Elastic Non-contiguous Sequence Pattern Detection for Data Stream Monitoring

Xinqiang Zuo[1], Jie Gu[2], Yuanbing Zhou[1], and Chunhui Zhao[1]

[1] State Power Economic Research Institute, China
{zuoxinqiang,zhouyuanbing,zhaochunhui}@chinasperi.com.cn
[2] School of Software, Tsinghua University
guj05@mails.tsinghua.edu.cn

Abstract. In recent years, there has been an increasing interest in the detection of non-contiguous sequence patterns in data streams. Existing works define a fixed temporal constraint between every pair of adjacent elements of the sequence. While this method is simple and intuitive, it suffers from the following shortcomings: 1)It is difficult for the users who are not domain experts to specify such complex temporal constraints properly; 2)The fixed temporal constraint is not flexible to capture interested patterns hidden in long sequences. In this paper, we introduce a novel type of non-contiguous sequence pattern, named **Elastic Temporal Constrained Non-contiguous Sequence Pattern(ETC-NSP)**. Such a pattern defines an elastic temporal constraint on the sequence, thus is more flexible and effective as opposed to the fixed temporal constraints. Detection of ETC-NSP in data streams is a non-trivial task since a brute force approach is exponential in time. Our method exploits an similarity measurement called **Minimal Variance Matching** as the basic matching mechanism. To further speed up the monitoring process, we develop pruning strategies which make it practical to use ETC-NSP in streaming environment. Experimental studies show that the proposed method is efficient and effective in detecting non-contiguous sequence patterns from data streams.

1 Introduction

Sequence patterns, which may describe a meaningful tendency or an important phenomenon of the monitored objects, refer to a series of ordered elements. Sequence pattern detection in streaming environment is to find the data streams that contain the specified sequence patterns through subsequence matching. Given a distance function, a pattern is said to be contained in a data stream if the distance between the pattern and a subsequence of the data stream is within a predefined threshold.

Recently, there is an increasing interest in the monitoring of non-contiguous sequence patterns[2, 3], which can be obtained from a long sequence by discarding some irrelevant parts. Non-contiguous sequence pattern is of great importance since some applications require retrieving a specific ordering of events without

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
S_1	4	x	x	x	7	x	x	x	x	5
S_2	x	4	x	7	x	x	5	x	x	x
S_3	4	x	x	x	x	x	7	x	5	x

Fig. 1. Fixed Temporal Constraint on Non-contiguous Sequence Patterns

caring about the events which interleave them in the actual sequence[2, 3, 5]. As in the above example, previous work related to non-contiguous sequence patterns define a fixed temporal constraint on the adjacent elements of a sequence, i.e, every pair of adjacent elements in the sequence should satisfy a temporal interval. This mechanism is convenient for indexing and searching for interested sequence patterns, but suffers from the following shortcomings: 1)It is difficult, if not impossible, for users who are not domain experts to define such non-contiguous patterns properly. 2)Though non-contiguous sequence patterns may repeat themselves in a long sequence, it is not likely that they always follow the fixed temporal constraint in a strict way. Thus the fixed temporal constraint is not flexible to capture interested patterns hidden in a long sequence. Consider the situation in Figure.1, where a non-contiguous sequence S_n=(4, 7, 5) appears in 3 long sequences $\{S_1, S_2, S_3\}$ with different temporal intervals among its elements. Any fixed temporal constrained sequence pattern can only detect the occurrence of S_n in at most one of long sequences and omit the other two. And at least 3 non-contiguous sequence patterns has to be defined to detect all the occurrences of S_n. Obviously this strategy is ineffective in handling large amounts of sequence patterns and a natural requirement of detecting all the occurrences of the same sequence with one single pattern emerges.

In this paper, we introduce a novel type of non-contiguous sequence pattern, what we name **Elastic Temporal Constrained Non-contiguous Sequence Pattern(ETC-NSP)**. Such a pattern defines an elastic temporal constraint on the sequence, i.e, only the first and the last element of the sequence should satisfy a specified time-interval. This mechanism is more flexible and effective as opposed to the fixed temporal constraints.

Non-contiguous sequence pattern detection is not a trivial task. For a long sequence of length m, the number of its n-length non-contiguous subsequences is as many as C_m^n. A brute-force approach is exponential in time complexity since all the non-contiguous subsequence should be considered.

Motivated by be the requirement of elastic temporal constraint and the following challenge mentioned above, we exploit **Minimal Variance Matching(MVM)**[4] as the basic similarity model for efficient non-contiguous sequence pattern detection.

The contribution of this paper can be summarized as: 1)We propose the concept of elastic temporal constrained non-contiguous sequence pattern (ETC-NSP) in stream monitoring, which is more accurate and robust. 2)We reduce the computation complexity of ETC-NSP from $O(C_m^n)$ to $O(mn)$ by exploiting the

Minimal Variance Matching(MVM). 3)To further speed up the pattern detection process, we develop 2 pruning methods for ETC-NSP to make it feasible to be applied in stream monitoring.

The remainder of this paper is as follows: Section 2 describes related work on stream monitoring in the past years. A formal definition of ETC-NSP together with the corresponding MVM algorithm is provided in Section 3. We propose the pruning methods to speed up ETC-NSP in Section 4. Section 5 describes the experimental evaluation of ETC-NSP on different data sets, which shows the efficiency and effectiveness of ETC-NSP.

2 Related Work

The problem of sequence pattern detection has been studied extensively in the past years. In [1], a solution for similar sequence matching has been proposed, where the trick of Discrete Fourier Transformation(DFT) has been used. After the features of sequence be extracted by DFT, they are indexed by R-Tree. The method of Minimal Variance Matching(MVM) is developed in [4]. MVM handles the matching of sequences of different lengths, but it is only applicable in static environment.

The first work about non-contiguous sequence is proposed in [3], but only fixed temporal constraint was considered. Another important work about non-contiguous sequence is [2]. Indexing methods are used in both [3] and [2] to speed up the calculation process. In [6], the authors solved the problem of non-contiguous sequence patterns in both spatial dimension and temporal dimension. The sequence pattern monitoring problem is well studied in [7] under the notation of Dynamic Time Warping(DTW) distance, where elastic temporal constraint is not considered, either.

3 Non-contiguous Subsequence

3.1 Problem Formalization

We first give a formal definition of the non-contiguous subsequence.

Definition 1. ***Non-contiguous Subsequence(NCS)*** *Given a long sequence of length m:* $X=(X_1, X_2, ..., X_m)$*, where* X_i *is the* i_{th} *element of* X*. A non-contiguous subsequence of* X *is a sequence* $X^{'}=(X_{i_1}, X_{i_2}, ..., X_{i_n})(n \leq m)$*, where* $1 \leq i_1 \leq i_2 \leq ... \leq i_n \leq m$*, denoted as* $X^{'} \subset X$

Definition 2. ***Fixed Temporal Constrained Non-Contiguous Sequence Pattern(FTC-NSP)*** *A fixed temporal constrained non-contiguous sequence pattern is a non-contiguous sequence with fixed temporal constraints.* $P =< (P_1, t_1), (P_2, t_2), ..., (P_n, t_n) >$*. A long sequence* S *contains a FTC-NSP* P*, if there exists a non-contiguous sequence* $S^{'} =< (S_1^{'}, t_1^{'}), (S_2^{'}, t_2^{'}), ..., (S_n^{'}, t_n^{'}) >, S^{'} \subset$

S such that $\sum_{i=1}^{n} |P_i - S_i^{'}| < \epsilon$ [3] *and* $|t_i - t_{i-1}| = |t_i^{'} - t_{i-1}^{'}|$, $\forall i \in 2, ..., n$, *where* ϵ *is a given threshold.*

From the above definition, it can be found that every pair of the adjacent elements in a non-contiguous sequence should satisfy the specified temporal constraint. Though a non-contiguous sequence pattern may repeat itself in data streams, it is not likely that the time interval between its elements may remain unchanged all the time. Thus the fixed temporal constraint is not flexible and effective enough to capture interested patterns hidden in data streams.

Definition 3. ***Elastic Temporal Constrained Non-contiguous Sequence Pattern(ETC-NSP)*** *An elastic temporal constrained non-contiguous sequence pattern is a non-contiguous sequence* $P=(P_1, P_2, ..., P_n)$ *together with a temporal interval* t_p. *A long sequence* S *contains a ETC-NSP* P *if there exists a non-contiguous sequence* $S^{'} =< (S_1^{'}, t_1^{'}), (S_2^{'}, t_2^{'}), ..., (S_n^{'}, t_n^{'}) >, S^{'} \subset S$ *such that* $\sum_{i=1}^{n} |P_i - S_i^{'}| < \epsilon$ *and* $t_n^{'} - t_1^{'} \leq t_p$, *where* ϵ *is a given threshold.*

In ETC-NSP, the temporal constraint is more flexible since only the first and last element of the sequence should satisfy the temporal interval. Thus it is tolerant to time shift in the sequence and robust to noise, and is more powerful in capturing hidden patterns. Consider the example in Figure.1, the pattern in the 3 sequence can be detected by a ETC-NSP $(4, 7, 5)$ with a temporal interval 10.

3.2 ETC-NSP Detection in Streaming Environment

In this subsection, we discuss the problem of ETC-NSP detection in data streams. A data stream is a non-contiguous, ordered sequence of values $x_1, x_2, ..., x_n, ...$, where x_n is the most recent values[7]. Due to the semi-infinite characteristic of data stream, it is not feasible to monitor on the whole stream. We exploit a sliding window in the streaming environment, and search for non-contiguous subsequence in the window. We now define the distance between an ETC-NSP and a data stream.

Definition 4. ***Distance Between ETC-NSP and Data Stream*** *Given a data stream* S *and an ETC-NSP* P *with a temporal constraint* t_p. *We apply a sliding window* W *of length* $m=|P|+t_p$, *where* $|P|$ *is the length of* P, *on the data stream.* $W^{'}$ *is a non-contiguous subsequence of length* $|P|$ *in* W, *and* $\mathcal{W}$ *is the set of all the non-contiguous subsequence of length* $|P|$ *in* W. *The distance between* S *and* P *can be defined*

$$D(S, P) = \min\{\mathcal{D}_{L_1}(P, W^{'}), W^{'} \in \mathcal{W}\}$$

[3] Note that any L_p distance can be used here. Without the loss of generality, we exploit the L_1 distance

From the above definition, we can see that the distance between a sequence pattern and a data stream is the minimal value of the L_p distances between the sequence pattern and all the subsequences in a sliding window applied on the data stream. If the pattern sequence is seemed to be a query, pattern detection in data stream can be considered as a range query process.

Definition 5. ***Non-contiguous Subsequence Based Range Query*** *Given a data stream S, an ETC-NSP P with a temporal constraint t_p and a threshold ϵ. When applying a sliding window W of length m=$|P|$+t_p on the data stream, the range query process is to determine whether the distance between S and P is within ϵ, i.e., $D(S, P) < \epsilon$*

4 Proposed Method

4.1 Naive Approach

It can be found from Definition.4 that the distance between a sequence pattern and a data stream is in essence the distance between the sequence pattern and the sliding window applied on the data stream, which is in the form of a long sequence. Given a long sequence X of length m and a pattern sequence P of length $n(n \leq m)$, the distance between X and P is the minimum L_p distance between P and all the non-contiguous subsequences of X.

Lemma 1. *For a data sequence X of length m, there are C_m^n non-contiguous subsequences of X whose lengths are $n(n \leq m)$.*

A brute-force approach to calculate $D(X, P)$ is to enumerate all the non-contiguous subsequences of X, for each subsequence $X^{'}$, calculate the L_P distance between $X^{'}$ and P. The minimum $D_{L_p}(X^{'}, P)$ is the distance between $X^{'}$ and P. Without loss of generality, we exploit L_1 distance in the rest of this paper. This simple approach is rather straightforward and understandable, but is precluded due to its extremely expensive time cost.

4.2 Minimal Variance Matching

Since the naive method to calculate the distance is rather expensive in terms of time cost, we exploit a more efficient technique, the **Minimal Variance Matching(MVM)**, which outperforms the naive method in a magnitude of time (from $O(n * C_m^n)$ to $O(m * (m - n))$. MVM is first proposed in [4] and considered as a directed acyclic graph problem. In this paper, we formulate this technique in a more formal dynamic programming manner.

We first introduce the notion of non-contiguous path. Let M be a $m \times n$ matrix of pairwise distances between elements of X and P, i.e., $M[i][j]$=$|X_i - P_j|$. A non-contiguous path $P_{nc} = (m_1, m_2, ..., m_n)$ is a sequence of n matrix cells, $m_k = M[i_k][k], k \leq i_k \leq (k + m - n)$ and $i_k < i_{k+1}$. Any non-contiguous path determines an alignment between X and P, thus there are total C_m^n different

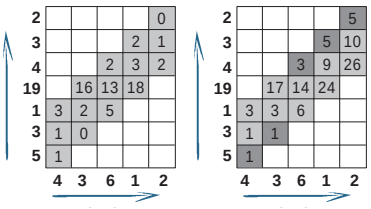


Fig. 2. Example of the MVM Algorithm

non-contiguous paths for X and P. The distance between X and P corresponds to the non-contiguous path that has the minimum sum of matrix cells, i.e., $D(X,P) = \min_{P_{nc}}\{\sum_{i=1}^{n} m_i, m_i \in P_{nc}\}$. The algorithm of finding such a non-contiguous path is described in Algorithm.1.

Algorithm 1 Calculate non-contiguous subsequence distance

Input: Long sequence $X = (X_1, ..., X_m)$ and pattern sequence $P = (P_1, ..., P_n), n \leq m$
Output:non-contiguous subsequence distance between X and P

for $j = 1$ to n **do**
  for $i = j$ to $j + m - n$ **do**
    M[i][j]=$|X_i - Y_j|$
  end for
end for
for $j = 2$ to n **do**
  for $i = j$ to $j + m - n$ **do**
    M[i][j]+=$\min_{j-1 \leq k \leq i-1}\{M[k][j-1]\}$
  end for
end for
return $\min_{j \leq k \leq j+m-n}\{M[k][j]\}$

Now we provide a running example in Figure.2 to illustrate the method in more detail. Consider the two sequences X=(5, 3, 1, 19, 4, 3, 2) and Y=(4, 3, 6, 1, 2), we show how to obtain $D(X, Y)$ with our proposed algorithm. The left table in Figure.2 shows the initial state of the matrix M. In the right table, the series of cells with shadow is corresponding to the non-contiguous path and the minimal value 5 in the right-most column is the MVM distance between X and Y. That is to say, among all the subsequences of X, $X^{'}$=(5, 3, 4, 3, 2) has the minimal L_1 distance 5 with Y. This algorithm is based on a dynamic programming technique and has a time complexity of $O(m * (m - n + 1))$, which is a great improvement as opposed to the naive method. One important property of the MVM distance we will use later is that if the two sequences being calculated are of the same lengths, then MVM distance is identical to L_1 distance.

5 Pruning Methods in Streaming Environment

Although the method in last section already outperforms the naive approach by orders of magnitude, a time complexity of $O(m*(m-n+1))$ is still needed. Thus it is still impractical in streaming environment where data elements are coming continuously at high rate. We propose two techniques in this section to further speed up the monitoring process and make it feasible to be used in streaming environment.

5.1 Triangle Inequality

In metric space, there exists a triangle inequality which is used as a pruning tool in the search process, that is: $D_{L_p}(X,Y) + D_{L_p}(Y,Z) > D_{L_p}(X,Z)$. In this subsection, we prove that the MVM distance also follows a triangle inequality which is different from that in the metric space.

Theorem 1. *triangle Inequality Theorem: Given three data sequence Q, S and R. Q is a long sequence of length m. S and R are short sequences of length n. The following inequality holds: $D(Q,S) + D(S,R) > D(Q,R)$.*

5.2 Lower Bound Distance

In this subsection we propose a lower bound distance of the MVM distance. Suppose that $P=(P_1, P_2, ..., P_n)$ is an ETC-NSP with a temporal constraint t_p and a threshold ϵ, and $W=(W_1, W_2, ..., W_n, ..., W_{n+t_p})$ is a sliding window on a data stream S. We define the data structure $W^L=(W_1^L, W_2^L, ..., W_n^L)$ and $W^U=(W_1^U, W_2^U, ..., W_n^U)$, where

$$W_i^L = \min\{W_i, .., W_{i+t_p}\}, W_i^U = \max\{W_i, .., W_{i+t_p}\}$$

We can derive a lower bound distance $D_{lb}(P,W)$ of the MVM distance between W and P from the proposed W^L and W^U,

$$D_{lb}(P,W) = \sum_{i=1}^{n} g(P_i, W_i^L, W_i^U)$$

$$g(P_i, W_i^L, W_i^U) = \begin{cases} |P_i - W_i^L| & \text{if } P_i \leq W_i^L \\ |P_i - W_i^U| & \text{if } P_i \geq W_i^U \\ 0 & \text{otherwise} \end{cases}$$

The proposed lower bound distance $D_{lb}(P,W)$ can be calculated in a linear time and used as a pruning tool. When $D_{lb}(P,W) > \epsilon$, it is straightforward that $D(P,W) > \epsilon$ since $D(P,W) > D_{lb}(P,W)$. Then it is unnecessary to calculate $D(P,W)$ explicitly.

5.3 Complexity Analysis

Compared with the naive method, the novel algorithm uses a matrix to reach the lower bound distance. Given m the length of the query data and n the length of the incomplete data, the time complexity of this method is $\mathcal{O}(n(m-n+1))$. As can be found directly from the running example, for each of the n columns in the matrix, $m-n+1$ cells has to be visited in our algorithm. Thus this method is especially efficient when m and n are close. One extreme case is when m is the same as n, our algorithm is identical to the L_1 distance and the time complexity has been reduced to $\mathcal{O}(m)$. That is, since no value is missing, the lower bound of the L_1 distance is just itself. Another extreme case is when n is much smaller than m, i.e., a large amount of elements is missing, the time complexity is as large as $O(mn)$. The two pruning methods further reduce the time complexity since it is not necessary to really conduct MVM calculation for all the data.

6 Experiment Evaluation

To evaluate the effectiveness of our method, we carried out experiments on both real and synthetic data sets. Our experiments were conducted on an Intel CPU of 1.7GHz with 512 MB of Memory. The programming language we exploited was Java(JDK 1.5).

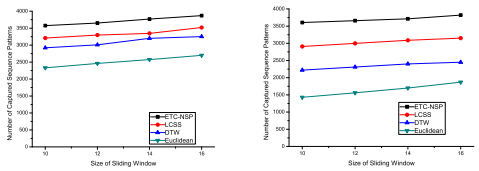
6.1 Data Set

- Synthetic Data Set: The data set consists of a synthetic data stream, which contains 4000 predefined non-contiguous sequence patterns interleaved by white noise. The noise and the values of the sequence patterns follow the same Gaussian distributions.
- S&P 500 Data Set: This data contains stock information about 500 companies for about 250 days, including the prices of all the 500 stocks in the form of time series.[4]

6.2 Capture of Non-contiguous Sequence Patterns

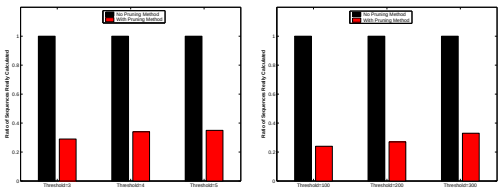
We present the power of our method in detecting of non-contiguous sequence patterns in data streams in this subsection. We generate two data sets with different Gaussian distributions, i.e, with $< mean = 10, variance = 2 >$ and $< mean = 10, variance = 5 >$ respectively. in Figure.3, we compare the number of sequence patterns captured by different similarity measurements, including LCSS, DTW, Euclidean distance, and the proposed ETC-NSP based MVM. For LCSS, it is needed that a parameter has to be specified to judge whether two real-valued data elements are "equal". The perform of LCSS will vary according to this parameter. We show the best performance of LCSS in the figures. ETC-NSP

[4] http://biz.swcp.com/stocks/



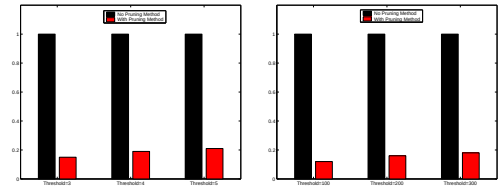
(a) Mean=10,Variance=2 (b) Mean=10,Variance=5

Fig. 3. Power of Pattern Detection



(a) Synthetic Data Set (b) S&P 500 Data Set

Fig. 4. Effectiveness of Pruning with Lower Bound



(a) Synthetic Data Set (b) S&P 500 Data Set

Fig. 5. Effectiveness of Pruning with triangle Inequality

outperforms other technique in terms of number of sequence patterns captured. When the variance of the distribution decreases, the performance of LCSS, DTW and Euclidean degrades, but ETC-NSP is influenced slightly by this change. This shows that ETC-NSP is robust to noise when applied to monitor sequence patterns. The elastic temporal constraint allows it to avoid the noise element in the data stream automatically. DTW and Euclidean distance are sensitive to noise, so they dismiss many sequences in the presence of noise.

6.3 Pruning Power

The proposed triangle inequality and the lower bound can be used to pruning unqualified data sequences in the monitoring process. The measurement for its effect is pruning power, which can be defined as follows:

$$P = \frac{NumberofObjectsDoNotRequireFullComputation}{NumberofObjectsInDataset}$$

We combine the lower bound and the triangle inequality to prune unqualified sequence data in the monitoring process. The pruning power on both the data sets is shown in Figure.4 and Figure.5. When the threshold to judge whether a sequence is a defined pattern increases, the pruning power of method decrease. This is because that if the calculated lower bound is not larger than the threshold, it is not permitted to dismiss the sequence data without conduct the MVM algorithm.

7 Conclusion

We propose the problem of elastic temporal constrained non-contiguous subsequence detection in data streams. The MVM distance is used to evaluate the similarity between sequence patters and data in streaming environment. To further speed up MVM, we develop two pruning methods, i.e, the triangle inequality and the lower bound to make it feasible in streaming environment.

References

1. Christos Faloutsos, M Ranganathan, Yannis Manolopoulos. *Fast Subsequence Matching in Time-Series Databases*. In Proceedings 1994 ACM SIGMOD Conference, Mineapolis, MN.
2. Nikos Mamoulis and Man Lung Yiu. *Non-contiguous Sequence Pattern Queries*. In International Conference on Extending Database Technology,2004.
3. Haixun Wang, Chang-Shing Perng, Wei Fan, Sanghyun Park and Philip S.Yu. *Indexing Weighted-Sequences in Large Databases*. In Proceedings of 19th International Conference On Data Engineering, 2003.
4. L.J.Latecki, V. Megalooikonomou, Q.Wang, R. Lakaemper, C.A. Ratanamahatana, and E. Keogh. *Elastic Partial Matching of Time Series*. In European Conference on Principles and Practice of Knowledge Discovery in Databases, 2005.
5. Marios Hadjieleftheriou, George Kollios, Petko Bakalov, Vassilis J.Tsotras. *Complex Spatio-Temporal Pattern Queries*. In Proceedings of the 31st International Conference on Very Large Data Bases, 2005.
6. Marios Hadjieleftheriou, George Kollios, Vassilis J. Tsotras and Dimitrios Gunopulos. *Efficient Indexing of Spatiotemporal Objects*. In International Conference on Extending Database Technology, 2002.
7. Yasushi Sakurai, Christos Faloutsos and Masashi Yamamuro. *Stream Monitoring under the Time Warping Distance*. In International Conference on Data Engineering(ICDE),2007.