Trend Based Periodicity Detection for Load Curve Data

Zhihui Guo[1]
zhihuig@cs.sfu.ca

Wenyuan Li[2&1], Adriel Lau[2], Tito Inga-Rojas[2]
{wen.yuan.li, adriel.lau, tito.inga-rojas}@bchydro.com

Ke Wang[1]
wangk@cs.sfu.ca

1. School of Computing Science, Simon Fraser University, Burnaby, BC, Canada
2. BC Hydro and Power Authority, Vancouver, BC, Canada

***Abstract*—The authors propose a novel periodicity detection for load curve data that is *trend based*, therefore, noise resilient. This method models key information in load curve data by a sequence of peaks and valleys extracted from a smoothing curve, and extends Dynamic Time Warping technique to discover repeating subsequences of such shapes while allowing variations due to background noises. Our experimental results show that it is able to detect periodicities more accurately than existing algorithms.**

Keywords-Load curve; time series; periodicity detection; smoothing techniques; noise resilient.

I. INTRODUCTION

Load curve data are time series of electricity consumption recorded at regular time intervals. Power utilities rely on periodic consumption patterns for important applications such as load models, load forecasting, system analysis, maintenance scheduling, energy economics, system risk evaluation and data cleansing [1][2]. While periodic patterns are typically known when electricity usage is stable and regular, they need to be discovered in the following cases: the periodicity has evolved, a new load curve is collected, or the conditions that affect consumption have changed. For example, a factory may change from two shifts to three shifts per day due to changed market demand, which will lead to pattern changes in the load curve data. For a load curve at a substation where there are mixed customers with different energy consumption patterns, identifying the combined periodicity is very useful for building a meaningful load model in power system applications but it is a real challenge.

We consider the problem of automatically detecting the periodicities of load curve data, assuming that there are periodic patterns in the data. The load curve data can be noisy in that there are variations in exact consumption values and variations in re-occurrence time. We also assume that there may be occasionally missing and corrupted data. For example, a factory shutdowns production line during one week union strike, so the consumption during this week is unexpected low. This kind of occasional deviations should not affect detection of periodic patterns because it does not periodically occur.

Periodicity detection has been an active topic in statistics [3][4][5][6][7][8][9]. To deal with real-valued data such as electricity consumption, one approach in the literature [1][3][4] is first discretizing a real valued time series into a sequence of discrete symbols and then searching for *segment periodicity*, which consists of repetition of a segment of symbols in the sequence. Common discretization methods include equi-width binning, where each bin has the same width in load, or equi-depth binning, where each bin contains the same number of data points.

Unfortunately, the above approach suffers from major drawbacks. Consider the four days' hourly load curve in Figure 1. This data has a strong daily periodicity consisting "large peak, small valley, small peak, small valley, small peak", as indicated by the rectangles in blue. The equi-width binning could discretize the y-values into four equal-sized bins *a*, *b*, *c*, *d*, which breaks each big peak into several bins (i.e., *a, b,* and *c*) and collapses all small peaks and valleys into one bin *d*. Clearly, the daily trend is lost after the discretization. Using fewer but larger bins may be able to preserve a large peak as one piece, but will not be able to separate small peaks and valleys, which are part of the daily pattern. This problem is rooted from the fact that discretization considers each time point independently and is insensitive to patterns and trends that are neighborhood based. There is a similar problem with equi-depth binning that is insensitive to patterns too.

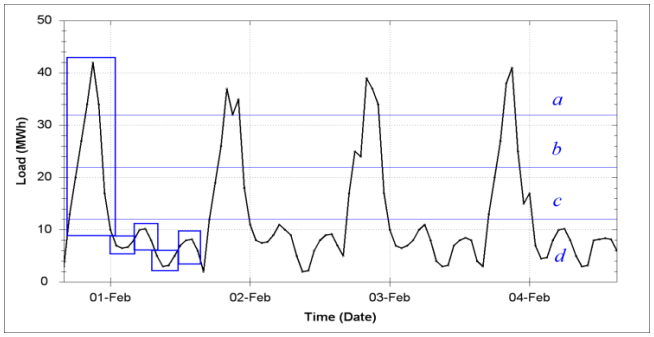

Figure 1 Four days' data with daily periodicity

The data in Figure 1 illustrates two related points. First, the peaks and valleys of each day are not exactly the same but the trends represented by them are similar. Second, the peaks and valleys should be "large enough" to repeat themselves in order to be a trend. Therefore, if we can identify such peaks and valleys, we could detect the periodicity as re-occurrence of certain subsequences of peaks and valleys, allowing small variations due to background noises.

With these observations, we propose a novel *trend based* algorithm to detect periodicities in load curve data. The term "trend based" means that the method focuses on the trends, rather than the actual consumption value of every single data point. The idea is to approximate a load curve by a smoothing curve and decompose the latter into a sequence of valleys and peaks. We model each valley or peak by a feature vector and define a metric to measure the similarity between a pair of valleys or a pair of peaks. Then we extend the symbol based WARP algorithm [4] to the valley/peak based sequence to discover periodicities in terms of repeating segments of valleys and peaks.

The novelty of this approach is modeling a load curve as a sequence of neighborhood based features, i.e., valleys and peaks, which preserves trends and ignores background noises. Another feature of the proposed approach is the easiness of detecting multiple periodicities (e.g., daily, weekly, seasonal and yearly), by adopting a proper choice for the smoothing parameter to model the trends at a desired resolution level. Importantly, the user is not required to have good knowledge on such choices; a user-friendly visualization tool has been developed to help users converge to a proper choice with little effort. We evaluated the proposed approach using real life load curve data. The evaluation shows that the proposed method is able to detect more accurately than the discretization based methods.

The rest of the paper is organized as follows. In Section II, some concepts of the WARP algorithm are reviewed. In Section III, the trend based algorithm is presented. In Section IV, the performance of the trend based algorithm is evaluated. Finally, we conclude the paper in Section V.

II. CONCEPTS OF WARP ALGORITHM

We review the WARP algorithm [4], a periodicity detection algorithm for a sequence of discrete symbols. In the next section we will extend the WARP algorithm to deal with a real valued load curve time series.

A *time series* $T = e_1\, e_2 \ldots e_n$ is an ordered list of n feature values e_i at times i, $1 \le i \le n$. A *sequence* is the special case of time series where each feature value e_i is a discrete symbol taken from a dictionary of alphabets. We adopt the notion of *segment periodicity* from [3]: A sequence T is *periodic* with a *period* p if it can be divided into segments of length p, that are "almost similar". For example, the sequence T = "*abcabcabb*" has a period 3 with the noise "*b*" at the end.

Dynamic time warping (DTW) [10]is a measure of the distance between two sequences $A = a_1\, a_2 \ldots a_m$ and $B = b_1\, b_2 \ldots b_n$. The DTW distance of A and B, DTW(A, B) or DTW(m, n), is computed by a dynamic programming formulated as

$$DTW(i, j) = d(a_i, b_j) + \min \begin{cases} DTW(i-1, j-1) \\ DTW(i-1, j) \\ DTW(i, j-1) \end{cases} \quad (1)$$

where the function $d(a_i, b_j)$ returns the distance between two symbols a_i and b_j, defined as

$$d(a_i, b_j) = \begin{cases} 0 & a_i = b_j \\ 1 & a_i \neq b_j \end{cases} \quad (2)$$

To detect the periodicity in a single sequence $T = e_1\, e_2 \ldots e_n$, the WARP algorithm compares the original sequence T with a sequence obtained by shifting some number of symbols, p. If there is high similarity between the two in terms of the DTW distance, p is considered a candidate period. In Figure 2, with $T = e_1\, e_2 \ldots e_9$ = "*abcabcabd*" and p=3, let $T_{(3)}$ denotes "*abcabc*", the first 9-3=6 symbols in T, and let $T^{(3)}$ denotes "*abcabd*", the last 9-3=6 symbols in T. DTW($T_{(3)}$, $T^{(3)}$) = 1. If this warping cost is considered small enough, p = 3 is returned as a candidate period.

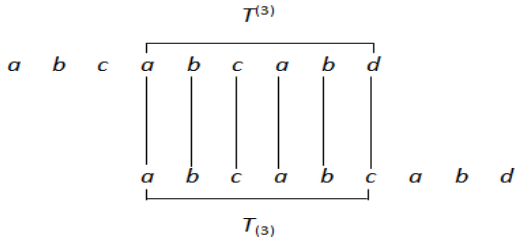

Figure 2 Alignment for $T_{(3)}$ and $T^{(3)}$ where T = "*abcabcabd*"

To find all candidate periods, for p = 1, …, n/2, DTW($T_{(p)}$, $T^{(p)}$) is computed. The maximum value of DTW($T_{(p)}$, $T^{(p)}$) is $n - p$. For each possible value of p, the *confidence* of p is

$$(n - p - DTW(T_{(p)}, T^{(p)})) \big/ (n - p).$$

If the confidence of p is larger than or equal to a given threshold τ, p is considered a candidate period. For a more detailed description of the WARP algorithm, readers are referred to [4].

III. TREND BASED ALGORITHM

The symbol based WARP algorithm cannot be directly applied to real valued load curve time series because it can only tell the equality, not distance, between two real values. In this section, we present the *trend based algorithm* to detect the periodicity in a real valued load curve time series. The algorithm has four steps:

1) Approximate the load curve using a smoothing curve;

2) Model the smoothing curve by a sequence of special $\cup$ shapes and $\cap$ shapes that correspond to peaks and valleys in the smoothing curve;

3) Identify the periodicity by extending the symbol based DTW distance to sequences of ∪ and ∩ shapes, taking into account the similarity between such shapes;

4) Express the periodicity in the length of time.

Below, we explain each step in details.

A. Approximating Load C urve by Smoothing Curve

Given a load curve $T = e_1\ e_2\ \ldots\ e_n$, the trends of the data can be modeled by a smoothing curve obtained by applying a smoothing technique to *T*. In this paper, we use the kernel smoothing technique and the *Nadaraya-Watson estimator* [10]. The level of modeling details is controlled by the *smoothness level*. To help a user to choose a proper smoothness level, we have developed a software tool with a user-friendly interface, which allows the user to slide a bar for the smoothness level and displays the corresponding smoothing curve interactively. Based on the visual inspection of the fit between the smoothing curve and the load curve, the user can adjust the smoothness level using the sliding bar. After several trials the user is able to converge to a desired smoothness level.

B. Approximating Smoothing Curve by Peaks and Valleys

The interesting information of a smoothing curve lies at the "peaks" and "valleys" that correspond the ups and downs of the load while paying less attention to actual data values. Therefore, we shall detect the periodicity in the load curve using patterns of such peaks and valleys. First, let us define valleys/peaks and represent them as feature vectors. We start with some terminologies.

Consider a smoothing curve $\hat{T} = \hat{e}_1\hat{e}_2...\hat{e}_n$ of the load curve time series *T*. The *slope* at time *i* is defined by $\Delta e_i / \Delta t_i$, where $\Delta e_i = \hat{e}_i - \hat{e}_{i-1}$ and Δt_i is the time span between time *i* and time $i - 1$, $2 \le i \le n$. The smoothing curve is a sequence of alternating peaks and valleys. An adjacent pair of peak and valley is separated at a time point that has a minimal or maximal slope. Such time points are called *steep points*. In other words, at a steep time point, the smoothing curve ascends or descends at a maximal rate.

A time interval [*a*, *b*] is *maximal-decreasing* if the slope at every time point in [*a*, *b*] is ≤ 0 and any larger interval containing [*a*, *b*] has a time point with a positive slope. For any time point *c* in a maximal-decreasing interval [*a*, *b*], [*c*, *b*] is *convex-decreasing* if *c* is the right-most steep point in [*a*, *b*], and [*a*, *c*] is *concave-decreasing* if *c* is the left-most steep point in [*a*, *b*]. Consider the smoothing curve in Figure 3 where the values on the curve indicate the slopes and the horizontal axis represents the time *i*. $[t_1, t_5]$ is a maximal-decreasing interval, and t_3 and t_4 are two steep points in $[t_1, t_5]$. $[t_4, t_5]$ is a convex-decreasing interval, but $[t_3, t_5]$ is not because t_3 is not the right-most steep point in $[t_1, t_5]$. $[t_1, t_3]$ is a concave-decreasing interval.

Similarly, we can define *maximal-increasing* intervals, *concave-increasing* intervals, and *convex-increasing* intervals. In Figure 3, $[t_6, t_{10}]$ is a maximal-increasing interval, t_8 and t_9 are steep points, $[t_6, t_8]$ is a convex-increasing interval, and $[t_9, t_{10}]$ is a concave-increasing interval.

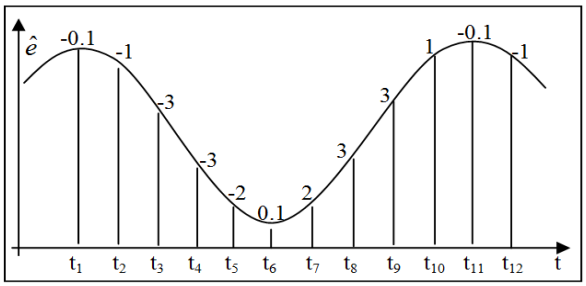

Figure 3 A smoothing curve over time interval $[t_1, t_{12}]$. The values on the curve are the slopes at each point. $[t_1, t_5]$ is a maximal-decreasing interval; $[t_6, t_{10}]$ is a maximal-increasing interval. $[t_4, t_8]$ is a ∪ shape.

Definition 1 (∪ *shape*): For a smoothing curve $\hat{T} = \hat{e}_1\hat{e}_2...\hat{e}_n$, a ∪ shape is a sub curve $\hat{e}_i...\hat{e}_j$ such that for some *k* with $i \le k \le j$, [*i*, *k*] is a convex-decreasing interval and [*k* + 1, *j*] is a convex-increasing interval.

Definition 2 (∩ *shape*): For a smoothing curve $\hat{T} = \hat{e}_1\hat{e}_2...\hat{e}_n$, a ∩ shape is a sub curve $\hat{e}_i...\hat{e}_j$ such that for some *k* with $i \le k \le j$, [*i*, *k*] is concave-increasing interval and [*k* + 1, *j*] is a concave-decreasing interval.

In Figure 3, the curve over $[t_4, t_8]$ is a ∪ shape formed by a convex-decreasing interval $[t_4, t_5]$ and a convex-increasing interval $[t_6, t_8]$. The curve over $[t_9, t_{12}]$ is a ∩ shape formed by a concave-increasing interval $[t_9, t_{10}]$ and a concave-decreasing interval $[t_{11}, t_{12}]$. Note that two adjacent ∪ shape and ∩ shape overlap by at most one point. Consider the adjacent ∪ shape $[t_4, t_8]$ and ∩ shape $[t_9, t_{12}]$ in Figure 3. The ∪ shape $[t_4, t_8]$ must end at the left-most steep point t_8 in the maximal-increasing interval $[t_6, t_{10}]$, and the ∩ shape $[t_9, t_{12}]$ must start at the right-most steep point t_9 in $[t_6, t_{10}]$.

There might be a gap between adjacent ∪ shape and ∩ shape. To cover all time points for the smoothing curve, we extend each shape to cover a half of the gap on each of its two ends. With this extension, the smoothing curve can be partitioned into a sequence of alternating ∪ shapes and ∩ shapes. This sequence is called the *shape sequence* and is denoted by *S*. Each ∪ shape and ∩ shape is represented by a feature vector (*sig, len, max, ave, min*), where *sig* indicates whether it is a ∪ shape or ∩ shape; *len* is the number of time points of the shape; *max* is the highest value of the shape; *ave* is the average value of the shape; *min* is the lowest value of the shape. Two shapes are similar to each other, if their feature vectors are similar.

C. Identifying Periodicities

The next step is detecting the periodicity using the shape sequence. Let us consider an example to illustrate the idea. Figure 4 represents eight weeks' load curve data and the smoothing curve. The second rectangle box indicates a periodic pattern that occurs in most of the weeks, i.e., one ∩ shape (weekdays) followed by one ∪ shape (weekend). The first and third rectangle boxes indicate deviations from this pattern where the ∪ shapes and ∩ shapes have quite different time lengths and different y values at some points

from the other weeks. Despite such deviations, there is a periodicity because majority of weeks have a "similar" sequence of a large ∩ followed by a small ∪ shape. Below, we describe an algorithm for detecting periodicity based on this idea.

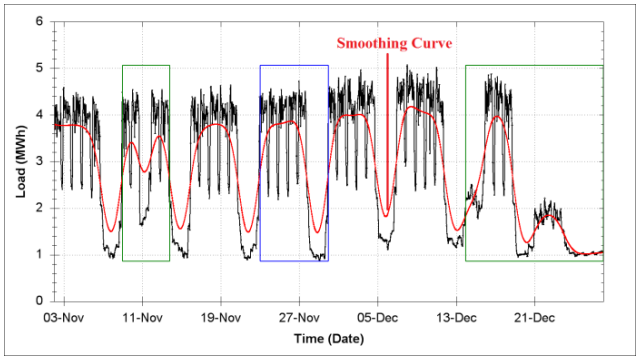

Figure 4 Example for a period

Our algorithm extends the symbol based WARP algorithm in Section II to the shape based sequence S. A key component of the WARP algorithm is the DTW distance between two sequences $A = a_1\ a_2\ \ldots\ a_m$ and $B = b_1\ b_2\ \ldots\ b_n$ of symbols. The DTW distance makes use of a distance function $d(a_i, b_j)$ for the two symbols a_i and b_j that are either identical or different. For two shapes a_i and b_j, a direct application of this distance function always yields $d(a_i, b_j) = 0$ because it is unlikely that two shapes are exactly identical.

To adopt the distance function $d(a_i, b_j)$ to two shapes a_i and b_j, we introduce a similarity threshold ε: a_i and b_j are considered *similar* if they have the same type, i.e., either both are ∪ shapes or both are ∩ shapes, and if their relative difference in length, max value and min value is no more than ε. Precisely, let a_i = (*sig_a*, *len_a*, *max_a*, *ave_a*, *min_a*) and b_j = (*sig_b*, *len_b*, *max_b*, *ave_b*, *min_b*). $d(a_i, b_j) = 0$ if all of the following conditions hold:

1) $sig\_a = sig\_b$,
2) $|len\_a - len\_b| / \mathrm{Max}(len\_a, len\_b) \le \varepsilon$,
3) $|max\_a - max\_b| / \mathrm{Max}(max\_a, max\_b) \le \varepsilon$,
4) $|ave\_a - ave\_b| / \mathrm{Max}(ave\_a, ave\_b) \le \varepsilon$, and
5) $|min\_a - min\_b| / \mathrm{Max}(min\_a, min\_b) \le \varepsilon$.

Otherwise, we define $d(a_i, b_j) = 1$. With this definition of $d(a_i, b_j)$ for two shapes, the WARP framework in Section II can be applied to the shape sequence S.

D. Computing Length of Candidate Periods

The above algorithm returns a set of candidate periods, where each candidate period is a sequence of alternating ∪ shapes and ∩ shapes. The final step of our algorithm is to transform each candidate period into a candidate period in terms of the length of time. Consider a candidate period p. Suppose that the shape sequence S has n shapes in total. For $i = 1, 2, \ldots, p$, the i-th shape in the period p is expected to occur at the location for all the j-th shapes in S, where $j = i + k \times p$, $0 \le k \le (n - i)/p$. The *time length* for the i-th shape in the period p is defined as the average length of these j-th shapes, and the *time length* for the period p is defined as the sum of the time length for the i-th shape in the period p over $i = 1, 2, \ldots, p$.

IV. Experiments

We studied the performance of the trend based algorithm by comparing it with the WARP algorithm [4]. Before applying the WARP algorithm, we first discretized load values in a load curve into four bins using equi-width binning. We also tried other ways of binning and the results are similar.

A. Setup

A real life load curve dataset recorded by BC Hydro was used in our experiments. It is hourly electricity consumption from January 2008 to December 2008, and has a weekly periodicity. Therefore, the periods are known: $24 \times 7 \times i$ hours, where $i = 1, 2, \ldots, 52/2$, which will be used as the ground truth to evaluate the accuracy of detection algorithms. About 15% of the data was corrupted into low values due to unknown reasons. The difference threshold ε for two shapes a_i and b_j (Section III.C) is set to 30%. The confidence threshold θ (Section II) is ranged from 0.7 to 0.9. We used five smoothness levels for kernel smoothing:

$$h = \frac{2^{i-1}}{1000}, i = \{1,2,...,5\} \quad (3)$$

Level $i = 1$ corresponds to the roughest level and level $i = 5$ corresponds to the smoothest level.

B. Accuracy

We say that a detected period p_d is *correct* if there exists a real period p_i such that $|p_d - p_i| / \max(p_d, p_i) \le \eta$. In our experiments, η is set to 5%. *True positive* (TP) is the number of correctly detected periods; *false positive* (FP) is the number of wrongly detected periods; *false negative* (FN) is the number of real periods that are not detected. We use the standard precision (P), recall (R) and F-measure (F) to measure accuracy, which are defined as follows:

$$precision = \frac{tp}{tp + fp} \quad (4)$$

$$recall = \frac{tp}{tp + fn} \quad (5)$$

$$F = \frac{2 \times precision \times recall}{precision + recall} \quad (6)$$

TABLE I ACCURACY COMPARISON

Confidence Threshold (%)	Trend based algorithm (Smoothness level = 3)						WARP algorithm					
	TP	FP	FN	P	R	F	TP	FP	FN	P	R	F
70	22	4	4	85%	85%	0.85	26	51	0	34%	100%	0.50
75	22	4	4	85%	85%	0.85	26	51	0	34%	100%	0.50
80	22	4	4	85%	85%	0.85	21	49	5	30%	81%	0.44
85	19	4	7	83%	73%	0.78	3	46	23	6%	12%	0.08
90	9	2	17	82%	35%	0.49	0	0	26		0%	

The accuracy comparison is shown in TABLE I. FP of the WARP algorithm is much larger than that of the trend based algorithm. This is because the WARP algorithm maps real-valued load data to a fixed number of bins, which is not sensitive to the trends in the data. Consequently, many false patterns were generated. In contrast, the trend based algorithm preserves the weekly pattern through the smoothing curve and the $\cup$ shapes and $\cap$ shapes on the smoothing curve. Except for the very high confidence threshold 90%, the trend based algorithm finds most periods correctly (i.e., 19 or 22 out of 26) while returning a few false positives. This yields the F-measure of 0.85 that is significantly higher than that of the WARP algorithm. This study suggests that the trend based algorithm is able to detect periodicity.

C. *Effect of Smoothness Levels*

TABLE II shows the effect of smoothness level. When the smoothness level is low, say level 1, both TP and FP are extremely low because the smoothing curve models too many details, which leads to many small $\cup$ shapes and $\cap$ shapes that are largely contributed by noises. Most of such small shapes are considered dissimilar, therefore, little repetition is found. When the smoothness increases to level 3, the smoothing curve correctly models a sequence of $\cap$ shapes and $\cup$ shapes corresponding to high usages during the weekdays and low usages at the weekend. In this case, the trend based algorithm finds all real periods with only four false positives. When the smoothness reaches level 5, the smoothing curve is rather flat and there are only a few large size $\cup$ shapes and $\cap$ shapes each of which includes more than one week's data, resulting in the weekly periods not being detected.

TABLE II EFFECT OF SMOOTHNESS LEVEL IN THE TREND BASED ALGORITHM (CONFIDENCE THRESHOLD SET AS 70%)

Smoothness Level	TP	FP	FN	P	R	F
1	0	0	26		0%	
2	4	0	22	100%	15%	0.27
3	22	4	4	85%	85%	0.85
4	14	1	12	93%	54%	0.68
5	2	0	24	100%	8%	0.14

This study suggests that a proper smoothness level is crucial for the trend based algorithm. We have developed a user-friendly interface and visualization tool to assist the user. With this tool, the user can converge to a proper smoothness level by sliding a bar for the smoothness level a few times.

V. CONCLUSION

Identifying multiple periodicities in a load curve, particularly those representing mixed customers with different energy consumption patterns, is very useful and important for various system studies in power utilities. We proposed a novel trend based algorithm to detect the underlying periodicity of load curve data in power utilities. This approach is trend preserving, noise resilient, and flexible for detecting multiple periodicities, yet not requiring the user's knowledge on a proper choice of the smoothing parameter.

REFERENCES

[1] Chen, J., Li, W., Lau, A., Cao, J. and Wang, K., "Automated load curve data cleansing in power systems", *IEEE Transactions on Smart Grids*, Vol. 1, No. 2, pp 213-221, September 2010.

[2] Li, W., "Risk assessment of power systems: models, methods, and applications", IEEE Press—Wiley, 2005.

[3] Elfeky, M.G., Aref, W.G. and Elmagramid, A.K., "Periodicity detection in time series databases", *IEEE Transactions on Knowledge and Data Engineering*, 2005.

[4] Elfeky, M.G., Aref, W.G. and Elmagarmid, A.K., "WARP: time warping for periodicity detection", *IEEE International Conference on Data Mning*, 2005.

[5] Indyk, P., Koudas, N. and Muthukrishan, S., "Identifying representative trends in massive time series data sets using sketches", *International Conference on Very large Data Bases*, 2000.

[6] Ma, S. and Hellerstein, J., "Mining partially periodic event patterns with unknown periods", *IEEE International Conference on Data Engineering*, 2001.

[7] Papadimitriou, S., Brockwell, A. and C. Faloutsos, "Adaptive, hands-off stream mining", *International Conference on Very Large Data Bases*, 2003.

[8] Berberidis, C. Aref, W. Atallah, M., Vlahavas, I. and Elmagarmid, A., "Multiple and partial periodicity mining in time series databases", *ECAI*, 2002.

[9] Yang, J. Wang, W. and Yu, P., "InfoMiner +: mining partial periodic patterns with gap penalties", *IEEE International Conference on Data Mining*, 2002.

[10] Berndt, D. and Clifford, J., "Using dynamic time warping to find patterns in time series,", *KDD*, 1994.

[11] Hrdle, W., *Applied Nonparametric Regression.* Cambridge University Press, 1990.