MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Finding Multidimensional Patterns in Multidimensional Time Series

Laftchiev, E.; Liu, Y.

TR2018-044 June 30, 2018

Abstract

Exact pattern matching is a method of localizing arbitrarily sized patterns in time series data. To date, the problem of exact pattern matching has only been fully addressed for one query pattern on one time series with a single best match location. This paper addresses the broader problem of finding the top-K pattern matches for a multidimensional time series pattern in a large multidimensional time series. The problem is addressed in two stages using an algorithm that combines ideas from the fields of data mining and bi-clustering. The first stage of the algorithm addresses selecting the dimension subset that matches the query pattern and locating the matching pattern. The second stage of the algorithm addresses the problem of finding the top-K matches of the pattern in the selected time series dimensions. The performance of the proposed algorithm is evaluated against the best single dimensional exact pattern matching algorithm on real and simulated data.

SIGKDD Workshop on Mining and Learning From Time Series

Finding Multidimensional Patterns in Multidimensional Time Series

Emil Laftchiev
Mitsubishi Electric Research Labs
Cambridge, MA
laftchiev@merl.com

Yuchao Liu
University of California San Diego
La Jolla, CA
yul085@ucsd.edu

ABSTRACT

Exact pattern matching is a method of localizing arbitrarily sized patterns in time series data. To date, the problem of exact pattern matching has only been fully addressed for one query pattern on one time series with a single best match location. This paper addresses the broader problem of finding the top-K pattern matches for a multidimensional time series pattern in a large multidimensional time series. The problem is addressed in two stages using an algorithm that combines ideas from the fields of data mining and bi-clustering. The first stage of the algorithm addresses selecting the dimension subset that matches the query pattern and locating the matching pattern. The second stage of the algorithm addresses the problem of finding the top-K matches of the pattern in the selected time series dimensions. The performance of the proposed algorithm is evaluated against the best single dimensional exact pattern matching algorithm on real and simulated data.

CCS CONCEPTS

• **Information systems → Data mining**;

KEYWORDS

multidimensional time series, similarity search, dimension reduction, top-K search

ACM Reference Format:
Emil Laftchiev and Yuchao Liu. 2018. Finding Multidimensional Patterns in Multidimensional Time Series. In *Proceedings of 4th SIGKDD Workshop on Mining and Learning from Time Series (KDD Workshop on MiLeTS'18).* ACM, New York, NY, USA, Article 4, 9 pages.

1 INTRODUCTION

The exponential growth of time series data that are collected via cheap and widely available sensors is revolutionizing the scientific field of time series data mining with applications in predictive maintenance and anomaly detection. A important problem in this field is the localization of abnormal time series patterns or time series subsequences.

In many applications the shape of these abnormal patterns is known, yet it is difficult to locate these patterns in very long time series. For example, suppose that an engineer monitoring a plant can recognize an anomalous pattern from a prior failure. To learn from the prior failure, the engineer may try to localize the anomalous pattern in stored time series data from the plant sensors. However, because the plant log is long, sometimes stretching months or years, it is difficult for the engineer to find the exact time at which the prior failure occurred.

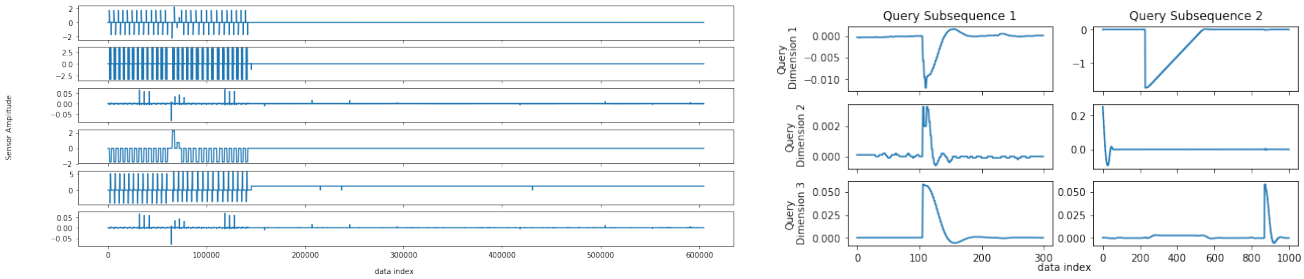
To address the problem of localizing one pattern in one long time series, researchers have often used classical data mining tools like feature extraction and indexing. Both fail as the data size scales. Feature extraction fails because the uniqueness of the extracted features is strongly dependent on the signal-to-noise ratio of the data and feature uniqueness diminishes as the data set grows [9]. Indexing fails because the growth of the index structure becomes unwieldy as the size of the data grows.

Thus researchers have focused on methods of exact pattern localization. The UCR suite [11] is an example of a scalable algorithm that is capable of localizing patterns in very long time series. The key to this approach is realizing that the best metric to compare two subsequences is Dynamic Time Warping (DTW) [12] and that this metric can be approximated by lower bounds to speed up the localization process. Because these lower bounds increase in tightness as their computational complexity scales, nesting them leads to a significant reduction in the computational cost of searching for a pattern.

To date, the problem of fast exact pattern matching for time series data has only been fully explored in the one dimensional case. That is, for the localization of one pattern within one time series. Thus there are two interesting extensions of this problem that are explored in this paper. First, it is of interest to explore the localization of a known multidimensional pattern in a multidimensional time series. That is, the localization of several patterns with a common time window across several time series. Second, it is of interest to find the top-K matches of a known pattern in one and multidimensional time series data.

Searching for multidimensional patterns is important because while a single pattern may appear at multiple instances on one time series, it's more likely that an anomaly can be uniquely identified when multiple patterns across multiple time series coincide. As an example, consider the Inertial Measurement Unit (IMU) data shown in Fig. 1a. Such data can be readily collected from mobile phones, and fitness trackers. The figure shows six representative time series. The top three time series show estimates of bias on the measurements, and the bottom three time series show angle measurements. Note that in the first 150,000 data points the device appears to be moving in a predictable pattern, and in the next 450,000 data points the device appears to have settled into a steady state. Hence, the normal behavior for this device is steady, but with times of periodic motion. Importantly, during periodic motion, the device sometimes records bias errors in its measurement. A multidimensional subsequence containing the spike in bias error is shown Query Subsequence 1 in Fig. 1b.

It is important to note that the anomalous multidimensional pattern shown in Query Subsequence 1 has three patterns whose start points occur simultaneously across multiple dimensions. However, the patterns may not have such coinciding start points but may



(a) Six time series recorded from an IMU.

(b) Anomalies in IMU Bias Measurements.

Figure 1: Real IMU Data

generally share the same time window. This is shown in Query Subsequence 2 in Fig. 1b. To illustrate this point the experiments in this paper use both Query Subsequence 1 and Query Subsequence 2.

Also note that in Fig. 1 the number of query dimensions is smaller than the number of data dimensions. This is because the dimensions of an anomalous multidimensional pattern are often smaller than the total number of available dimensions. Thus any algorithm localizing a multidimensional pattern may also need to act as a dimension reducing algorithm.

1.1 Paper Goal and Contributions

Formally, given a multidimensional time series of dimension M with length N, and a multidimensional query of dimension $m \leq M$ with length n, where $n \ll N$, locate K time intervals $[t, t+n]_k$ that most closely match the given multidimensional query pattern.

This paper proposes a new algorithm to solve the problem of finding a multidimensional pattern in a multidimensional time series. The algorithm is based on [11] and a bi-clustering algorithm named Largest Average Submatrix (LAS) [15]. The combined algorithm is further extended to find the top-K matches. This paper makes the following contributions:

- A novel distance measure that evaluates the closeness between multidimensional time series queries and data sequences in the presence of arbitrary dimension alignment.
- An approximation for the proposed distance measure to reduce its computational complexity and speed up sequential search.
- An iterative algorithm to find the pattern match in the case when there are more data time series sequences than queries. This algorithm is an efficient time series selection method.
- An algorithm for finding the top-K matches in the one and multidimensional cases.

2 PAPER NOTATION

The following is the notation used throughout this paper.

- Q: The collection of query patterns, denoted as a sequence of patterns $\{q_1, \ldots, q_m\}$. Here referred to as a multidimensional query;
- q_i: The i-th pattern in a query;
- P: The collection of candidate matching patterns extracted using a sliding window in time series data, denoted as a sequence of patterns $\{p_1, \ldots, p_m\}$. Here referred to as a multidimensional candidate pattern match;
- m: The total number of query patterns;
- n: The length of a single query pattern;
- M: The total number of data time series;
- N: The length of a single data time series;
- $[M]$: The integer set $\{1, \ldots, M\}$. Similarly we have $[m]$, etc;
- X: The collection of data time series. Here referred to as a multidimensional time series;
- X_i: The i-th data time series. $X_i(t)$ stands for the value of time series i at time stamp t;
- $X|_I$: The collection of $\{X_i\}$ with $i \in I$;
- $X|_t$: The data of X within the time window $[t, t+n]$, which is the collection of time series $\{X_i(t), \ldots, X_i(t+n)\}$, with $i \in [M]$;
- $X|_{I,t}$: The data of $X|_I$ within the time window $[t, t+n]$, which is the collection of time series $\{X_i(t), \ldots, X_i(t+n)\}$, with $i \in I$;

2.1 Algorithm Assumptions

The algorithm presented in this paper makes 2 assumptions:

- The subsequence which matches each of the query patterns appears at the same time stamp in the data across the multidimensional time series. Formally, there exists a pair (I, t) such that $X|_{I,t}$ is a close match to Q.
- The query patterns on each dimension are distinguishable with each other ($\text{DTW}(q_i, q_j)$ is well bounded away from zero for all pairs of (i, j)).

The first assumption simply states that there exists a time window, that is common among a subset of the time series dimensions, that contains the query subsequences of interest. The second assumption states that the patterns on separate query dimensions are distinct and their normalized sum does not equal zero. This assumption facilitates distance approximations that reduce computational complexity, and enables us to propose a top-K localization algorithm capable of finding patterns for one and multidimensional data. In addition, the uniqueness of patterns helps to simplify challenging combinatorial problems that are discussed in the sections to come.

3 RELATED WORK AND BACKGROUND

3.1 Dynamic Time Warping

Dynamic time warping (DTW) is an algorithm measuring the similarity between two time series that is particularly useful in detecting similarities when the patterns are stretched, compressed, skewed, or slightly misshaped, i.e. the DTW will provide a good distance measure of the intuitive understandings that humans have when comparing patterns. Unfortunately DTW has polynomial computational complexity in the size of the time series pattern. To search for a pattern of fixed length subsequence in a large time series, it is necessary to examine as many fixed windows as the length of the time series. Therefore an exhaustive search for a pattern match using the DTW has a very high computational cost.

A large body of work has been dedicated to speeding up the DTW computation. An example of this can be found in [13, 14] which aim to approximate the DTW distance efficiently and accurately by eliminating unpromising matching profiles in the DTW matching matrix. Another example is the UCR Suite [11]. This algorithm is an optimized approach to leveraging all prior research in DTW search speed optimization. Here lower bounds, combined with early abandonment, are nested with respect to their computational complexity. Low computational complexity bounds are computed first, and the most complex lower bounds are computed last. The final calculation is the optimized DTW. Two specific lower bounds are argued to be optimal in [11] via the creation of a Pareto front between bound tightness and bound computational complexity: the Kim lower bound [8] and the Keogh lower bound [4, 7].

3.2 Evaluating the DTW in Multiple Dimensions

There are three methods of extending the DTW to multiple dimensions: match multiple time series using the same warping path inside the DTW matrix, a (weighted) sum of the DTW distance along multiple dimensions, and DTW using multidimensional points. The first case, when the DTW warping path is optimized over all dimensions of the multidimensional time series is proposed by Holt et. al. [17]. The second and third cases are proposed by Shokoohi-Yekta et. al. [16], and then later by Górecki and Luczak [6] who build a classifier using a weighted sum of the two DTW candidates.

3.3 Bi-Clustering

Bi-clustering is a widely-studied problem in the fields of bioinformatics and statistics. In this field, a large data matrix is observed and the goal is to find a subset of row indexes and a subset of column indexes such that the submatrix constructed by these indexes, has an anomaly or useful information. A common example is an interaction map X with chromosomes and different environments, where each entry X_{ij} represents the expression level of chromosome i under environment j. In this example, it is desirable to identify a subset of chromosomes that are over-expressed in certain environments (e.g. identify cancer genes and its associated cancerous environments).

The bi-clustering problem and popular algorithms that solve it can be found in Cheng et. al. [3] and Pontes et. al. [10]. A widely applied algorithm in bi-clustering is the Largest Average Submatrix (LAS) algorithm, developed by Shabalin et. al. [15], and tested on real and simulated data in [1, 2, 15].

The LAS algorithm aims to find the submatrix with the largest entry average among all the submatrices given the number of rows and columns. To find this submatrix, the algorithm randomly selects a fixed set of rows from among the available matrix rows. The selected rows are summed column wise. The sums are sorted, and largest column sums (known number) are identified. Having found the largest column sums, the column indices are fixed. Next a row sum is performed along the fixed columns. Based on the resulting row sum, a new set of rows is selected with several largest row sums. This iteration is repeated until no update is performed on the row and column indices with respect to the last iteration.

The LAS algorithm has several known drawbacks. First, the random initialization of the algorithm makes convergence analysis difficult. The best work analyzing this approach to date is by Gamarnik and Li [5]. Second, in some cases the algorithm converges to local minima where the objective error is high. In such cases the algorithm is re-initialized and a new answer is obtained. Convergence to local minima is sufficiently rare that this re-initialization can be automated via a tunable threshold on the algorithm output error without significantly affecting the average algorithm run time.

4 ALGORITHM DEVELOPMENT

The first step in developing a localization algorithm is to develop a generalized sequence matching distance. In this paper, no knowledge is assumed about the permutation of the time series with respect to the query subsequence dimensions. This removes a significant assumption made in prior work [6, 16, 17].

Thus the focus here is to propose a new distance metric for multidimensional patterns that is effective in its **discovery power** (a term frequently used in the statistics community), the ability to find matching patterns and discard erroneous matches, and effective with respect to the computational requirements.

4.1 A Distance Metric Invariant to Permutations

The matching metric should satisfy the following properties:

- $d(Q, P) \geq 0$. $d(Q, P) = 0$ if and only if P is equivalent to Q up to a row permutation.
- $d(Q, P) = d(Q, P') = d(Q', P)$, where P' and Q' denote row permutations of P and Q, respectively.
- $d(Q, P)$ is close to 0 when P and Q consist of similar patterns but never exactly 0 due to measurement noise and approximate query specification.

The first two properties imply that information regarding the permutation of the time series with respect to the query will be unknown a-priori. The third property acknowledges that the query pattern will only be known up to a certain inherent noise level in the time series data. Given these stipulations, we propose the following distance measure which is built on the DTW distance measure between two time series patterns in one dimension.

$$d(Q, P) = \min_{\pi \in \Pi} \sum_{i=1}^{m} \mathrm{DTW}(q_i, p_{\pi(i)}). \tag{1}$$

Here q_i and p_i have the same length which means we cannot match the query patterns to more than one dimension and v.v., meaning a single pattern in either the query or the candidate matching location cannot be matched to more than one dimension in the other. The set Π is the set of all possible permutations of the dimensions $[m]$, i.e all possible permutations of the sequence $\{1, \ldots, m\}$. The permutations are denoted with π and the result of the permutation is $\{\pi(1), \ldots, \pi(m)\}$. This definition satisfies the requirement that the distance be neutral with respect to the permutations of the dimensions because it sequentially searches through all possible permutations to find the most likely ordering. The optimal permutation is returned with the distance.

4.2 Approximate the distance

The distance measure defined in eq. (1) has maximum discovery power because it must yield the smallest possible distance for any pair of multidimensional patterns. However, this distance has a high computational cost. For example, for m query dimensions each calculation of this distance would require $m * m!$ calculations each with complexity $O(n^2)$. Similar to [11], we want to develop approximations to speed up a sliding window search with this distance. We propose the following approximation is proposed:

$$\hat{d}(Q, P) = \sum_{i=1}^{m} \min_{j} \mathrm{DTW}(q_i, p_j). \tag{2}$$

The approximated distance in eq. (2) is a trade-off of discovery power for computational efficiency. Here the worst case number of calculations is on the order of m^2, a fraction of the original computational complexity. However, because distance is found discretely over each pattern dimension, some dimensions may be over matched, and thus $\hat{d}$ is a lower bound of d, i.e. for any Q and P, $\hat{d}(Q, P) \leq d(Q, P)$. This means the discovery power of this approximated distance is reduced.

As an example, note that the calculation in eq. (2) moves the summation to the left of the minimization term, then for each row of the query, the dimension index of the candidate pattern match which minimizes the DTW distance is determined. Importantly, the candidate pattern dimension chosen to correspond for one dimension of the query is not eliminated when searching over the next dimension of the query. The lack of elimination of chosen dimensions leads to the lower bounding nature of the approximation.

Concretely, consider the case where $m = 2$ and the query consists of two identical subsequences each on its own dimension. The candidate matching sequence P has one dimension on which the time series sequence matches the query sequence well, and another dimension which is significantly different from the query sequence. On this proposed match, the original distance measure, d, will return a distance greater than zero because the second dimension of P will necessarily be included in the final matching permutation, i.e. permutations such as {1,1} or {2,2} are not allowed. However, the relaxed distance $\hat{d}$ will return a distance close to zero because both query dimensions will simply choose the candidate sequence dimension that matches to them. This example motivates the requirement that query patterns are each distinguishable from one another, i.e. the distances between queries are well bounded away from zero, such that the approximated distance provides a meaningful lower bound to the true proposed distance.

To discern the accuracy of the lower bound, a second approximation, $\tilde{d}$, is proposed. This approximation is an upper bound to equation (1), and is a greedy approximation to the proposed distance. The greedy search is shown in Alg. 1. Alg. 1 begins by finding the best match to the first query pattern, q_1 among p_j. This match is denoted as p_{j_1}. Then the next best match between the second query pattern, q_2, and the rest of p_j excluding p_{j_1} is found. The procedure is iterated until all dimensions of the query pattern are matched. The determined matching distance is the cumulative sum of the distances found at each step in the match above. The drawback of this bound is that it is dependent on the original order of the query which means it can be loose. Both approximations, $\hat{d}$ and $\tilde{d}$ are aimed at improving the speed of sliding window search by reducing the combinatorial cost of the problem.

Algorithm 1 Calculate $\tilde{d}$

Input: Query sequences Q, time series sequences P
Output: $\tilde{d}(Q, P)$

```
Initialization:
J ← [m]
d̃(Q, P) ← 0
LOOP Process:
for i = 1 to m do
  minD ← ∞
  for j ∈ J do
    d ← DTW(q_i, p_j)
    if (d < minD) then
      r ← j
      minD ← d
    end if
  end for
  Remove r from J
  d̃(Q, P) ← d̃(Q, P) + minD
end for
return d̃(Q, P)
```

5 MAIN ALGORITHM

Recall the stated problem, given a multidimensional query pattern Q of size m with length n in each dimension, and multidimensional time series X with length N with M component dimensions (time series), we want to find a location within the time series (time stamp) t^*, and a subset of the time series I^* of $[M]$ with cardinality m, such that

$$d(Q, X|_{I^*, t^*}) = \min_{I, t} d(Q, X|_{I, t}). \tag{3}$$

Here $|I| = m$, $X|_{I,t}$ is of the same time length and of the same number of dimensions as Q. The minimum is found jointly over all subsets of $[M]$ that are of size m and all possible time index locations in the time series.

This problem is similar to the bi-clustering problem because the multidimensional time series X can be viewed as a large matrix with rows corresponding to the time series dimension, and columns corresponding to the time series time indices. Then each $X|_{I,t}$

is a submatrix with row size m, column size n, and the problem presented in this paper is to locate the submatrix that is closest to the query matrix Q. This can be solved using an iterative strategy.

(1) Fix I^*, find t^* corresponding to $\min_t d(Q, X|_{I,t})$;
(2) Fix t, find a new I^* corresponding to $\min_I d(Q, X|_{I,t})$;
(3) Repeat step 2 until the subset I^* no longer changes.

The algorithm has converged when the subset I^* does not change between successive iterations. This algorithm returns the subset and order of $[M]$ which corresponds to the query pattern dimensions. To reduce the computational cost, we employ the approximations of the matching distance discussed in Sec. 4: the lower bound to d, $\hat{d}$, and the upper bound to d, $\tilde{d}$. These approximations are inserted as follows.

(1) Fix I^*, find t^* corresponding to $\min_t \hat{d}(Q, X|_{I,t})$;
(2) Fix t, find a new I^* corresponding to $\min_I \tilde{d}(Q, X|_{I,t})$;
(3) Repeat step 2 until the subset I^* no longer changes.

5.1 Step 1: Searching for Pattern Location on a Subset of Dimensions

The first step is to identify the correct window, t, within a subset of the multidimensional time series dimensions that minimize $\hat{d}(Q, X|_{I,t})$. To find t, use a sliding window and for each window evaluate $\hat{d}(Q, X|_{I,t})$ where t represents the starting time index and the length of the pattern ranges from t to $t + N - 1$. The window with the minimum value of $\hat{d}$ is the optimal time index t^* for this subset of time series.

The computational complexity of finding t^* using the DTW time series distance can be further reduced by cascading lower bounds [11]. Building on prior results that showed a Pareto front between lower bound tightness and computational complexity, here we extend two bounds into the multidimensional case: the Kim bound [8] and the Keogh bound [4, 7]. The Kim bound is a lower bound of $\hat{d}(Q, P)$ and is extended as,

$$LB_{KimMulti}(Q, P) = \sum_{i=1}^{m} \min_j LB_{Kim}(q_i, p_j). \quad (4)$$

The Keogh bound accounts for points in both patterns that deviate more than an envelope about one pattern defined by the Sakoe-Chiba DTW band width. The Keogh bound can be extended as,

$$LB_{KeoghMulti}(Q, P) = \sum_{i=1}^{m} \min_j LB_{Keogh}(q_i, p_j). \quad (5)$$

To cascade these multidimensional bounds first calculate the extended Kim bound (eq. (4)). If the result exceeds the best matching distance found so far then eliminate this window. If the result is smaller than the best matching distance so far, then calculate the extended Keogh bound (eq. (5)). If this result is greater than the best matching distance so far, then discard this window as a possible match. If the distance is smaller than the best matching distance so far, then we perform the DTW calculation. For any of these calculations, if the partially calculated distance exceeds the best DTW matching distance found so far, then discard the current matching window. This algorithm is shown in Alg. 2.

Algorithm 2 Algorithm for time stamp search

Input: Query sequences Q, time series sequences X
Output: t^*
Initialization:
Do z-normalize by row on Q
Construct upper and lower envelope by row on Q for calculating LB_{Keogh}
$minD \leftarrow \infty$
$lowerBnd \leftarrow 0$
LOOP Process:
for $t = 1$ to $N - n$ **do**
  $lowerBnd \leftarrow LB_{KimMulit}(Q, X|_t)$
  if ($lowerBnd < minD$) **then**
    $lowerBnd \leftarrow LB_{KeoghMulti}(Q, X|_t)$
    if ($lowerBnd < minD$) **then**
      $lowerBnd \leftarrow \hat{d}(Q, X|_t)$
      if ($lowerBnd < minD$) **then**
        $t^* \leftarrow t$
        $minD \leftarrow lowerBnd$
      end if
    end if
  end if
  $lowerBnd \leftarrow 0$
end for
return t^*

5.2 Step 2: Iterating the Search Dimensions

The first matching time instant t^* is determined on a randomly chosen subset of m dimensions of the multidimensional time series. After t^* is determined, the dimension subset choice is revisited so that a better subset can be found. The choice of dimension subset is revisited after each iteration of the first step until the subset of dimensions does not change between successive iterations.

More concretely, for all iterations following the initialization, the next set of time series is chosen by minimizing the approximated distance shown in Alg. 1. This means that for a fixed t on a given set of I^*, we find a new set I^* that corresponds to $\min_I \tilde{d}(Q, X|_{I,t})$. Here it is important to optimize $\tilde{d}$ instead of $\hat{d}$ because the calculation of $\tilde{d}$ returns a valid permutation of the time series dimensions, while $\hat{d}$ may not. The selection process is shown in Alg. 3.

Here note that the convergence time of Alg. 3 depends on the order of the dimensions of the query pattern. However, this can be mitigated through implementation of optimizations that are discussed in Sec. 7.

5.3 Combined Algorithm

The combined algorithm shown in Alg. 4 is a hill-climbing algorithm that searches for a minimum matching distance of the sequences. As noted by the bi-clustering community, this type of algorithm can become stuck in local objective minimums. However, in the case of pattern matching, this is usually not the case when the pattern is discernible in the data. Moreover, recalling that we are looking for an exact pattern match, we expect the output distance of the algorithm to be very small. In contrast when a local minimum is

Algorithm 3 Algorithm for query match

Input: Query sequences Q, time series sequences X
Output: Integer set I
Initialization:
Do z-normalize by dimension on Q
Do z-normalize by dimension on X
$I \leftarrow \emptyset$
$J \leftarrow [M]$
LOOP Process:
for $i = 1$ to m **do**
  $minD \leftarrow \infty$
  for $j \in J$ **do**
    $d \leftarrow \text{DTW}(q_i, x_j)$
    if $(d < minD)$ **then**
      $r \leftarrow j$
      $minD \leftarrow d$
    end if
  end for
  Append I with r
  Remove r from J
end for
return I

found, the resulting output distance is large. Thus a threshold can be set to detect any outliers that can be found.

To understand Alg. 4, begin with the following intuition. Assume that there is a close pattern match in the time series dimensions denoted by the index set I^*. Then suppose a random subset I is chosen to perform the initial pattern search. If there is a partial overlap between I^* and I in the search, the location of the subsequence which is returned before the start of the second iteration should be reasonably close to the final location of the matching pattern. The reason for this is that the partial match of the pattern should return a relatively smaller distance than non-matching instances in the selected time series. If this is the case, then the dimension choosing algorithm, Alg. 3 will choose the correct I^* for the next pattern search iteration. When this happens the second pattern search executed using Alg. 2 will complete the search since the same pattern dimensions will be identified.

When I and I^* have no overlap, the search is looking for a non-existing pattern match in the time series set indexed by I, and will find some arbitrary location, and Alg. 3 will be equivalent to choosing the next set of indexes randomly, making it possible that the next set of indexes and I^* will overlap in the following iteration. In practice, one may use the following strategy to ensure an initial overlap between I and I^*. Run the algorithm sequentially with initialization $I_1 = \{1, \ldots, m\}$, $I_2 = \{m+1, \ldots, 2m\}$, etc, such that the initializations span all M indexes. Compare the resulting $\tilde{d}$ associated with each result, and return the set of indices whose output resulted in the smallest $\tilde{d}$. Iterate the time stamp search on the set of indices with the smallest $\tilde{d}$ and complete the algorithm.

6 SEARCHING TOP-k MATCHES

The proposed algorithm is capable of finding the best multidimensional matching subsequence within a multidimensional time series.

Algorithm 4 Aggregate algorithm

Input: Query sequences Q, time series sequences X
Output: Integer set I, time stamp t
Initialization:
Do z-normalize by row on Q
Construct upper and lower envelope by row on Q for calculating LB_{Keogh}
Sample m numbers without replacement from $[M]$ to construct I_{out}
$I \leftarrow \emptyset$
LOOP Process
while $I \neq I_{out}$ **do**
  $I = I_{out}$
  Run Algorithm 1 on $X|_I$ to get t^*
  Run Algorithm 2 on $X|_t$ to get I_{out}
end while
return t, I

Importantly, this algorithm is able to find this pattern when the data dimensions exceed the query dimensions. In essence, this approach offers the capability of dimension reduction in addition to the ability for pattern matching.

We now propose to extend the developed algorithm to find the top-K matches. This task is needed in predictive maintenance and anomaly detection applications because similar matches can provide context for the state of the system or the anomaly. For example, note the IMU data presented in Fig. 1(a) and the three dimensional pattern that occurs when bias spikes in Fig. 1(b). Recall that bias spikes a total of ten times in this time series. Finding and comparing all ten time instants provides an operator a richer set of information than the single best match to a sample bias spike.

However, finding the top-K matches is not trivial. The naive solution is to iterate the best match algorithm K times, removing the determined time sequence after each run. This solution is not satisfactory because it scales the computational complexity of pattern detection by a factor of K.

6.1 One dimensional case

To develop a more sophisticated approach to finding the top-K matches, we begin with the one dimensional case. In this case the best algorithm for pattern matching is the UCR suite [11], a natural building block for a top-K match searching algorithm. Because the goal is to find the top-K matches without finding repeating the algorithm K times, then we need to implement a new method of keeping an ordered list of the K best matches found so far. This list must be updated each time a new match is discovered. Here we choose to maintain this list via a priority queue. A priority queue is a data type where each element is assigned a priority with respect to the other elements in the queue. Priority queues can be depicted as a tree, where each parent node in the tree is larger than its children nodes. The top node is thus the largest, and in the case of pattern matching, the largest observed distance so far.

Embedding a priority queue into the algorithm proposed dramatically reduces the computational cost of finding K matches. There are two reasons for this computational improvement. First,

the cost of referencing the top element in a priority queue is O(1), thus by placing the K^{th} distance at the top of the priority queue, we are make the comparison cost to be the same as that of the original algorithm. Second, insertion into the priority queue has a computational cost of O($\log K$) which is minimal as long as $K \ll N$.

The proposed approach is a very computationally efficient method of finding the top-K distance matches in a time series. However, due to the warping capability of the DTW, the top-K distance matches are not the top-K qualitative matches in the time series. To find the top-K qualitative matches in the time series we add a pre-processing step to the priority queue. In this step we track the trend in the DTW distance as the sliding window changes positions. A decreasing trend indicates that a matching location is being approached and an increasing trend indicates that the matching location has been passed. If a matching location has been passed and the matching distance is smaller then the K^{th} best distance to date, then the new matching location is inserted into the queue. If not, the pre-processing is continued. This pre-processing step increases the quality of the matches produced by the proposed algorithm and reduces the computational complexity of the approach because it prevents the unnecessary re-ordering of the queue as the sliding window approaches a match location.

6.2 Extending the Top-K Matching Algorithm to Multiple Dimensions

Now we want to extend the algorithm to find K pairs of time stamps and index sets $(I_1, t_1), \ldots, (I_K, t_K)$ such that $d(Q, X|_{I_i, t_i}, i)\ \forall i \leq K$ are the K smallest DTW matching distances in the data set. To facilitate the algorithm two key assumptions must hold. First, the top-k matches are found in the same subset of time series in the data (that is, $I_i = I^*$, for all $i \leq k$). In essence, the dimensions cannot change for matches within the top-K. However, the match dimension permutation does not need to be the same for every match. Second, the query patterns (dimensions) as not identical and opposite to one another. That is, if the queries are added together, the resulting time series is not close to zero (DTW$(\sum_i q_i, \mathbf{0})$ is well above 0) or otherwise below the noise floor of the signal. This second assumption allows us to further speed up matching by reducing the need to solve a combinatorial problem in the match dimensions.

If the query pattern, and the desired K matches satisfy these assumptions, then we propose the following algorithm to find the top-K matches in a multidimensional time series. First, add the query dimensions, collapsing the multidimensional pattern into one dimension. Second, add the chosen data sequences together, collapsing the m-dimensional time series into a single dimension. Finally apply a modified one-dimensional top-K matching algorithm in the preceding section.

Here the algorithm implementation is augmented in two important ways; first multiple data sequences can only be added after they are normalized. This is because sequences of different magnitudes may distort the results and provide inaccurate shape results. Second, only two nested lower bounds are used in the algorithm. The first bound is the Kim bound, described in previous sections, and the second is the Keogh bound, again described in the prior sections. The reverse Keogh bound is not used. This is because the reverse Keogh bound requires a recalculation of an envelope over the candidate matching data sequence. This combined with the continuous data normalization, which is necessary when the time series are added, significantly reduces the search speed of the algorithm. Removing this lower bound did not significantly impact the accuracy or speed of the algorithm as will be shown in the results section.

7 NUMERICAL EXPERIMENTS

This section demonstrates the performance of both the best match algorithm and the best match algorithm extended to find the top-K matches. Each algorithm's performance is evaluated against the performance of the UCR suite, the best one-dimensional pattern matching algorithm. Here the one dimensional algorithm is run on each dimension and the solution is found via post-processing of the identified pattern matches. In the top-K setting, the algorithm is run K times on each time series, each time removing the located match before the new run. The top-K matches are then found via post processing.

Two data sets are used in the experiments, the first is the real data set collected from an IMU and depicted in Fig. 1. The second is a synthetic data set comprised of six randomly generated series. The synthetic data set is seeded with search patterns on 3 of the 6 dimensions. These search patterns are inserted in five random locations and with five different Signal-to-Noise Ratios (SNRs).

Using synthetic data allows us to test the algorithms more extensively for two reasons. First, given that the UCR suite is deterministic, running the algorithm once on our real data set does not provide clear evidence of the generalizability of the results. However, running the UCR suite on randomly generated data that is generated before each run provides a notion of the variance in the performance that might be observed over a large set of real data time series. Second, the SNR of the collected experimental data is fixed. Experimenting with synthetic data allows us to vary the pattern SNR and observe the effect of increasing noise on the top-K match search.

7.1 Data and Query Generation

The synthetic data set contains six times series each containing 500,000 (N) data points. The time series are generated using a Gaussian random number generator, seeded with a known seed for each experiment, where the Gaussian distribution is defined as $N(0, 1)$.

To detect a known pattern within this data, a three dimensional pattern is inserted. Each dimension of the pattern is added to one of the six time series at five random locations. The patterns are described as follows and shown in Fig. 2.

$$\text{Query Pattern 1} = (x-5)^2, \quad 0 \leq x \leq 10 \tag{6}$$

$$\text{Query Pattern 2} = \sin(0.75x), \quad 0 \leq x \leq 10 \tag{7}$$

$$\text{Query Pattern 3} = 1 - 2\mathbf{I}(x \leq 5), \quad 0 \leq x \leq 10 \tag{8}$$

In these equations $\mathbf{I}(\cdot)$ is the indicator function. Because only the query shape is of interest the shapes are defined over a convenient interval of $(0, 10)$, but sub-sampled into n data points during insertion. In the experiments below $n = 100$. In order to produce ranked matches and to test the robustness of the matching algorithm to noise, each query is inserted with a sequentially worse SNR. Each

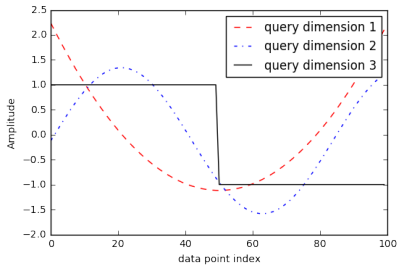


Figure 2: Dimensions of the Query Pattern Plotted Together.

SNR is achieved by first normalizing the query pattern and then scaling it to achieve the desired SNR. For example, scaling the normalized query pattern by $\sqrt{2}$ is equivalent to setting the SNR to 3dB. The inserted patterns in the synthetic data are scaled to SNRs 10dB, 8dB, 6dB, 5dB, and 3dB which correspond to scale factors $\sqrt{9}$, $\sqrt{6}$, $\sqrt{4}$, $\sqrt{3}$, and $\sqrt{2}$, respectively.

Figure 3: Top match query and 5th best query plotted in the time series

The lowest SNR was chosen empirically such that when searching for the top 5 matches, the 5th match would be found with 100% accuracy in any random sample of 100 experiments. No SNR was chosen that would make the patterns plainly visible to humans (as is the case with the IMU data). The corrupting effect of the Gaussian data points on the queries is shown in Fig. 3. In this figure the match with the best SNR (10dB) is plotted on the left, and the worst SNR (3dB) is plotted in all dimensions on the right. Note that when plotted on the same vertical scale, the degradation in SNR is obvious from 10dB to 3dB. In fact, it may even be surprising that the 3dB query pattern is identifiable by the top-K search algorithm.

7.2 Multidimensional Pattern Search on Synthetic Data

The results of the algorithm performance for all experiments on synthetic data are presented in Fig. 4[1]. Fig. 4a shows the experiment search time for the best multidimensional pattern match when the query dimension is the same as the data dimension, $M = m = 3$ and the SNR of the query pattern is 6dB. Here note that the median search time of the proposed algorithm was 3.55 seconds while the extended UCR suite has a median run time of 17.3 seconds.

[1] Experiments performed on a Windows 10 PC with 16GB RAM, an Intel i7-4770/3.40 GHz CPU, and a 1TB 7200 RPM HDD.

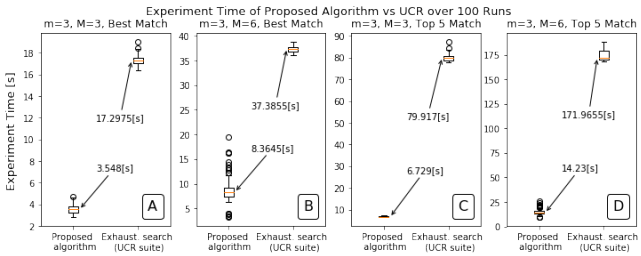


Figure 4: Proposed Algorithm Results Compared to the Exhaustive Search.

Increasing the data dimensions to $M = 6$ in Fig.4b shows the dimension reduction capabilities of the algorithm. In particular, when the $m = 3$ and $M = 6$, the median run time to obtain the pattern match was 8.36 seconds while the exhaustive search has median convergence time of 37.39 seconds. Notably when poorly initialized the proposed algorithm has a run time as long as 20 seconds. In both figures, the search pattern is always correctly found by the algorithm.

The greater spread of the experiment results sheds light on the issue of algorithm initialization discussed earlier in the paper. Given that there are six time series from which to choose and only three query patterns, then it is possible for the initial choice of time series to contain no matching patterns. In such a case, the algorithm becomes trapped in the afore mentioned local minimum. To address this a data dependent matching distance threshold was set, such that if the final matching result is larger than this threshold, then the result is deemed erroneous and the algorithm is re-initialized with a new random set of rows. The extended run time here reflects the occurrence of such a re-initialization.

Re-initialization is an acceptable solution when the relative size of m and M is comparable. In the case where $m \ll M$ an alternate method may be preferable. The alternate method is to divide $[M]$ into subsets $\{1, \ldots, m\}$, $\{m+1, \ldots, 2m\}$, $\ldots$ and run the search algorithm in parallel on each subset. The results can then be re-aggregated, and the subsets containing the smallest matching distances can be further explored.

7.3 Top-k Matches Numerical Experiments

The third experiment with results shown in Fig. 4c demonstrates the performance of the top-K matching algorithm. From the figure we note that on average the top-K matching algorithm has a search time of 6.72 seconds. Note here that the ordering of the identified matches was 100% correct. In comparison, the extended UCR suite search had a median search time of 79.92 seconds. Thus the proposed algorithm has an improved search time of approximately 12 times when compared to simply extending 1-dimensional methods.

7.4 Total Search Time Experiments

Putting the algorithms together, the combined algorithm is demonstrated in Fig. 4d. The results are again compared to an exhaustive search that extends the one dimensional method. The figure shows that the median run time of the combined top-K multidimensional pattern matching algorithm is 14.23 seconds, while the combined

median run time of the extended one dimensional is 171.97 seconds. This is a 12 times improvement in the median search times.

7.5 Multidimensional Pattern Search on Real IMU Data

To provide real world context for this work, we now test the algorithms on the IMU data shown in Fig. 1. We perform two experiments, one experiment where three anomalous patterns occur simultaneously and another with three concurrent subsequences of data of interest. The results of these experiments are summarized in Table 7.5.

The first experiment is to find the best match to the pattern seen in Fig. 1b, Query Subsequence 1. Here the subsequence have a length of 300 and has a dimension of m=3. The combined algorithm finds this match within a multidimensional time series of dimension M=6 in 11 seconds. The extended 1-dimensional method is finds the same match in 50 seconds.

To show that the patterns need not begin simultaneously, we find the top 5 matches for Query Subsequence 2 shown in Fig. 1b. Note here that this subsequence is much larger at 1000 data points in length. For this size subsequence, the proposed algorithm was able to: determine the appropriate matching dimensions, and find the top 5 patterns, in 338 seconds, while the extended 1-dimensional method was able to find the same top 5 matches in 553 seconds.

Experiment Type	Proposed Algorithm	Extended UCR Suite	Query Length
Best Match 3/6	11s	50s	300
Top 5 Matches	338s	553s	1000

These experiments show, that the algorithms proposed herein are capable of finding the anomalous bias measurement spikes in Query Subsequence 1 in Fig. 1b, as well as larger subsequences that describe the system condition such as Query Subsequence 2 in Fig. 1b. Furthermore, these experiments show that the computational advantage of the proposed algorithm is clear even as the query size grows to 1000 data points. Thus the proposed algorithm provides an effective method of: detecting concurrent anomalous patterns, detecting concurrent subsequences of interest in a multidimensional time series, and dimension reduction of multidimensional time series.

8 CONCLUSION

This paper addresses the problem of finding the top-K matches to a multidimensional query pattern in a multidimensional time series data set. The presented algorithms leverage a new proposed distance function, two computational approximations of the distance function, a novel algorithm framework that iteratively searches for the correct time series dimensions and pattern location, and an innovative algorithm extension that allows the identification of top-K matches with one data pass. The resulting combined algorithm is tested on two data sets, a real IMU data set, and a synthetic data set of Gaussian noise with inserted search patterns. Using the real data set we show the applicability of the approach for anomaly detection and condition monitoring. Using the synthetic data set we show the ability of the algorithm to detect multiple matching subsequences in the correct order in poor SNR. The results are compared to the trivial extension of 1-dimensional methods.

REFERENCES

[1] Ery Arias-Castro and Yuchao Liu. 2017. Distribution-free detection of a submatrix. *Journal of Multivariate Analysis* 156 (2017), 29–38.

[2] Cristina Butucea, Yuri I Ingster, et al. 2013. Detection of a sparse submatrix of a high-dimensional noisy matrix. *Bernoulli* 19, 5B (2013), 2652–2688.

[3] Yizong Cheng and George M Church. 2000. Biclustering of expression data.. In *Ismb*, Vol. 8. 93–103.

[4] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment* 1, 2 (2008), 1542–1552.

[5] D. Gamarnik and Q. Li. [n. d.]. Finding a Large Submatrix of a Gaussian Random Matrix. *ArXiv e-prints* ([n. d.]). arXiv:math.PR/1602.08529

[6] Tomasz Górecki and Maciej Łuczak. 2015. Multivariate time series classification with parametric derivative dynamic time warping. *Expert Systems with Applications* 42, 5 (2015), 2305–2312.

[7] Eamonn Keogh, Li Wei, Xiaopeng Xi, Sang-Hee Lee, and Michail Vlachos. 2006. LB_Keogh supports exact indexing of shapes under rotation invariance with arbitrary representations and distance measures. In *Proceedings of the 32nd international conference on Very large data bases*. VLDB Endowment, 882–893.

[8] Sang-Wook Kim, Sanghyun Park, and Wesley W Chu. 2001. An index-based approach for similarity search supporting time warping in large sequence databases. In *Data Engineering, 2001. Proceedings. 17th International Conference on*. IEEE, 607–614.

[9] Emil Laftchiev. 2015. *Robust dynamical model-based data representations and structuring of time series data for in-sequence localization.* PhD dissertation. The Pennsylvania State University.

[10] Beatriz Pontes, Raúl Giráldez, and Jesús S Aguilar-Ruiz. 2015. Biclustering on expression data: A review. *Journal of biomedical informatics* 57 (2015), 163–180.

[11] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 262–270.

[12] Chotirat Ann Ratanamahatana and Eamonn Keogh. 2004. Making time-series classification more accurate using learned constraints. In *Proceedings of the 2004 SIAM International Conference on Data Mining*. SIAM, 11–22.

[13] Yasushi Sakurai, Masatoshi Yoshikawa, and Christos Faloutsos. 2005. FTW: fast similarity search under the time warping distance. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. ACM, 326–337.

[14] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[15] Andrey A Shabalin, Victor J Weigman, Charles M Perou, and Andrew B Nobel. 2009. Finding large average submatrices in high dimensional data. *The Annals of Applied Statistics* (2009), 985–1012.

[16] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. 2015. On the non-trivial generalization of dynamic time warping to the multi-dimensional case. In *Proceedings of the 2015 SIAM International Conference on Data Mining*. SIAM, 289–297.

[17] Gineke A Ten Holt, Marcel JT Reinders, and EA Hendriks. 2007. Multi-dimensional dynamic time warping for gesture recognition. In *Thirteenth annual conference of the Advanced School for Computing and Imaging*, Vol. 300.