Improving SPRING Method in Similarity Search Over Time-Series Streams by Data Normalization

Bui Cong Giao(✉) and Duong Tuan Anh

Faculty of Computer Science and Engineering,
Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
giao.bc@cb.sgu.edu.vn, dtanh@cse.hcmut.edu.vn

Abstract. Similarity search in streaming time series is a crucial subroutine of a number of real-time applications dealing with time-series streams. In finding subsequences of time-series streams that match with patterns under Dynamic Time Warping (DTW), data normalization plays a very important role and should not be ignored. SPRING proposed by Sakurai et al. conducts the similarity search by mitigating the time and space complexity of DTW. Unfortunately, SPRING produces inaccurate results since no data normalization is taken into account before the DTW calculation. In this paper, we improve the SPRING method to deal with similarity search for prespecified patterns in streaming time series under DTW by applying incremental min-max normalization before the DTW calculation. For every pattern, our proposed method uses a monitoring window anchored at the entry of one streaming time series to keep track of min-max coefficients, and then the DTW distance between the normalized subsequence and the normalized pattern is incrementally computed. The experimental results reveal that our proposed method obtains *best-so-far* values better than those of another state-of-the-art method and the wall-clock time of the proposed method is acceptable.

Keywords: Similarity search · Streaming time series · Data normalization · Dynamic Time Warping

1 Introduction

A streaming time-series is a sequence of real values, where new values are continuously appended at a steady high-speed rate as time progresses, so time-series streams are potentially unbounded in size. There have been more and more applications of data mining on streaming time series recently. The typical examples are traffic monitoring using GPS, sensor network monitoring, and online stock analysis. In these applications, pattern discovery by similarity search is a core function to identify immediately which new-coming time-series subsequences of streaming time series match with prespecified patterns.

For the past, the Euclidean metric was widely used to compute the distance between two time-series sequences. However, the distance measure is unsuitable

P.C. Vinh et al. (Eds.): ICTCC 2016, LNICST 168, pp. 189–202, 2016.
DOI: 10.1007/978-3-319-46909-6_18

when the two time-series sequences have the same shape but they are variable in the time axis and in the circumstance, Dynamic Time Warping (DTW) [1] shows its superiority in accuracy over the Euclidean metric. DTW has been noted in several domains, including bioinformatics, robotics, and finance, especially in multimedia such as speech recognition [2]. For instance, in a query-by-humming system [3], the computer searches a list of melodies that are most similar to a tune or a sung query echoed from a user, yet users almost tend to sing slower or faster than the digital song stored in a music database. Consequently, the system can hardly find the intended song from the out-of-phase query if the similarity search is conducted with the Euclidean metric. The DTW metric enables the system to retrieve the desirable song since it is able to find two similar time-series sequences although they are out of phase with each other. Nevertheless, using DTW is not enough for accurate retrievals of the song, since most users do not sing queries in the same pitch levels as the song. Normalization of both time-series subsequences of digital songs and the queries is needed to eliminate any existing offsets prior to any distance calculations.

There have been very few methods proposed for similarity search in streaming time-series under DTW. In 2007, Sakurai et al. [4] proposed SPRING, an outstanding method that can quickly find *best-so-far* subsequences for patterns over time-series streams under DTW. The authors claimed that SPRING can offer significantly improvements in speed (up to 650,000 times) over the naïve implementation of similarity search under DTW. However, the obtained results of SPRING are inaccurate in our experiments because the authors did not normalize the data before the DTW distance between any pair of time-series sequences is calculated. In 2012, Rakthanmanon et al. [5] introduced UCR-DTW, a method that can find *best-so-far* subsequences in static time series for patterns under DTW. UCR-DTW applies incremental z-score normalization and performs the task quickly and accurately. Recently, we have improved UCR-DTW to SUCR-DTW [6] so that the similarity search can be done in streaming time-series context. SUCR-DTW also performs the task as quickly and precisely as UCR-DTW does in static context. However, UCR-DTW and SUCR-DTW can find only the subsequences which have the same length as patterns. This characteristic of the two methods limits the benefit of DTW: the distance measure can be done on two sequences of different lengths.

The above shortcomings of some previous methods motivate us to develop a new method which can find accurately new-coming subsequences of streaming time series that are most similar to patterns under DTW. The proposed method is an improvement of SPRING method by applying incremental min-max normalization before the DTW calculation. Furthermore, our method can work in a very important scenario of streaming context where incoming data are from many concurrent high-speed time-series streams, and patterns for query are prespecified time-series sequences.

The rest of paper is organized as follows. Section 2 presents the background including an overview of DTW, data normalization, and the problem definition. Section 3 reviews related work. Section 4 describes the proposed method.

Section 5 discusses experimental evaluation, and Sect. 6 gives conclusions and future work.

2 Background

2.1 Dynamic Time Warping

Dynamic Time Warping (DTW) is a robust distance measure for time series. DTW allows time-series sequences stretched along the time axis to minimize the distance between them. DTW is calculated by dynamic programming depicted as follows. Consider two time-series sequences $C = c_1, c_2, \ldots, c_m$ and $Q = q_1, q_2, \ldots, q_n$. The DTW distance between C and Q is defined as:

$$\begin{aligned} &DTW(C,Q) = f(m,n) \\ &f(m,n) = d(c_i,q_j) + \min \begin{cases} f(i,j-1) \\ f(i-1,j) \\ f(i-1,j-1) \end{cases} \\ &f(0,0) = 0, f(i,0) = f(0,j) = \infty \\ &\qquad (1 \le i \le m, 1 \le j \le n) \end{aligned} \tag{1}$$

in which $d(c_i, q_j) = (c_i - q_j)^2$ is the Euclidean metric between two numerical values, c_i and q_j. Notice that any choice (e.g. $d(c_i, q_j) = |c_i - q_j|$) would be fine. Our proposed method is absolutely independent of such choices.

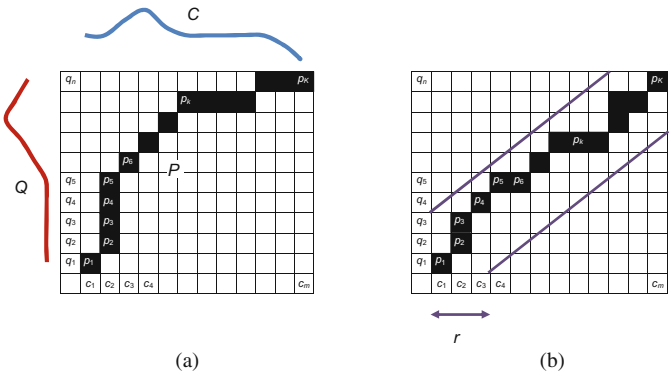

Fig. 1. (a) To align C and Q, a warping path P, shown with solid squares, is constructed. (b) The Sakoe-ChiBa band with a warping scope r is used as a global constraint to limit the scope of P.

To align C and Q using DTW, an n-by-m matrix, which is also referred to as an accumulated cost matrix, is constructed. The (i^{th}, j^{th}) cell of the matrix

contains the value of $f(c_i, q_j)$. A warping path P is a sequence of continuous cells in the matrix, which defines a mapping of C and Q such that $f(m, n)$ is minimum. An example of P is illustrated as in Fig. 1(a).

Since DTW uses a dynamic programming algorithm whose the time and space complexity are $\mathcal{O}(mn)$, the distance measure is almost very slow, especially for long time-series sequences. To accelerate the DTW calculation, we can limit the number of cells evaluated in the accumulated cost matrix. Figure 1(b) depicts a Sakoe-Chiba band [2] that prevents pathological warping paths, where a data point in one time series matches to too many data points of another as in Fig. 1(a). The Sakoe-Chiba band makes a warping window constrained by two lines parallel to the diagonal of the matrix.

2.2 Data Normalization

In temporal data mining, data normalization is essential to achieve a meaningful calculation of the distance between two time series because the normalized data have similar offset and distribution, regardless of any distance measure used, especially for the DTW measure. There are two popular ways to normalize time-series data: min-max and z-score. The normalized data would be computed from the coefficients of the two data normalization types. Min-max coefficients (minimum and maximum values) of evolving subsequences in streaming time series are changed now and then, whereas z-score coefficients (mean and standard deviation) of the subsequences are almost changed whenever there is a new-coming data. For the reason, we use min-max normalization in our proposed method so that its time complexity is lower. Min-max normalization maps a value x of time series $X = x_1, x_2, \ldots, x_n$ to x_{norm} by computing

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

where x_{min} and x_{max} are the minimum and the maximum values of time-series X.

As time-series sequences change continuously in the streaming context, data normalization becomes a burden for pre-processing time-series data prior to subsequence matching. Therefore, it is necessary to have a complementary technique for data normalization in the streaming setting. We propose incremental min-max normalization to preclude data normalization completely in the following paragraph.

In the beginning, an ascending numeric array is created from the data points of X with the algorithm of *Quicksort*, so x_{min} is the first element and x_{max} is the last one of the ordering array. After that, when there is a new-coming data point, the oldest data point of X is deleted out of the array, and then the new data point is inserted into the array. The course of the deletion and insertion must retain the ascending order of the array, so the algorithm of *Binary search* is used to find the element which needs deleting and the suitable position in the array to insert the new data point. As a result, the time complexity of the incremental min-max normalization is $\mathcal{O}(log(n))$.

2.3 *Best-so-far* Search in Streaming Time Series

The problem is defined as follows. A time-series stream S is a discrete, semi-infinite time-series sequence of real numbers $x_0, x_1, \ldots, x_n \ldots$ where x_n is the most recent value. In other words, X is a univariate time series, which is evolving with an increase of n after each time tick. Let $S[x_s : x_e]$ be the subsequence starting from time tick s, and ending at time tick e; and $NS[nx_s : nx_e]$ be the normalized subsequence of $S[x_s : x_e]$. Let $P[p_0 : p_{m-1}]$ be a time-series pattern of length m, and $NP[np_0 : np_{m-1}]$ be the normalized sequence of P. We want to find such a NS that is most similar to NP. This means that $DTW(NS, NP)$ is smallest until the most recent time tick n (see Fig. 3(a)). The smallest value is the *best-so-far* value, which is denoted as $P.bsf$, and S is also the *best-so-far* subsequence of P till time tick n.

3 Related Work

The section describes SPRING and SUCR-DTW which are closely related to our proposed method.

3.1 SPRING

The method detects high-similarity subsequences over time-series streams. Since SPRING works on un-normalized data, we only consider S and P. The method uses a subsequence time warping matrix (STWM) whose cells record the starting position of each candidate subsequence. Let denote the starting position as s for the (i, j) cell. In addition, the cell contains the value $f(i, j)$, which is the best distance to match the prefix of length $j - s + 1$ from a certain $S[x_s : x_h](h \geq j)$ with the prefix of length i from $P[p_0 : p_{m-1}]$. In other words, the values s and $f(i, j)$ in the STWM mean that the subsequence from s through j obtains $f(i, j)$, which is the minimum distance for the i- and j- prefix of P and S, respectively.

$p_3 = 4$	**0 – 27**	**1 – 15**	**1 – 10**	1 – 18	1 – 38	1 – 28	1 – 52	
$p_2 = 9$	0 – 26	**1 – 6**	1 – 10	1 – 2	1 – 3	1 – 3	1 – 7	
$p_1 = 6$	0 – 10	**1 – 2**	1 – 1	3 – 4	3 – 16	5 – 10	5 – 26	
$p_0 = 8$	0 – 9	**1 – 1**	2 – 4	3 – 0	4 – 4	5 – 1	6 – 9	
	$x_0 = 5$	$x_1 = 7$	$x_2 = 6$	$x_3 = 8$	$x_4 = 10$	$x_5 = 9$	$x_6 = 11$	$x_7 = \ldots$

Fig. 2. The subsequence time warping matrix shows *best-so-far* values until time tick 6.

We consider an illustration of SPRING. Assume that $P = 8, 6, 9, 4$ and $S = 5, 7, 6, 8, 10, 9, 11, \ldots$. The evolution of the STWM from time tick 0 to 6 is depicted as in Fig. 2. In the beginning, at time tick 0, the candidate subsequence $S[x_0 : x_0]$ has the distance $f(0, 3) = 27$. At time tick 1, the *best-so-far* value

is 15 with the subsequence $S[x_1 : x_1]$. Next, we found the *best-so-far* value is 10 with the subsequence $S[x_1 : x_2]$ at time tick 2. Until the most recent time tick 6, the *best-so-far* value is still 10 with the subsequence $S[x_1 : x_2]$. From the example, we note that SPRING is of low computational time. If the length of P is m, SPRING requires the time complexity of $\mathcal{O}(m)$ per time tick. Besides, every column is computed from its preceding column.

Recently, Gong et al. [7] have introduced NSPRING, an extension of SPRING supporting z-score normalization. The authors normalize data of the current column in the STWM with the current z-score coefficients, and then compute this normalized data with the normalized data of the preceding column, which were derived from the z-score coefficients of the preceding time tick. Since z-score coefficients are frequently changed during the course of streaming time series, data of the preceding time tick need normalizing again with the current z-score coefficients. For the reason, in our opinion, NPRING is inaccurate.

3.2 SUCR-DTW

The method uses the Sakoe-Chiba band and many lower bounding functions to speedup DTW. The lower bounding functions prune off unpromising time-series subsequences in a cascading fashion. These functions are arranged in the ascending tightness of the lower bounding property such that front lower bounding functions with low time complexities rule out most unpromising subsequences. As a result, the number of post-checking times using the classical DTW so as to determine if a candidate subsequence is a true hit is very tiny. For a detailed explanation, SUCR-DTW [6] employs improved LB_Kim [8] with the computation complexity $\mathcal{O}(1)$, LB_Keogh [9] with $\mathcal{O}(m)$, and finally reversed LB_Keogh [5] with $\mathcal{O}(m)$. The experiments in [6] indicated that the pruning powers of improved LB_Kim, LB_Keogh, and reversed LB_Keogh are roughly 55 %, 35 %, and 9 %, respectively. Therefore, the calculation of the classical DTW is about 1 % in the post-checking phase.

Since SUCR-DTW integrates data normalization in similarity search under DTW, the finding results are relatively accurate. As mentioned in Sect. 1, the major limit of SUCR-DTW is that the method retrieves only time-series subsequences that have the same length as patterns.

4 Proposed Methods

The proposed method is an improvement of SPRING by combining with incremental min-max normalization. Let denote the proposed method as ISPRING (Improved SPRING). In comparison with SPRING, ISPRING consists of two novel ideas.

Firstly, each pattern has a monitoring window anchored at the entry of one time-series stream to keep track of min-max coefficients in the window. Thanks to these coefficients, the normalized new-coming subsequence of the time-series

stream is derived. The time warping distance between the normalized subsequence and the normalized pattern is incrementally computed. Let l be the size of the monitoring window. If the min-max coefficients in the monitoring window are changed, the time warping distances need completely calculating from the starting time tick of the monitoring window $n - l + 1$, to the most recent time tick n, in order that the course ensures that finding results are still accurate. Figure 3(a) shows that while a streaming time series is evolving; the monitoring window must check minimum and maximum values of the corresponding subsequence.

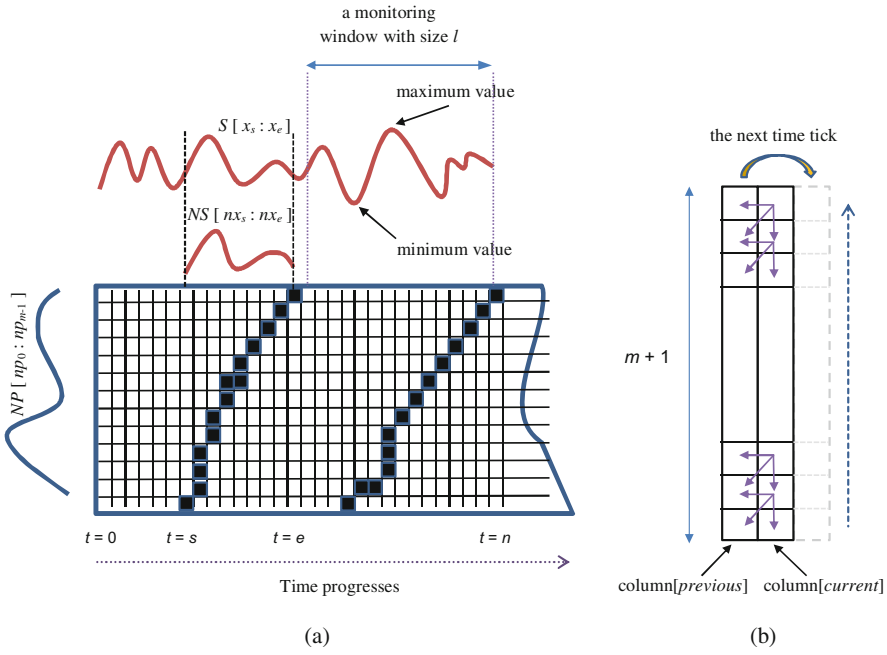

Fig. 3. (a) A window monitors the min-max coefficients. (b) The time warping distances are incrementally computed in a bottom-up fashion with the two columns.

Secondly, ISPRING uses two columns with size of $m + 1$ to maintain time warping distances computed incrementally. Hence, the memory space of ISPRING reduces significantly instead of using the subsequence time warping matrix (STWM) in SPRING. The space complexity is reduced from $\mathcal{O}(mn)$ of SPRING to $\mathcal{O}(m)$ of ISPRING. At the beginning of the similarity search, the two columns are initiated by Procedure *Reset_columns*. The two columns are exchanged the role with each other in every time tick. If a column is current at time tick i, then it will become previous at $i + 1$. Similar to SPRING, each cell of the column contains two kinds of information. The first is the starting

time tick from which the time warping distance is calculated. The second is the time warping distance. While streaming time-series is evolving, the information of cells in the current column is computed in a bottom up fashion. Figure 3(b) illustrates the calculation of the time warping distances in cells complies with the formula (1), so the time warping distance in the m^{th} cell of the current column will be the minimum distance from the starting time tick to the most recent time tick n. Next, this time warping distance is compared with the current *best-so-far* value of the pattern.

ISPRING is briefly described in Algorithm *ISPRING*. Notice that line 10 checks whether the subsequence $S[x_s : x_n]$ contains the min-max coefficients of the monitoring window. If the min-max coefficients are not in the subsequence, the subsequence needs expanding backward to contain them. The expansion increases the time warping distance in line 12 when the normalized data point nx_s needs matching with the first data point of the normalized sequence *NP*, that is np_0.

```
Algorithm ISPRING(Streaming time-series S, Pattern P)
begin
When there is a new-coming data of S: x_n
1.  if the monitoring window of P detects a change in their
    min-max coefficients then
2.     Reset_columns
3.     for (i = n - l + 1 ; i ≤ n ; i++)
4.        Set_current_column(i, nx_i)
5.  else
6.     Set_current_column(n, nx_n)
7.  dtw ← column[current][m].dtw
8.  if dtw < P.bsf then
9.     s ← column[current][m].start
10.    while the min-max coefficients ∉ S[x_s : x_n]
11.       s--
12.       dtw += d(nx_s, np_0)
13.       if dtw ≥ P.bsf then
14.          break
15.    end while
16.    if dtw < P.bsf then
17.       P.bsf ← dtw
18.       Record S[x_s : x_n] as the best-so-far subsequence of
          P
end
```

The two procedures *Reset_columns* and *Set_current_column* are presented in detail to clarify the subtle techniques in manipulating the two columns.

```
Procedure Reset_columns
begin
1.  for (i = 0 ; i ≤ m ; i++)
2.     column[1][i].dtw ← ∞
3.  current ← 1
4.  previous ← 0
end
```

```
Procedure Set_current_column(i, nx)
begin
1.  if current = 1 then
2.     current ← 0
3.     previous ← 1
4.  else
5.     current ← 1
6.     previous ← 0
7.  column[current][0].dtw ← 0
8.  column[current][0].start ← i
9.  for (i = 0 ; i ≤ m ; i++)
10.    temp ← column[current][i - 1]
11.    if temp.dtw > column[previous][i - 1].dtw  then
12.       temp ← column[previous][i - 1]
13.    if temp.dtw > column[previous][i].dtw  then
14.       temp ← column[previous][i]
15.    column[current][i].dtw ← temp.dtw + d(nx, np_{i-1})
16.    column[current][i].start ← temp.start
end
```

5 Experimental Evaluation

ISPRING and SUCR-DTW have been compared in terms of accuracy and wall-clock time. SUCR-DTW has been modified to implement *best-so-far* search and incremental min-max normalization instead of doing range search and incremental z-score normalization as in our previous work [6]. We use 5 % constraint on the warping path in SUCR-DTW. For example, if the length of a query is 200, then the warping scope r in Sakoe-Chiba band is 10. ISPRING is implemented with monitoring windows whose lengths are same as those of patterns.

The experiments have been conducted on an Intel Dual Core i3 M350 2.27 GHz, 4 GB RAM PC. Because of the characteristic of the search methods and the strength of today's CPU, we employ multi-threading in the implementation of ISPRING and SUCR-DTW. This means each threading process handles one time-series stream to perform the similarity search. For the sake of fairness, all threading processes are the same priority. Microsoft C# is powerful for multithreaded programming, so we use the programming language to implement the two methods. Another point we cannot ignore is that since many threading processes can compete to update the *best-so-far* value of one pattern at a time; the system must lock the shared resources and check the *best-so-far* value again before the update can be done.

Seven text files are used to simulate time-series streams. The sources of the text files are given in column 4 of Table 1. A pattern set is created from the text files. The number of patterns of the set is 20 and the lengths of all patterns are 256. Notice that ISPRING can work with many patterns concurrently, and every pattern that needs to find the *best-so-far* subsequence over time-series streams has its own monitoring window on each time-series stream. We used patterns of same length for the sake of clarity but no loss of generality. The number of patterns is created from a time-series file is proportional to the number of data points in the file. Every pattern is extracted from random positions in the time-series files. Next, all data points of a pattern are added by a numerical constant, and then the data points are virtually increased or decreased by a relatively small numeric value (e.g. 0.3 or −0.3). After that, 33 % of the data points are changed in which they get the value of the preceding data point or successive one, or the mean of neighboring ones.

Table 1. Text files are used to simulate time-series streams.

No	Time-series file	Length	Source
1	D2	50,000	[10]
2	ItalyPowerDemand_TEST	25,725	[11]
3	Lightcurve	27,204	[10]
4	MedicalImages_TEST	76,000	[11]
5	SonyAIBORobotSurface_TEST	42,671	[11]
6	SonyAIBORobotSurfaceII_TEST	62,898	[11]
7	TwoLeadECG_TEST	94,537	[11]

Since in reality, time-series streams are potentially unbounded in size, we design circular buffers whose sizes are 1,024, to store data points of time-series streams. This size of a circular buffer makes sure that ISPRING does not cause false dismissals for patterns whose lengths are 256.

The experimental results of the two methods are depicted in Table 2. We consider an illustration of the *best-so-far* search in case of pattern 3. By SUCR-DTW, while the seven time-series streams are arriving continuously,

Table 2. Statistic of the experimental results

Pattern	SUCR-DTW				ISPRING			
	#TS	#Position	Length	*bsf* value	#TS	#Position	Length	*bsf* value
1	3	5,420	256	1.9439	3	12,445	168	1.8089
2	3	5,435	256	1.9794	3	12,479	150	1.8518
3	5	24,164	256	2.0358	3	4,464	179	1.8659
4	2	19,606	256	1.2783	2	19,608	258	1.2779
5	3	21,199	256	2.2661	3	23,374	227	2.2059
6	4	75,955	256	0.8458	4	23,652	153	0.6341
7	4	52,014	256	0.4502	4	52,015	266	0.4144
8	4	1,812	256	1.1243	5	51,891	42	1.0536
9	4	29,699	256	0.5394	4	13,998	235	0.5012
10	5	60,364	256	0.9283	5	60,363	253	0.9270
11	5	37,530	256	0.9967	5	37,524	249	0.9937
12	5	32,947	256	1.48737	5	2,132	217	1.4151
13	6	22,872	256	1.0436	6	22,871	255	1.0426
14	6	17,702	256	0.9500	6	17,702	256	0.9500
15	7	33,139	256	0.9978	6	33,139	263	0.9938
16	7	86,976	256	1.1809	3	3,450	153	1.1597
17	7	69,489	256	0.9239	7	69,483	250	0.9207
18	7	40,087	256	1.0497	7	71,328	287	1.0264
19	7	76,957	256	0.9125	7	76,975	275	0.9110
20	7	24,316	256	0.9857	7	38,114	279	0.9757

the *best-so-far* (*bsf*) value of the pattern is discovered in streaming time-series 5 at time point 24,164 with a time-series subsequence of length 256. This means the starting time point of the subsequence is $24,164 - 256 - 1 = 23,907$. Similarly, ISPRING discovers the *best-so-far* value of pattern 3 in streaming time-series 3 at time point 4,464 with a time-series subsequence of length 179. The subsequence begins at time point $4,464 - 179 - 1 = 4,284$. As regards the pattern, the *best-so-far* value obtained by SUCR-DTW is 2.0358, whereas ISPRING returns 1.8659. Of twenty cases, there is one case (pattern 14) in which ISPRING has the same results as SUCR-DTW. Therefore, in the testbed, there are 5 % cases in which ISPRING and SUCR-DTW obtain the same results. In remaining 95 % cases, ISPRING gives better *best-so-far* values. However, as regards SUCR-DTW, the wall-clock time is 3:42.884 min and the average CPU time to process a new-coming data point is 5,880 ticks; whereas in respect of ISPRING, theses values are 4:42.68 min and 7,217 ticks, respectively. The evidence shows that the recalculation of the time warping distances in ISPRING is costly whenever the min-max coefficients of the monitoring window are changed. Another note is

that there are seven cases (35 %) in which the *best-so-far* subsequences obtained by ISPRING are longer than those done by SUCR-DTW, and remaining twelve cases (60 %) in which the *best-so-far* subsequences obtained by ISPRING are shorter than those done by SUCR-DTW. Hence, we note that ISPRING tends to find *best-so-far* subsequences shorter than the patterns.

We also experimented on the two methods with various testbeds and the obtained results indicated that ISPRING is better than SUCR-DTW in finding *best-so-far* results. Space limits preclude the presentation of the statistic of these testbeds.

Notice that sometimes ISPRING finds *best-so-far* subsequences, whose lengths are very short in comparison with the lengths of the patterns. For instance, in the above testbed, pattern 8 has the *best-so-far* subsequence whose length is 40, compared with 256 of the pattern. This implies that there is a relatively small section of the subsequence mapped onto a relatively large section of pattern 8. As a result, the matching between the subsequence and the pattern creates an unavoidable pathological warping path in the accumulated cost matrix. Figure 4 depicts an illustration of matching between two time-series sequences S and P whose lengths are very uneven, so a pathological warping path is built to match these two sequences. It is obviously that the undesirable matching of the two time-series sequences is a shortcoming of ISPRING.

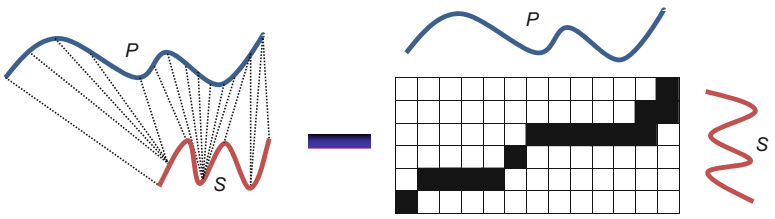

Fig. 4. The pathological matching between P and S

To determine which length of the monitoring window for every pattern is optimal, we vary the length by the formula:

$$\begin{aligned} &\text{The length of the pattern} \times (1 + \Delta) \\ &\Delta \text{ is changed to} - 5\,\%, -4\,\%, ..., 4\,\%, \text{ and } 5\,\% \text{ in turns.} \end{aligned} \tag{3}$$

We used again the dataset in Table 1 and the same pattern set for the test. Table 3 shows that with each Δ, the number of patterns has the *best-so-far* values better and worse when these values are compared to those with Δ of 0 %. Figure 5 presents the wall-clock times of ISPRING when the method is experimented with Δ from -5 % to 5 %. With Δ of -5 %, there is one case in which ISPRING finds a better *best-so-far* subsequence compared with the corresponding *best-so-far* subsequence with Δ of 0 %. However, the wall-clock time with Δ of -5 % is larger than that with Δ of 0 %, since if the length of the monitoring window is

Table 3. The number of cases in which *best-so-far* values are better and worse than Δ of 0 %

Δ (%)	−5	−4	−3	−2	−1	1	2	3	4	5
# better	1	0	0	0	0	0	0	0	1	1
# worse	0	1	1	1	1	1	1	1	1	1

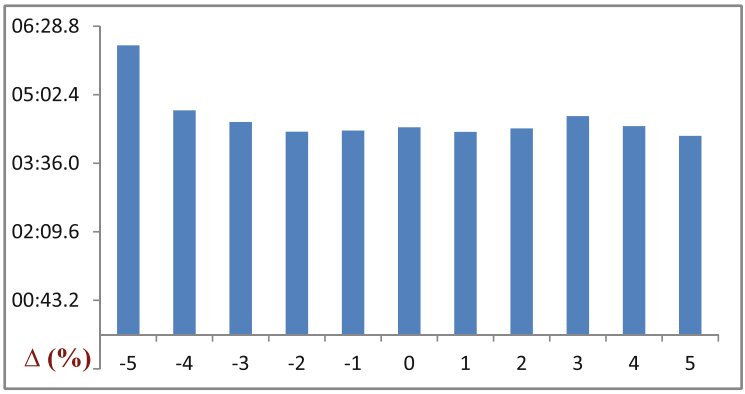

Fig. 5. The wall-clock times of ISPRING with various Δs

shorter, the min-max coefficients are changed more often. For the reason, the time warping distances are frequently recalculated. With other remaining values of Δ, their results are generally not better than the result of Δ of 0 % in terms of the *best-so-far* quality as well as the wall-clock-time, so the monitoring window should have the same length as the pattern.

6 Conclusions and Future Work

The paper has introduced a new method for finding new-coming subsequences in streaming time-series that are most similar to prespecified patterns under DTW. The proposed method, ISPRING, is an improvement of SPRING by applying incremental min-max normalization before the DTW calculation, so that obtained results are accurate. As regards each query pattern, a monitoring window is anchored at the entry of one streaming time series and then the min-max coefficients of the corresponding subsequence are extracted to infer the normalized subsequence. The experiments show that ISPRING can find *best-so-far* subsequences more accurate than those of SUCR-DTW, a state-of-the-art method of similarity search over time-series streams under DTW. The size of the monitoring window should be equal to the length of the pattern. Moreover, ISPRING can discover *best-so-far* subsequences whose lengths are different with those of patterns. Since whenever the min-max coefficients in the monitoring

window are changed, the time warping distances need incrementally recalculating. Consequently, the wall-clock time of ISPRING is relatively longer than that of SUCR-DTW.

In future work, we plan to improve ISPRING to find *best-so-far* subsequences whose lengths are reasonable to those of patterns. This means that the found *best-so-far* subsequences should not be too short in comparison to the lengths of patterns.

References

1. Bemdt, D., Clifford, J.: Using Dynamic Time Warping to find patterns in time series. In: Proceedings of AAAI Workshop on Knowledge Discovery in Databases, Seattle, Washington, USA, pp. 359–370 (1994)
2. Sakoe, H., Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process. **26**(1), 43–49 (1978)
3. Zhu, Y., Shasha, D.: Warping indexes with envelope transforms for query by humming. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 181–192 (2003)
4. Sakurai, Y., Faloutsos, C., Yamamuro, M.: Stream monitoring under the time warping distance. In: The IEEE 23rd International Conference on Data Engineering, Istanbul, Turkey, pp. 1046–1055 (2007)
5. Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., Keogh, E.: Searching and mining trillions of time series subsequences under Dynamic Time Warping. In: Proceedings of the 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2012), Beijing, China, pp. 262–270 (2012)
6. Giao, B. C., Anh, D.T.: Similarity search in multiple high speed time series streams under Dynamic Time Warping. In: Proceedings of the 2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), Ho Chi Minh City, Vietnam, pp. 82–87 (2015)
7. Gong, X., Fong, S., Chan, J., Mohammed, S.: NSPRING: the SPRING extension for subsequence matching of time series supporting normalization. J. Supercomput. 8, 1–25 (2015). doi:10.1007/s11227-015-1525-6
8. Kim, S.-W., Park, S.: An index-based approach for similarity search supporting time warping in large sequence databases. In: Proceedings of the 17th IEEE International Conference on Data Engineering, Heidelberg, Germany, pp. 607–614 (2001)
9. Keogh, E., Ratanamahatana, C.: Exact indexing of Dynamic Time Warping. Knowl. Inf. Syst. **7**(3), 358–386 (2004)
10. Weigend, A.: In Time series prediction: Forecasting the future and understanding the past. http://www-psych.stanford.edu/~andreas/Time-Series/SantaFe.html. Accessed December 2013
11. Keogh, E.: The UCR time series classification/clustering page. http://www.cs.ucr.edu/~eamonn/time_series_data/. Accessed August 2013