail J. Atallah

Purdue University
Department of Computer Sciences
West Lafayette, IN 47907

CSD TR #02-029
December 2002

Periodicity Detection in Time Series Databases *

Mohamed G. Elfeky
Purdue University
mgelfeky@cs.purdue.edu

Walid G. Aref
Purdue University
aref@cs.purdue.edu

Mikhail J. Atallah
Purdue University
mja@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. Discovering the rate at which the time series is periodic has always been an obstacle for fully automated periodicity mining. In existing periodic patterns mining algorithms, the period length is user-specified. This is a considerable disadvantage, especially in time series datasets where the period length 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 and computationally efficient algorithm is proposed for each type. The algorithms require only one scan over a time series database of size n, with $O(n \log n)$ time complexity.

1. INTRODUCTION

"The only reason for time is so that everything doesn't happen at once" - Albert Einstein

A time series database is one that abounds with data evolving over time. Life embraces several examples of time series data, e.g., meteorological data containing several measurements (temperature, humidity, etc.), stock prices depicted in financial market, power consumption data reported in energy corporations, etc. 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. Agrawal et al. [2] develop a model for similarity of time sequences that can be used for mining periodic patterns. In [3], Agrawal et al. define a shape definition language for retrieving user-specified shapes contained in histories (time series data). An Apriori-like [4] technique for mining sequential patterns is presented in [5, 20], which is then extended by Garofalakis et al. [12]. In [9], Bettini et al. develop algorithms for discovering temporal patterns. Ozden et. al. [19] study mining of association rules of periodic nature. Han et al. [14, 13] introduce the notion of partial periodic patterns and present two algorithms for mining this type of patterns. That work has been extended by Yang et al. [22] to account for the intervention of random noise, and by Aref et al. [6] to account for the incremental nature of time series data.

The periodicity mining techniques described in the above research require the user to specify a period length that determines the rate at which the time series is periodic. Hence, these techniques assume that users either know the length of the period beforehand or that they are willing to try various period length 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. Moreover, even in the case of time series data with a priori known period lengths, there may be obscure periods, and consequently interesting patterns, that cannot be easily discovered. The solution to these problems is to devise a technique for discovering potential periods in the time series data, followed by the application of any existing pattern mining technique to determine the interesting patterns. A trial to handle the unknown period problem for partial periodic patterns mining has been presented in [18]. However, the algorithm misses a lot of correct periods, and it generates many false positives.

In this paper, we address the problem of discovering potential periods for time series databases, hereafter referred to as *periodicity detection*. We define two types of periodicities: *segment periodicity* and *symbol periodicity*. For each periodicity type, an efficient algorithm is proposed and is analyzed, both theoretically and empirically. Although segment periodicity has been addressed in [16] under the name *periodic trends*, to our knowledge, no previous research has introduced the notion of symbol periodicity. Moreover, the algorithm presented in [16] performs in $O(n \log^2 n)$ time, where n is the size of the time series. Our proposed algorithms perform in $O(n \log n)$ time for both segment and symbol periodicities.

The structure of the rest of the paper is as follows. In Section 2, we introduce the notation that is used throughout the

*This work has been supported in part by the Natioanl Science Foundation under grants IIS-0093116, EIA-9972883, IIS-0209120, by the Purdue Research Foundation, by the NAVSEA/Naval Surface Warfare Center, Crane, and by grants from NCR and Wal-Mart.

paper, and we formally define the periodicity detection problem and the segment and symbol periodicity types. Sections 3 and 4 describe the two proposed algorithms for periodicity detection in time series databases. In Section 5, the performance of the algorithms is studied. A further discussion is given in Section 6, and we summarize our conclusions in Section 7.

2. PERIODICITY DETECTION PROBLEM

2.1 Notation

Assume that a sequence of n timestamped feature values is collected in a time series database. For 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, and the feature in a time series database for stock prices might be the final daily stock price of a specific company. If we quantize the time series feature values into 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.

The time series database may also be a sequence of n timestamped events drawn from a finite set of 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 Segment Periodicity

A time series T is said to be periodic with a period of length p if it can be divided into equal length segments, each of length p, that are "almost" similar. For example, the time series $T = abcabcabc$ is clearly periodic with a period of length 3. Likewise, the time series $T = abcabdabc$ is periodic with a period of length 3 despite the fact that its second segment is not identical to the others. A simple way to measure the similarity between two segments is the hamming distance between them, defined as follows:

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

where u, v are two segments of equal length m, and u_j, 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 1. If a time series T of length n can be sliced into segments $T_0, T_1, \ldots, T_i, \ldots, T_N$, where $T_i = t_{ip}, \ldots, t_{ip+p-1}$, $N = \lfloor n/p \rfloor - 1$, and $S(T_i, T_j) \geq \varepsilon \quad \forall i, j = 0, 1, \ldots, N$ where $0 < \varepsilon \leq 1$, then T is said to be periodic with a period of length p with respect to a *segment similarity threshold* equal to ε.

Remark. 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 $T = abcabcabc$ has a perfect period of length 3, and the time series $T = abcabdabc$ is periodic with a period of length 3 with respect to a segment similarity threshold $\varepsilon \leq 2/3$.

2.3 Symbol Periodicity

In a time series T, a symbol s is said to be periodic with a period of length p if it exists "almost" every p timestamps. For example, in the time series $T = abcabbabcb$, the symbol b is periodic with a period of length 4, since it exists every four timestamps (positions 1, 5 and 9). Moreover, the symbol a is periodic with a period of length 3, since it exists almost every three timestamps (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. This ratio, however, 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 a period of length 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 $\mathcal{F}(s,T)$ denote the number of times the symbol s occurs in two consecutive positions in the time series T. For example, if $T = abbaaabaa$, then $\mathcal{F}(a,T) = 3$ and $\mathcal{F}(b,T) = 1$.

Definition 2. If a time series T of length n contains a symbol s such that $\exists l, p$ where $l < p$, and $\frac{\mathcal{F}(s,\pi_{p,l}(T))}{\lceil (n-l)/p \rceil - 1} \geq \psi$ where $0 < \psi \leq 1$, then s is said to be periodic in T with a period of length p with respect to a *symbol periodicity threshold* equal to ψ.

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

Symbol periodicity can be considered a generalization of segment periodicity. To show that, let us consider the outputs of their corresponding definitions. Segment periodicity determines candidate periods for the time series as a whole, thus it outputs period length values. Symbol periodicity determines the candidate symbols *and* their candidate periods, thus it outputs symbols and period length values for each symbol. Therefore, symbol periodicity output cannot be a subset of segment periodicity output since it contains not only period length values, but also symbols. On the other hand, consider the example $T = abcaababc$, whose segment periodicity output does not include a period of length 3 unless the segment similarity threshold is less than 1/3. Symbol periodicity output, however, includes a period of length 3 for the symbol a no matter how large the value of the symbol periodicity threshold. The importance of segment periodicity is that it examines the time series in less time and produces much less output, as will be shown in Section 5.

3. SEGMENT PERIODICITY DETECTION

The idea behind our algorithm is that segment periodicity detection can be viewed as a variation of the *approximate string matching* problem, which considers finding occurrences of small variations of a pattern string P of length m in a text string T of length n [7]. Approximate string matching is solved by computing a score vector of matches between T and P. This is defined as the vector $C^{T,P}$ whose ith component $c_i^{T,P}$ is the number of matches between the text and the pattern when the first letter of the pattern is positioned in front of the ith letter of the string, where $i = 0, 1, \ldots, n-1$. For example, for a text string $T = abcdaccd$ and a pattern string $P = bcd$, the score vector will be $C^{T,P} = [0,3,0,0,1,2,0,0]$, i.e., $c_1^{T,P} = 3$ and $c_5^{T,P} = 2$, which correspond to a perfect match of the pattern at position 1 of the text, and an approximate match at position 5, respectively. To see how approximate string matching can be tailored to fit segment periodicity detection, we let $P = T$ in the previous example, i.e., $P = T = abcdaccd$. Then $C^{T,T} = [8,1,0,1,3,0,0,0]$, i.e., $c_4^{T,T} = 3$, which implies that position 4 is a good candidate for matching T with T (ignoring position 0 which matches entirely). Noting that text T actually has a candidate period of length 4, we suggest that those candidate positions for matching T with T are good estimators for period lengths for the text T.

Let C^T denote the score vector of matches between a string T and itself, shortened as the score vector of matches for the string T. We formally introduce the following segment periodicity property followed by some observations.

Segment Periodicity Property. Let C^T be the score vector of matches for a time series T of length n. If $\exists p$ such that $\frac{c_p^T}{n-p} \geq \varepsilon$, where $p = 1, 2, \ldots, n/2$, and ε is the segment similarity threshold, then p is a candidate period for the time series T.

Observation 1. The values $1, 2, \ldots, n/2$ are the only possible values for candidate periods for T.

Observation 2. The maximum value for any c_i^T is $n - i$.

Observation 3. If p is a perfect period for T, then $c_p^T = n - p$.

To ascertain this property, we first examine the third observation. Let $c_p^T = n - p$ for a certain index p for a time series T. This implies that $t_p = t_0, t_{p+1} = t_1, \ldots, t_{n-1} = t_{n-1-p}$, and also, $t_{2p} = t_p, t_{2p+1} = t_{p+1}, \ldots$, and so on, which means that the segment of length p is periodic and p is a perfect period for T. If for another index q, c_q^T is slightly less than $n - q$ due to a few mismatches, then q can be considered a candidate period for T. That is the segment periodicity property.

Note that this property does not strictly comply with the definition of segment periodicity. However, it allows us to devise an algorithm of complexity $O(n \log n)$ for segment periodicity. Strict compliance with the definition would require measurements for similarity between each two segments, resulting in complexity $O(n^2)$.

3.1 The Intuition

A naïve approach for computing the score vector of matches for a string of length n is the deterministic algorithm, with a time complexity of $O(n^2)$. Needless to say, this algorithm is not scalable and is very difficult to convert to an external algorithm to handle large datasets. Rather than focusing on computing the exact values of the score vector, we propose a *randomized* algorithm of Monte-Carlo type [7] to compute an unbiased estimate of the score vector. Although randomized, its behavior does not depend on any a priori probabilistic assumptions on the input, nor does it depend on the size of the alphabet. It proceeds by computing and averaging k independent equally distributed estimates for the score vector. The expected value of the averaged estimator is equal to the exact value. In other words, the expected value of the ith component $\hat{c}_i$ of our estimate score vector $\hat{C}$ equals c_i.

The algorithm computes the score vector by a convolution, which makes it possible to use the fast Fourier transform (FFT) [17]. Therefore, we achieve two key advantages. First, the time complexity is reduced to $O(n \log n)$, where $n \log n$ is the time needed to perform the convolution of two vectors of length n by FFT, where an arithmetic operation takes constant time. Second, the algorithm is scalable since an external FFT algorithm [21] can be adopted for large datasets. It is worth mentioning that fast Fourier transform has been widely used as a similarity matching technique for time series databases [1, 11].

3.2 Description of the Algorithm

Assume that we have two strings of length n over a finite alphabet Σ of cardinality σ. If we renumber the letters by applying a map Φ from the alphabet to the integer interval $[0, \sigma) = \{0, 1, \ldots, \sigma - 1\}$, we obtain two integer sequences $x_0, x_1, \ldots, x_{n-1}$ and $y_0, y_1, \ldots, y_{n-1}$. A match between the two strings at position j, i.e., $x_j = y_j$ contributes 1 in the inner product $\sum_{j=0}^{n-1} \omega^{x_j} \bar{\omega}^{y_j} = \sum_{j=0}^{n-1} \omega^{x_j - y_j}$, where ω denotes any primitive σth root of unity[1]. A mismatch, $x_j \neq y_j$, contributes a perturbative term $\omega^{x_j - y_j}$.

Since the sum of the σth roots of unity is equal to zero, $\sum_{i=0}^{\sigma-1} \omega^i = 0$, one observes that the mean $E(\omega^X)$ is zero when X is a uniformly distributed random variable over $[0, \sigma)$. Consequently, we introduce the set Ξ of all possible mappings from Σ to $[0, \sigma)$, and turn Φ into a uniformly distributed random variable over Ξ so as to obtain the score between both strings as the mean of the inner product $\sum_{j=0}^{n-1} \omega^{x_j} \bar{\omega}^{y_j} = \sum_{j=0}^{n-1} \omega^{x_j - y_j}$ over all renumberings. Therefore, the main idea of our algorithm is to interpret the score vector as the mean over all renumberings of the convolution of two randomized finite sequences of complex numbers. In this way, the simultaneous calculation of all the components of the score vector is made possible by the use of FFT.

A convolution [15] 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

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

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

[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 take place between equal length sequences.

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

Output: candidate period lengths for $T$.

Algorithm:

  1. For $\ell = 1, 2, \ldots, k$,
      (a) randomly and uniformly select a $\Phi^{(\ell)}$ from
          $\Xi = [0, \sigma)^{\Sigma}$,
      (b) from $T$, obtain a complex sequence $T_1^{(\ell)}$ by
          replacing every symbol $t$ by $\omega^{\Phi^{(\ell)}(t)}$,
      (c) from $T$, obtain a complex sequence $T_2^{(\ell)}$ by
          replacing every symbol $t$ by $\omega^{-\Phi^{(\ell)}(t)}$,
      (d) compute the vector $C^{T^{(\ell)}}$ as the convolution
          of $T_1^{(\ell)}$ with the reverse of $T_2^{(\ell)}$, i.e.,
          $c_i^{T^{(\ell)}} = \sum_{j=0}^{n-1} \omega^{\Phi^{(\ell)}(t_{i+j}) - \Phi^{(\ell)}(t_j)}$.
  2. Compute the vector $\hat{C}^T = \sum_{\ell=1}^{k} C^{T^{(\ell)}}/k$ and
     consider its reverse as an estimate for $C^T$.
  3. For $p = 1, 2, \ldots, n/2$, if $\frac{c_p^T}{n-p} \geq \varepsilon$, output $p$ as a
     candidate period length for $T$.
```

Figure 1: Segment Periodicity Detection Algorithm

$\sum_{j=0}^{n-1-i} x_{n-1-j} y_{n-1-i-j}$, 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.

Therefore, our algorithm converts the time series into two finite sequences of complex numbers, reverses one of them, and then performs the convolution between them in order to obtain an estimate of the score vector. It is well known that convolution can be calculated by FFT as follows $x \otimes y = \text{FFT}^{-1}(\text{FFT}(x) \cdot \text{FFT}(y))$ [10]. The algorithm is sketched in Figure 1.

4. SYMBOL PERIODICITY DETECTION

Symbol periodicity detection can be solved easily using the same segment periodicity detection algorithm, shown in Figure 1, applied for each symbol separately. Only one symbol is considered at a time and all other symbols are replaced by a *"don't care"* symbol §. For example, if $T = abcdaccd$, when the symbol c is considered, T is converted to $\bar{T} =$ §§c§§cc§. One observes that with only two symbols, one of which is meaningless, there is no need for complex root multiplication. The main objective of using complex root multiplication was to obtain 1 when the symbols match, and a perturbed value otherwise. The mean value of this perturbed value is 0 when the process is repeated for several uniformly distributed mappings of the symbols. When only one symbol is considered at a time, replacing this symbol by 1 and the § symbol by 0 will contribute 1 when the symbol matches and 0 otherwise, leading to an exact calculation of the score vector of $\bar{T}$ without the need for several repetitions. We have already proposed this idea in [8]. The disadvantage of this algorithm is that it requires σ scans over the time series, one per symbol, which is very costly. In this paper, we propose a more efficient algorithm for symbol periodicity detection problem that has a complexity of $O(n \log n)$, and requires only one scan over the time series in order to detect all symbol periodicities.

4.1 Description of the Algorithm

Let $T = t_0, t_1, \ldots, t_{n-1}$ be a time series of length n, where the t_i's are symbols from a finite alphabet Σ of cardinality σ. The main challenge of our one-scan algorithm is to obtain a mapping Φ of the symbols that satisfies two conditions: (i) when the symbols match, this should contribute a non-zero value in the product $\Phi(t_{i+j}) \cdot \Phi(t_j)$, otherwise it should contribute 0, and (ii) the value of each component of the score vector, $c_i^T = \sum_{j=0}^{n-1} \Phi(t_{i+j}) \cdot \Phi(t_j)$, should identify the symbols that caused the occurrence of this value. Note that this latter condition is not important in the approximate string matching discussed in Section 3.1, since the main concern of approximate string matching is to determine number of matches at each position no matter which symbols match.

We map the symbols to the binary representation of increasing powers of two. For example, if the time series contains only 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, if $T = acccabb$, then it is converted to the binary vector $\bar{T} = 001100100100001010010$. Adopting regular convolution, defined previously, results in a score vector $C^{\bar{T}}$ 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 the required score vector C^T. The latter derivation of C^T can be written as $C^T = \pi_{\sigma,0}(C^{\bar{T}})$.

The first condition is satisfied since the only way to obtain a value of 1 contributing to a component of $C^{\bar{T}}$ 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 matched with itself at position 1. Those matches are seen from manual inspection of T to be two c's and one b, nevertheless, it is not possible to determine those matches only by examining the value of c_1^T. Therefore, the second condition is not yet satisfied. Thus, the proposed symbol periodicity detection algorithm uses a slightly modified version of the convolution definition introduced in Section 3.3, that is $(x \otimes y)_i = \sum_{j=0}^{i} 2^j x_j y_{i-j}$. 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 2 illustrates this calculation. The powers of 2 for this value are 1, 11, and 14. Examining those powers modulo 3, which is the cardinality of the alphabet in this particular example, results in 1, 2 and 2, respectively, which correspond to the symbols b, c, and c, respectively.

Figure 2 shows another example for c_4^T containing only one power of 2, which is 6, that corresponds to the symbol a since 6 mod 3 = 0 and a was originally mapped to the

```
$T$:                      0 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0 1 0 0 1 0
$\bar{T}$ shifted 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
$\bar{T}$ shifted 12 :                            0 0 1 1 0 0 1 0 0
$c_{12}^{\bar{T}} = c_4^T =$                          $2^6$
```

Figure 2: A Clarifying Example

binary representation of 2^0. This means that matching T with itself staring at position 4 results in only one match of the symbol a. Moreover, the power value of 6 reveals that the symbol a is in position 0 in the shifted T. 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 score vector $C^{\bar{T}}$ is computed such that $c_i^{\bar{T}} = \sum_{j=0}^{n-1} 2^j \bar{t}_{i+j} \cdot \bar{t}_j$ 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 matched with itself at position 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 $\mathcal{F}(s_k, \pi_{p,l}(T))$. Working out the example of Section 2.3 where $T = abcabbabcb$, $n = 10$, and $\sigma = 3$, let $s_0, s_1, s_2 = a, b, c$, respectively. Then, for $p = 3$, $W_3 = \{18, 16, 9, 7\}$, $W_{3,0} = \{18, 9\}$, $W_{3,0,0} = \{18, 9\} \Rightarrow \mathcal{F}(a, \pi_{3,0}(T)) = 2$ 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 $p = 4$, $W_4 = \{18, 6\}$, $W_{4,2} = \{18, 6\}$, $W_{4,2,0} = \{18\} \Rightarrow \mathcal{F}(c, \pi_{4,0}(T)) = 1$, and $W_{4,2,3} = \{9\} \Rightarrow \mathcal{F}(c, \pi_{4,3}(T)) = 1$ which are correct since $\pi_{4,0}(T) = ccd$ and $\pi_{4,3}(T) = cc$.

Therefore, our algorithm scans the time series once to convert it into a binary vector according to the defined mapping, performs the modified convolution on the binary vector in order to obtain its score vector, and analyzes this score vector so that the symbol periodicities can be output. The complexity of the 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 $x \otimes y = \mathrm{FFT}^{-1}(\mathrm{FFT}(x) \cdot \mathrm{FFT}(y))$. Although one might assume that the algorithm requires handling of large numbers $O(2^{\sigma n})$, careful implementation using bit structures and operations will avoid such complexity. The complete algorithm is sketched in Figure 3.

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

Output: candidate symbol periodicities for $T$ (a
symbol and a period length).

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 vector $C^{\bar{T}}$ where $c_i^{\bar{T}} = \sum_{j=0}^{n-1} 2^j \bar{t}_{i+j} \cdot \bar{t}_j$,
     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 $F(s_k, \pi_{p,l}(T))$.
       (c) if $\frac{F(s_k, \pi_{p,l}(T))}{\lceil (n-l)/p \rceil - 1} \geq \psi$, output $p$ as a candidate
           period length for the symbol $s_k$.
```

Figure 3: Symbol Periodicity Detection Algorithm

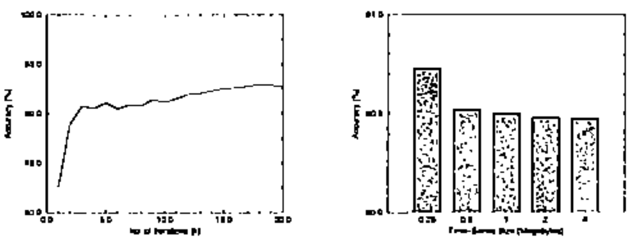
5. EXPERIMENTAL STUDY

This section contains the results of an extensive experimental study that examines the proposed algorithms for different aspects. In Section 5.1, experiments are conducted to examine the time performance as well as the accuracy and the correctness of the proposed algorithms. As data inerrancy is inevitable, Section 5.2 scrutinizes the resilience of the algorithms 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 5.3.

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 length, alphabet size, type, and amount of noise. Both uniform and normal data distributions are considered. Types of noise include replacement, insertion, deletion noise, or any combination of them. Inerrant data is generated by repeating a pattern, of length equals to the period length, that is randomly generated from the specified data distribution. The pattern is repeated till it spans the required data size. Noise is introduced randomly and uniformly over the whole time series. Whereas 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

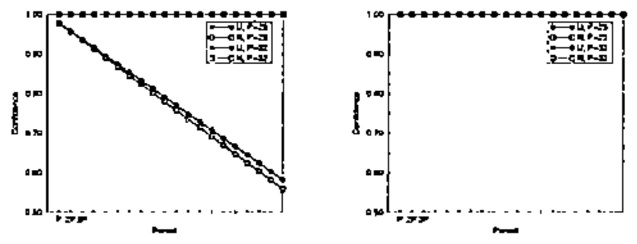
[3]CIMEG: Consortium for the Intelligent Management of the Electric Power Grid. http://helios.ecn.purdue.edu/~cimeg.



(a) Time Series Size = 1 Megabytes

(b) No. of Iterations = 3

Figure 4: Accuracy of Segment Periodicity Detection Algorithm



(a) Segment Periodicity Detection

(b) Symbol Periodicity Detection

Figure 5: Correctness of Periodicity Detection Algorithms

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 quantized into five levels, i.e., alphabet size equals to 5. The levels are *very low*, *low*, *medium*, *high*, and *very high*. For the power consumption data, quantization 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, quantization 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).

5.1 Accuracy, Correctness and Time Performance

The first experiment studies the accuracy of the proposed segment periodicity detection algorithm. We examine the accuracy by measuring the nearness of the score vector estimated by the algorithm to its exact value. Given that the exact calculation of the score vector requires $O(n^2)$, and given the execution time results shown in Figure 6 (a time series of size 128 Megabytes takes around 20 hours of execution time for only one iteration), smaller sizes of the time series are considered in this experiment. Figure 4 gives the results for the Wal-Mart timed sales transactions, where the number of iterations is varied first and then the time series size is varied. Figure 4(a) shows that the accuracy increases when more iterations are carried out. This is expected based on the discussion of the algorithm in Section 3.3, as the mean value of the estimated score vector over several iterations equals its exact value. Moreover, Figure 4(a) shows that an accuracy of 90% is achieved after only 3 iterations, thus Figure 4(b) is plotted using 3 iterations for each size. Figure 4(b) shows that the accuracy decreases when the time series size is increased, however it remains on the 90% level.

Inerrant synthetic data is used in the second experiment in order to inspect the correctness of both algorithms. The correctness measure will be the ability of the algorithms to detect the periodicities that were artificially embedded into the synthetic data. Figure 5 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 length. Recall that inerrant synthetic data is generated in such a way that it is

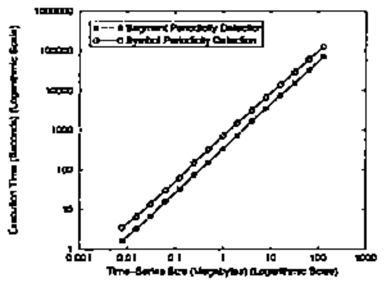


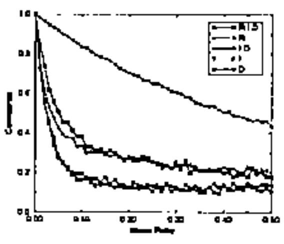
Figure 6: Time Behavior of Periodicity Detection Algorithms

perfectly periodic, and so the periodicities embedded are: $P, 2P, \ldots$. The confidence is the minimum threshold value required to detect a specific period. Since the data is perfectly periodic, the confidence of all the periodicities should be 1. Data sizes of 1 Megabytes are used with alphabet size of 10. The values collected are averaged over 100 runs. Figure 5(a) shows that the segment periodicity detection algorithm behavior depends on the period length. When the period length divides the data size, the algorithm detects all the periods at the highest confidence. When the period length does not divide the data size, the algorithm favors the lower values. However, Figure 5(b) shows an unbiased behavior of the symbol periodicity detection algorithm with respect to the period length.

To evaluate the time performance of the proposed periodicity detection algorithms, Figure 6 exhibits the time behavior of both algorithms with respect to the time series size. Wal-Mart timed sales transactions data is used in different portion sizes of powers of 2 up to 128 Megabytes. For the segment periodicity detection algorithm, only one iteration of the algorithm is considered. Figure 6 shows that the execution time is linearly proportional to the time series size. More importantly, the figure shows that segment periodicity detection takes less execution time than symbol periodicity detection.

5.2 Resilience to Noise

As enunciated before, there are three types of noise: replacement, insertion and deletion noise. This set of exper-


(a) Uniform, Period=32

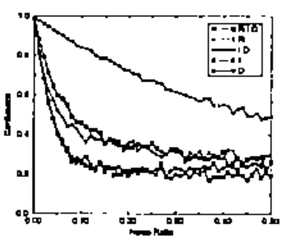
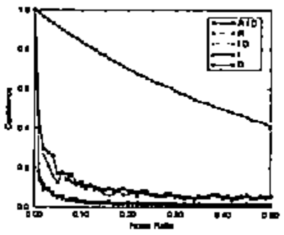

(b) Normal, Period=25

Figure 7: Resilience to Noise of Segment Periodicity Detection Algorithm


(a) Uniform, Period=25

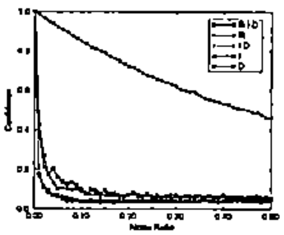
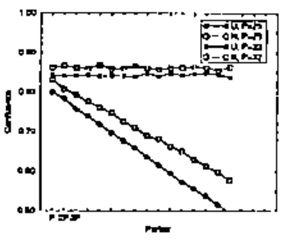

(b) Normal, Period=32

Figure 8: Resilience to Noise of Symbol Periodicity Detection Algorithm

iments studies the behavior of the periodicity detection algorithms towards these types of noise as well as different combinations of them. Results are given in Figure 7 and Figure 8 in which we use the symbols "R", "I", and "D" to denote the three types of noise, respectively. Combining two or more types of noise, e.g., "R I D", means that the noise ratio is distributed fairly among them. Data sizes of 1 Megabytes are used with alphabet size of 10. The values collected are averaged over 100 runs. Since the behaviors were similar disregarding the period length or the data distribution, an arbitrarily combination of period length and 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% thresholds, both algorithms can tolerate 50% replacement noise in the data. When the other types of noise get involved separately or with replacement noise, the algorithms perform poorly. However, segment periodicity detection can be considered roughly resilient to those other types since segment similarity thresholds in the range 10% to 20% are not uncommon.

Another experiment in this set is a rerun of the experiment given in Figure 5 but in the presence of noise. Only replacement noise is considered here as it is shown that the algorithms are more resilient to replacement noise than to the other types of noise. The results given in Figure 9 show a very similar behavior to that of Figure 5 but with a decrease in the confidence values due to the presence of noise.


(a) Segment Periodicity Detection

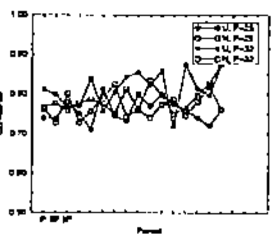

(b) Symbol Periodicity Detection

Figure 9: Correctness in presence of replacement noise

5.3 Real Data Experiments

Table 1 and Table 2 display the output for both algorithms for the Wal-Mart and CIMEG data for different values of the threshold. Clearly, the algorithms output fewer period lengths for higher threshold values. Moreover, the period lengths detected with respect to a certain value of the threshold are enclosed within those detected with respect to a lower value. To verify their correctness, the algorithms should at least output the period lengths that are known a priori in the time series. Wal-Mart data has a known period of length 24 that corresponds to the daily pattern of number of transactions per hour. CIMEG data has a known period of length 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 length 24 is detected when the threshold is 70% or less. Other interesting period lengths include those that are multiples of 24. A period of length 168 (24×7) can be explained as the weekly pattern of number of transactions per hour. A period of length 3961 or 4063, however, does not have a clear explanation, yet it proves that there may be obscure periods of interest. Similarly, for CIMEG data, the period of length 7 is detected when the threshold is 60% or less. Other clear period lengths are those that are multiples of 7. However, a period of length 123 days is difficult to explain. A similar behavior is shown in Table 2 for the symbol periodicity detection algorithm with higher number of output period lengths than in segment periodicity detection. This is expected since symbol periodicity detects period lengths 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 length 24 for Wal-Mart, and that of length 7 for CIMEG data, produces the results shown in Table 3. Note that periodicity is reported as a pair, consisting of a symbol and a starting position for a certain period. For example, Table 1 suggests a periodicity of the symbol *b* with a period of length 24 starting at position 7 for Wal-Mart data, with respect to a symbol periodicity threshold of 80% or less. Knowing that the symbol *b* represents the *low* level for Wal-Mart data (less than 200 transactions per hour), this periodicity can be interpreted as: less than 200 transactions per hour occur in the 7th hour of the day (be-

Segment Similarity Threshold (%)	Wal-Mart Data		CIMEG Data	
	# Output Period Lengths	Interesting Period Lengths	# Output Period Lengths	Interesting Period Lengths
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 1: Segment Periodicity Detection Output

Symbol Periodicity Threshold (%)	Wal-Mart Data		CIMEG Data	
	# Output Period Lengths	Interesting Period Lengths	# Output Period Lengths	Interesting Period Lengths
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 2: Symbol Periodicity Detection Output

tween 7:00am and 8:00am) for 80% of the days. This latter interpretation shows that symbol periodicity can itself be considered a pattern mining technique for time series databases, as each detected periodicity can be interpreted as an informative pattern form. Moreover, combining the periodicities that occur in the same period length may lead to further benefits, as will be shown in the next section.

6. DISCUSSION

The main objective of our algorithms is to discover candidate periods for a given time series, so that any time series pattern mining algorithm requiring a period length as input can make use of the candidate periods. In other words, our algorithms are applied as a preliminary step for the time series mining process, and then pattern mining algorithms such as [2, 5, 20, 9, 12, 13] are executed to detect the actual periodic patterns. We discuss an example that uses the partial periodic patterns mining algorithm introduced in [13].

A partial periodic pattern specifies the behavior of the time series at some but not all of the points in time. For example, in a stock price time series, a partial periodic pattern might state that the prices of the stock are high every Friday, low every Tuesday, and do not have such regularity on the other weekdays. In [13], Han et al. proposed a two-scan algorithm to mine those partial periodic patterns. This algorithm requires the user to specify the period length as input. The first scan constructs a maximal partial periodic pattern, whereas the second scan detects the actual partial periodic patterns. Adding periodicity detection, the entire mining process would require one scan to detect the candidate periods, and then two scans per candidate period to detect the partial periodic patterns.

A very important observation of the proposed symbol periodicity detection algorithm is that not only does it detect a candidate period for a certain symbol, but it also locates the position of that symbol in that candidate period. Accordingly, a maximal periodic pattern for each candidate period can be formed using all the symbols that appear to have this period. For example, the results given in Table 3 for Wal-Mart data show that with respect to a symbol periodicity threshold of 50%, there are 13 periodicities for a period of length 24, then a maximal periodic pattern of length 24 can be formed as *aaaa*§*bbbbc*§§§§§§*d*§§§*aaa*, where the letter § represents a "*don't care*" position. This maximal periodic pattern conforms to the one constructed at the first scan of the partial periodic patterns mining algorithm [13]. Note that although each symbol in this pattern occurs more than 50% of the time, the frequency of the entire pattern may be less than 50% [4]. Table 4 gives the results of applying the partial periodic patterns mining algorithm autonomously on Wal-Mart data using a period of length 24. It shows that all the discovered partial periodic patterns are sub-patterns of the maximal periodic pattern *aaaa*§*bbbbc*§§§§§§*d*§§§*aaa*.

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

Table 4: Partial Periodic Patterns for Wal-Mart Data

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

Symbol Periodicity Threshold (%)	Wal-Mart Data Period Length=24		CIMEG Data Period Length=7	
	# Output Periodicities	Interesting Periodicities	# Output Periodicities	Interesting Periodicities
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 3: Symbol Periodicity Detection Output (cont.)

Therefore, replacing that first scan by the proposed symbol periodicity detection algorithm would result in a mining process that requires one scan for detecting the candidate periods and the corresponding maximal periodic patterns, followed by one scan per candidate period to detect the partial periodic patterns.

It is worth mentioning that Ma and Hellerstein in [18] try to handle the unknown period problem for partial periodic patterns mining. However, their algorithm misses a lot of correct periods since it only considers the adjacent inter-arrivals. For example, if a symbol 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 in [18] will increase to $O(n^2)$.

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 size n. An empirical study of the algorithms using real-world and synthetic data sets proves the practicality of the problem, validates the correctness of the algorithms, and the usefulness of their outputs. Moreover, segment periodicity detection takes less execution time whereas symbol periodicity detects more period lengths. 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. Finally, we have studied the integration of our proposed periodicity detection algorithms in the entire process of time series mining, and have proved its effectiveness in the case of partial periodic patterns mining.

8. REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. In *Proc. of the 4th Int. Conf. on Foundations of Data Organization and Algorithms*, Chicago, Illinois, October 1993.

[2] R. Agrawal, K. Lin, H. Sawhney, and K. Shim. Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time Series Databases. In *Proc. of the 21st Int. Conf. on Very Large Databases*, Zurich, Switzerland, September 1995.

[3] R. Agrawal, G. Psaila, E. Wimmers, and M. Zait. Querying Shapes of Histories. In *Proc. of the 21st Int. Conf. on Very Large Databases*, Zurich, Switzerland, September 1995.

[4] 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.

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

[6] 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*.

[7] M. Atallah, F. Chyzak, and P. Dumas. A Randomized Algorithm for Approximate String Matching. *Algorithmica*, Vol. 29, No. 3, pages 468-486, 2001.

[8] 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 European Conf. on Artificial Intelligence*, Lyon, France, July 2002.

[9] 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.

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

[11] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast Subsequence Matching in Time Series Databases. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, Minneapolis, Minnesota, May 1994.

[12] 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.

[13] 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.

[14] 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.

[15] P. Indyk. Faster Algorithms for String Matching Problems: Matching the Convolution Bound. In *Proc. of the 39th IEEE Annual Symposium on Foundations of Computer Science*, Palo Alto, California, November 1998.

[16] 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.

[17] D. Knuth. *The Art of Computer Programming*, Vol. 2, Second Edition, Series in Computer Science and Information Processing, Addison-Wesley, Reading, MA, 1981.

[18] 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.

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

[20] 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.

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

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