Interactive Time Series Exploration Powered by the Marriage of Similarity Distances

Rodica Neamtu
Worcester Polytechnic Institute
Worcester, MA, USA
rneamtu@wpi.edu

Ramoza Ahsan
Worcester Polytechnic Institute
Worcester, MA, USA
rahsan@wpi.edu

Elke Rundensteiner
Worcester Polytechnic Institute
Worcester, MA, USA
rundenst@cs.wpi.edu

Gabor Sarkozy
Worcester Polytechnic Institute
Worcester, MA, USA
gsarkozy@cs.wpi.edu

ABSTRACT

Finding similar trends among time series data is critical for applications ranging from financial planning to policy making. The detection of these multifaceted relationships, especially time warped matching of time series of different lengths and alignments is prohibitively expensive to compute. To achieve real time responsiveness on large time series datasets, we propose a novel paradigm called Online Exploration of Time Series (ONEX) employing a powerful one-time preprocessing step that encodes critical similarity relationships to support subsequent rapid data exploration. Since the encoding of a huge number of pairwise similarity relationships for all variable lengths time series segments is not feasible, our work rests on the important insight that clustering with inexpensive point-to-point distances such as the Euclidean Distance can support subsequent time warped matching. Our ONEX framework overcomes the prohibitive computational costs associated with a more robust elastic distance namely the DTW by applying it over the surprisingly compact knowledge base instead of the raw data. Our comparative study reveals that ONEX is up to 19% more accurate and several times faster than the state-of-the-art. Beyond being a highly accurate and fast domain independent solution, ONEX offers a truly interactive exploration experience supporting novel time series operations.

1. INTRODUCTION

1.1 Motivation

[1]In applications ranging from finance, business, medicine to meteorology [14],[24], time series data is prevalent, presented as stock fluctuations, ECG, rainfall amounts, etc.

[1]First two authors equally contributed to the work

Proceedings of the VLDB Endowment, Vol. 10, No. 3

Motivating example. Let's consider a real life example revealing challenges involved in finding and leveraging data similarities for decision making. In 2013 in Massachusetts, organizations set out to repeal the Sales and Use Tax on computer and software services, perceived as potentially having a negative impact on the economic health of the state. Data-driven evidence was analyzed to show similarities between tax rates over time and fluctuations of social and economic factors, all modeled as *time series*, obtained from a large spectrum of public governmental websites. Many difficulties were encountered, chief among them being the time to find and interpret the similarities between economic indicators represented as time series.

(1) The presence of data from different domains reported over specific intervals requires the comparison of time series of different lengths and alignments, as the impact of a tax change might take different durations to become apparent. Such time-aware comparisons must be performed using robust distances like Dynamic Time Warping (DTW) [5]. The "power" of these measures, beneficial in terms of accuracy, is shadowed by their computational complexity and thus slow time responsiveness even for medium size datasets.

(2) During this process analysts used specific indicators, like the growth rate, to evaluate the potential impact of introducing a new tax. For example, they "designed" a sample growth rate time line indicative of a positive impact of the tax and searched for matches among all states. In such scenario, it is possible that the sequence may or may not exist in the dataset. Perfect matches, if found, reflect similar impacts in those specific states while close matches indicate slightly different impacts in specific states. Thus analysts need to have the ability to explore large time series data by using sample sequences that might or might not be present in these datasets.

(3) Analysts had to answer complex questions that were not necessarily based on finding the best match sequence. For example, they searched for recurring similarity patterns such as the growth rate of a state over a few years as well as similar growth rates and other economic indicators in different states over specific time lengths, indicating similar "short-term" impacts.

(4) Data from diverse domains requires the use of different parameter settings such as similarity thresholds, leading to repeated and redundant computations for each parame-

ter setting. For example, the most suitable thresholds for studying similarity of demographic data are different than the ones used for growth rates.

1.2 Research Challenges and Limitations of the State-of-the-Art

1. High data cardinality leading to decreased responsiveness. Time series datasets like financial records or collections of ECG data tend to be huge. Worse yet, the need for accommodating different temporal granularities requires us to consider time series of different lengths. For a dataset containing N time series, each of length n, the total number of subsequences to consider is $Nn(n-1)/2$. For example, a benchmark dataset like StarLightCurves from the UCR Time Series Data Mining Archive[2] with 9236 time series, each of length 1024, is composed of $4.83e^9$ subsequences. Real-world datasets tend to be orders of magnitude larger than this[3]. Performing similarity comparisons for all these subsequences is clearly impractical.

Many state-of-the-art techniques try to address this ***lack of instantaneous responsiveness.*** Lag in responsiveness risks losing an analyst's attention during the exploration process. The state-of-the-art is faced with the trade-off between accuracy and time response, especially in exploring very large datasets. Some systems provide an exact or a highly accurate solution [1], [19], [26] at the expense of a timely response. This could be detrimental for applications in medicine and finances that depend on immediate answers to act on. Others [2] use a preprocessing step to improve the timely responsiveness. However, their requirement for setting many different parameters limits their efficiency [22].

2. Supporting comparisons of sequences of different lengths and alignments. To compare sequences of different lengths and alignments, robust distances such as DTW [5] must be used. To precompute, store and explore the huge number of pairwise similarity comparisons between sequences using DTW is not feasible even for datasets that are not very large. Many systems resort to either accepting increased time responses or settling for using faster-to-compute distances, at the expense of decreased accuracy.

The state-of-the-art has been wrestling with the ***compromise between the choice of similarity distance and time responsiveness.*** Many applications rely on the use of fast-to-compute distances like the Euclidean Distance [12], [16], [31] to achieve a fast response time. As a consequence they cannot handle sequences that are not aligned or have different lengths. Yet the use of time-warping distances like DTW is overshadowed by their computational complexity [7], leading to slow responsiveness and poor scaling as data grows. Applications thus have to make a difficult choice between the ability to perform powerful comparisons and the resulting time delays and storage demands.

3. Offering rich classes of exploratory queries. As shown in our motivating example, getting insights into data involves not just finding the best match for a sequence, but also discovering similarity patterns. Thus, such requirement compounds the previous challenges.

For this reason, most state-of-the-art systems suffer from **lack of flexibility in exploring similarity** and focus only on finding the best match for a sequence [22], [12], [18]. Analysts would benefit from having the opportunity to answer other complex questions related to similarity, including finding patterns using the same system. This would help them to better understand the datasets.

4. Supporting guided similarity exploration and the use of varying similarity thresholds. Analysts in different domains interpret similarity differently and need to use different parameter settings like similarity thresholds. Keeping the result set for each possible similarity parameter is inefficient, resulting in large storage and repeated computation. Many state-of-the-art systems can not overcome the ***difficulty of supporting varying similarity parameters.*** Understanding the way similarity between time lines changes when using different parameter values is critical for analysts. Unfortunately, most existing systems [19], [22], [26] do not support flexible similarity insights.

Most state-of-the-art systems suffer from the ***lack of parameter recommendations.*** Analysts may not know the precise parameter settings that would result in the most interesting similarity results. It might take many successive trial-and-error interactions using different parameter values to get insights from the data. Existing exploratory systems [2], [12], [17] tend to be "black boxes" that provide no suggestions for parameter settings. A system that makes recommendations for parameter tuning would help analysts in finding similarities using fewer trial-and-error attempts. include the well-known Euclidean distance (ED) [12], Dynamic Time Warping (DTW) [5],[18] and distances based on Longest Common Subsequence (LCSS) [29]. They all involve trade-offs between the choice of distance which is connected to the ability to perform time warping and the time response.

In summary, it is very difficult to build interactive systems for the efficient exploration of time series. Such systems might compromise between the choice of similarity distance and time responsiveness. Even the most promising solutions focus on a specific class of queries, still leaving room for improvement in the area of answering richer classes of queries.

1.3 Our ONEX Approach

ONEX offers a viable answer to the trade-off between the need for real time responsiveness and the high computational complexity due to the use of non-metric distances for comparing sequences with different lengths and alignments. Because it is prohibitively expensive to use DTW, we solve this problem by employing two distances: while we use the computationally inexpensive Euclidean Distance to construct compact similarity groups for specific lengths, we are able to support the exploration of these groups using a robust time-warping method: DTW [5]. This unique combination yields very accurate results at much reduced response time rates, as DTW is successfully applied over the compact ONEX base instead of the raw data.

Contributions:

1. We formally prove a triangle inequality between ED and DTW that builds a conceptual bridge between the offline construction of the ONEX base and its online exploration. Our solution offers a viable answer to the trade-off dilemma between the use of complex time warped distances and timely responsiveness. (Sec. 3.2).

2. We precompute critical similarity relationships between

[2] *www.cs.ucr.edu/ eamonn/time_series_data/*

[3] https://followthedata.wordpress.com/2014/06/24/data-size-estimates/

subsequences and preserve them in the form of "similarity groups" and their representatives (Sec. 3.1). The resulting ONEX base plays a key role in achieving interactive responsiveness.

3. Our query strategies enable analysts to get insights into datasets by exploring similarity and its recurring patterns. ONEX also provides analysts with parameter tuning guidance to give a better intuition of the major changes in similarity. (Sec. 5.1)

4. Our experimental performance shows that ONEX is several times faster than the fastest known method [22] and many orders of magnitude faster than [19] and [26]. In addition, ONEX is up to 19% more accurate than [22] for traditional similarity queries (Sec. 6).

2. KEY CONCEPTS USED IN ONEX

A time series $X = (x_1, x_2, ..., x_n)$ is a sequence of n real values. A dataset $D = \{X_1, X_2, ..., X_N\}$ is a collection of N such time series.

DEFINITION 1. *The **subsequence of a time series** X_p, denoted $(X_p)_j^i$, is a time series of length i starting at position j where $1 \leq i \leq n$ and $0 \leq j \leq n-1$.*

We discuss in this paper the Euclidean Distance (ED) and Dynamic Time Warping (DTW)[5].

DEFINITION 2. *Given two time series of equal length $X = (x_1, ..., x_n)$ and $Y = (y_1, ...y_n)$, their **Euclidean distance** (ED) is defined as:*

$$ED(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (1)$$

Unfortunately, two time series can be similar while having a large ED when they are not aligned. Thus the distance measure Dynamic Time Warping (DTW) [5] that solves this problem has been developed. DTW allows for one-to-many alignment between points from the two to-be-compared time series to support both time shifting as well as sequences of distinct lengths. ED, which corresponds to a one-to-one mapping of the pairs of points composing the time series, can be seen as a special case of DTW.

Dynamic Time Warping. Suppose we have two time series $X = (x_1, x_2, ..., x_n)$ and $Y = (y_1, y_2, ..., y_m)$. To align these sequences using DTW, an $n \times m$ matrix $M(X,Y)$ is constructed, where the $(i,j)^{th}$ element of the matrix is the Euclidean Distance between x_i and y_j, i.e., $w_{i,j} = ED(x_i, y_j)$. Then a *warping* path P is a set of elements that forms a path in the matrix from $(1,1)$ to (n,m). The t^{th} element of P denoted as $p_t = (i_t, j_t)$ refers to the indices i_t, j_t of (x_{i_t}, y_{j_t}) of this matrix element in the path. Thus a path P is $P = (p_1, p_2, \ldots, p_t, \ldots, p_T)$, where $n \leq T \leq 2n-1$, $p_1 = (1,1)$ and $p_T = (n,m)$.

DEFINITION 3. *Given two time series $X = (x_1, ..., x_n)$ and $Y = (y_1, ..., y_m)$, the **weight of the warping path** P is defined as:*

$$w(P) = \sqrt{\sum_{t=1}^{T} w_{i_t,j_t}^2} \quad (2)$$

The **DTW distance** *between X and Y, or DTW(X,Y), is defined to be the weight of the path P with the minimum weight among all possible warping paths. (*$\min_P(w(P))$*.*

This path can be calculated using dynamic programming [25]. More details can be found in [20, 22].

DEFINITION 4. *Two time series X and Y are **similar** if the chosen distance such as ED or DTW between them is within the given similarity threshold ST, or $Dist(X,Y) \leq ST$ where $Dist \in \{ED, DTW\}$.*

The following normalized distances are crucial for establishing our theoretical foundation of time-warped retrieval.

DEFINITION 5. *Given two time series X and Y, we define their **Normalized Euclidean distance** as:*

$$\overline{ED}(X,Y) = ED(X,Y)/\sqrt{n} = \sqrt{1/n\sum_{i=1}^{n}(x_i - y_i)^2} \quad (3)$$

with $ED(X,Y)$ as defined in Def. 2.

DEFINITION 6. *Given two time series X and Y, with $m \leq n$, their **Normalized DTW distance** $\overline{DTW}$ is defined as*

$$\overline{DTW}(X,Y) = DTW(X,Y)/2n \quad (4)$$

with $DTW(X,Y)$ as defined in Def. 3.

Without loss of generality, we use $m \leq n$, but m and n are interchangeable.

3. THEORETICAL FOUNDATION OF ONEX FRAMEWORK

Here we establish a formal foundation for our ONEX framework. As core, we prove a triangle inequality between ED and DTW, which is conceptually similar to the well-known triangle inequality for ED. This allows us to build a compact space of representative sequences, called the ONEX base, based on the ED distance, and then to process DTW-based queries against this compact space instead of raw time series - while still guaranteeing the quality of the retrieval process.

3.1 ONEX Similarity Groups

Our ONEX base compactly encodes similarity relationships between all subsequences of the time series in D. As foundation, we group subsequences of the same length that are similar according to Def. 4 using the ubiquitous and inexpensive ED distance into so called "ONEX similarity groups" and then summarize these groups by a carefully chosen representative, as further described below.

DEFINITION 7. *Given a group S of time series of equal length i, then the **representative** R_k^i of S is defined as the point-wise average of the sequences in the set S [10]. That is, $R_k^i = avg((X_p)_j^i)$, for all $(X_p)_j^i \in S$.*

We now introduce the notion of ONEX similarity groups that encode similarity relationships between subsequences by imposing several key requirements that, as we prove in Sec. 3.2, assure that these groups instead of the raw data can be safely explored through their representatives.

DEFINITION 8. *Given the set T of all possible subsequences $(X_p)_j^i$ of the time series of dataset D, assume these subsequences $(X_p)_j^i \in T$ are grouped into similarity groups with their respective representatives R_k^i, such that all subsequences $(X_p)_j^i \in T$ are in one and only one group G_k^i. These similarity groups are defined to be **ONEX similarity groups**,*

denoted by G_k^i, if the following three properties hold:
(1) all subsequences $(X_p)_j^i$ in a group G_k^i must have the same length i,
(2) $\overline{ED}$ between any $(X_p)_j^i$ in G_k^i and the representative R_k^i of this group G_k^i is smaller than half of the similarity threshold ST used by the system, that is $\overline{ED}((X_p)_j^i, R_k^i)) \leq ST/2, \forall i, j \in [1, n] \forall p \in [1, N]$.
(3) $\overline{ED}$ between the subsequence $(X_p)_j^i$ and the representative R_k^i of the group G_k^i is the smallest compared to ED of $(X_p)_j^i$ and all other representatives R_l^i of the same length i defined over D, or $\overline{ED}((X_p)_j^i, R_k^i)) \leq \overline{ED}((X_p)_j^i, R_l^i))(\forall i, j \in [1, n])(\forall p \in [1, N])(\forall l \in [1, g])$, where g denotes the number of representatives of length i.

The key requirements for placing sequences into the same ONEX similarity group are two-fold. First, $\overline{ED}$ of the sequences to the representative of the group must be the smallest compared to the $\overline{ED}$ to any other representative, and second, it is also smaller than ST/2.

LEMMA 1. *For any two subsequences X and Y belonging to the same group G_k^i, with G_k^i defined in Def. 8, the $\overline{ED}(X,Y)$ defined in Def. 5 is within the threshold ST, that is, $\overline{ED}(X, Y) \leq ST$, for all X, $Y \in G_k^i$.*

Proof. Let $X = (x_i...x_j)$ and $Y = (y_i...y_j)$ be two time series in the same similarity group G_k^i and $R = (r_i...r_j)$ be the representative of that group. According to Def. 8, we have: $\overline{ED}(X, R) \leq ST/2$ and $\overline{ED}(Y, R) \leq ST/2$
Using Def. 5, this means:

$$\sqrt{\sum_{k=i}^{j}(x_k - r_k)^2} \leq ST/2 \tag{5}$$

$$\sqrt{\sum_{k=i}^{j}(y_k - r_k)^2} \leq ST/2, \tag{6}$$

We want to prove that: $\overline{ED}(X, Y) \leq ST$ which means:

$$\sqrt{\sum_{k=i}^{j}(x_k - y_k)^2} \leq ST, \tag{7}$$

Squaring Equations (5) and (6), we get:

$$ED^2(X, R) = \sum_{k=i}^{j}(x_k - r_k)^2 \leq \frac{ST^2}{4}. \tag{8}$$

$$ED^2(Y, R) = \sum_{k=i}^{j}(y_k - r_k)^2 \leq \frac{ST^2}{4}. \tag{9}$$

We have:

$$x_k - y_k = x_k - r_k + r_k - y_k = (x_k - r_k) + (r_k - y_k).$$

Using the Cauchy-Schwarz inequality [27], we get:

$$(x_k - y_k)^2 \leq 2(x_k - r_k)^2 + 2(r_k - y_k)^2.$$

Then using this and Def. 5 and (5), (6), we get:

$$\begin{aligned}\sum_{k=i}^{j}(x_k - y_k)^2 &\leq 2\sum_{k=i}^{j}(x_k - r_k)^2 + 2\sum_{k=i}^{j}(y_k - r_k)^2 \\ &\leq 2(ST^2)/4 + 2(ST^2)/4 = ST^2.\end{aligned} \tag{10}$$

Extracting the square root we proved that (7) is true. □

We "represent" each group constructed over a dataset D by only *one single* sequence, namely, the group's representative. We collect the representatives for all groups over G into a collection, called the **Representative Space** ($\mathcal{R}$-Space).

DEFINITION 9. *Given a a data set D and a collection of mutually exclusive groups $\{G_k^i\}$ covering all sequences of all lengths $i \in \mathcal{L}$, then the set of representatives, with $\{R_k^i\}$ the representative of $\{G_k^i\}$, along with their associated subsequences $(X_p)_j^i$ in $\{G_k^i\}$, is called the **Representative space**.*

Lastly, we compute the pairwise Inter-Representative Distances between all pairs of representatives.

DEFINITION 10. *Given the collection of representatives in D, i.e., the $\mathcal{R}$-Space, then the **Inter-Representative Distance** Dc between two representatives R_k^i and R_l^i of the same length i of $\mathcal{R}$-Space is defined by $Dc_{kl}^i = \overline{ED}(R_k^i, R_l^i)$.*

This distance plays a central role in dealing with variable similarity thresholds, as shown in Sec. 4.2.

3.2 Time-warped Retrieval based on ED-DTW Triangle Inequality

The cornerstone of our ONEX time-warped retrieval framework is this unique conceptual solution based on proving a triangle inequality between ED and DTW. We prove that the similarity between a sample sequence *seq* provided by the user and the representative of an ONEX similarity group as defined in Def. 8 "extends" to all the subsequences in that group. This empowers ONEX to *perform time warped comparisons of the sample sequence over the compacted $R-$ Space instead of the entire dataset.*

More specifically, if $\overline{DTW}$ between the *seq* and the representative R_k^i is smaller than $ST/2$, then we can guarantee that all sequences in that group G_k^i are similar to this sequence *seq* and that $\overline{DTW}$ between *seq* and any of these sequences is within the similarity threshold ST.

LEMMA 2. *Given $Y' = (y'_1, \ldots, y'_n)$ an arbitrary sequence of length n in any group as per Def. 8), with the representative of the group $Y = (y_1, \ldots, y_n)$ and a sample sequence $X = (x_1, \ldots, x_m)$, then the following is true: If $\overline{ED}(Y, Y') \leq ST/2$ and $\overline{DTW}(X, Y) \leq ST/2$ then we have $\overline{DTW}(X, Y') \leq ST$.*

Proof. (Case: subsequences of same length:). We know from Def. 1, Lemma 1 and assumptions of Lemma 2:

$$ED(Y, Y') = \sqrt{\sum_{i=1}^{n}(y_i - y'_i)^2} \leq \sqrt{n}\frac{ST}{2}.$$

Squaring this we get:

$$ED^2(Y, Y') = \sum_{i=1}^{n}(y_i - y'_i)^2 \leq n\frac{ST^2}{4}. \tag{11}$$

We define matrices $M(X, Y)$ and $M(X, Y')$ as in Section 3. Given the assumptions related to this case we know that there is a warping path P in $M(X, Y)$ from $(1, 1)$ to (n, n) with the DTW weight at most $2n\frac{ST}{2} = nST$. We now have to show that there is a warping path from $(1, 1)$ to (n, n) in $M(X, Y')$ with weight at most $2nST$. In fact we will show that the same warping path P from $M(X, Y)$ will be good. Let P be $P = (p_1, p_2, \ldots, p_t, \ldots, p_T)$, where $n \leq T \leq$

$2n-1$, $p_1=(1,1)$, $p_T=(n,n)$, $p_t=(i_t,j_t)$. Then from the assumptions of *Lemma* 2, we have $\sqrt{\sum_{t=1}^{T}(x_{i_t}-y_{i_t})^2} \leq 2n\frac{ST}{2} = nST$. Squaring this, we get:

$$\sum_{t=1}^{T}(x_{i_t}-y_{i_t})^2 \leq n^2ST^2. \tag{12}$$

We want to prove that

$$\sum_{t=1}^{T}(x_{i_t}-y'_{i_t})^2 \leq 4n^2ST^2. \tag{13}$$

We have:

$$x_i - y'_i = x_i - y_i + y_i - y'_i = (x_i - y_i) + (y_i - y'_i)$$

Using the Cauchy-Schwarz inequality [27], we get:

$$(x_i - y'_i)^2 \leq 2(x_i - y_i)^2 + 2(y_i - y'_i)^2.$$

Then using this and (12), we get:

$$\begin{aligned}\sum_{t=1}^{T}(x_{i_t}-y'_{i_t})^2 &\leq 2\sum_{t=1}^{T}(x_{i_t}-y_{i_t})^2 + 2\sum_{t=1}^{T}(y_{i_t}-y'_{i_t})^2 \\ &\leq 2n^2ST^2 + 2\sum_{t=1}^{T}(y_{i_t}-y'_{i_t})^2.\end{aligned} \tag{14}$$

We estimate the second terms as $\sum_{i=1}^{n}(y_i-y'_i)^2$ with some of the terms repeated. The total number of repetitions is at most n, since the length of the warping path is at most 2n. Each fixed term is repeated at most $n-1$ times. Thus from Equations (11) and (14), we have:

$$\begin{aligned}&\sum_{t=1}^{T}(x_{i_t}-y'_{i_t})^2 \leq \\ &2n^2ST^2 + 2n\sum_{i=1}^{n}(y_i-y'_i)^2 \leq 2n^2ST^2 + \frac{2nnST^2}{4}.\end{aligned} \tag{15}$$

Thus Equation (13) will be true, if we have:

$$\frac{nnST^2}{2} \leq 2n^2ST^2.$$

This in turn is true if $n^2 \leq 4n^2$, which is always true. □

Proof sketch (Case: subsequences of different length) For Y and Y' subsequences of length n where Y is the representative of the group, Y‘ an arbitrary sequence in the group and X a query sequence of length m, with $m \leq n$, without loss of generality we consider here the case of $m \leq n$ but the proof is very similar for $n \leq m$. In the $\overline{DTW}$ defined in Def. 6 we divide by $2n$ because the warping path may have length up to $m+n \leq 2n$. Then the matrix $M(X,Y)$ is an $m \times n$ matrix and the warping path connects $(1,1)$ to (m,n). *Other than this, the proof for sequences of different lengths and the proof for sequences of the same length are the same.*

4. THE ONEX BASE

4.1 Strategies for ONEX Base Construction

The algorithm for building the ONEX Base, namely, for finding the groups for specific length subsequences and then computing their representatives is illustrated in Algo. 1. As first step, we decompose the existing time series in the dataset D into subsequences of all possible lengths. We then randomize the order of subsequences of each length using the well-known RANDOMIZE-IN-PLACE method [8] to remove data-related bias (Lines 1-5). In lines 7-10, we construct the first group by randomly selecting a subsequence and designating it as the representative of this first group. Choosing the first representative randomly ensures that the groups are not biased by the order in which subsequences are supplied [4]. In lines 12-20, a new randomly chosen subsequence of the same length is compared with previous representatives. Among all representatives for which their $\overline{ED}$ is smaller than $ST/2$, the one group with the minimum distance ED is chosen. Then the subsequence is placed into this group. Otherwise it will be placed in a new group and designated as the representative of the new group. We repeat this until all subsequences of each specific length are placed into a similarity group. Lastly, the representative for each group is determined. The final result is a panorama of all groups G_k^i and their representatives R_k^i for all possible lengths. The **complexity of the R-Space construction**

Algorithm 1: Construction of Similarity Groups and Representatives for Subsequences of Equal Length

```
   Input: Similaritythreshold ST, TimeSeries {X}, Length L
   Output: Representatives {R}, Groups {G}
1  begin
2      {G} = ∅, {R} = ∅
3      Randomized_IN_Place(X);
4      {X}= all subsequences of length L
5      minSM=0, mink=0
6      for X_p ∈ X do
7          if (G = ∅) then
8              G ← G_1
9              G_k ← X_p
10             R ← X_p
11         else
12             for k = 1 to Representative.count do
13                 minSM=closest Representative distance
14                 mink=Representative index
15             if (minSM ≤ (sqrt(L) * ST/2)) then
16                 G_mink ← X_p
17                 update R_mink
18             else
19                 G ← G_{K+1} //new Group
20                 R_{k+1} ← X_p
```

is $O(nl^2g)$ where l is the number of distinct lengths that each time series is decomposed into, g the total number of groups and n the number of time series in the dataset. Typically, n is much larger than l. Certainly l does not tend to go towards infinity since our initial time series are of a fixed length, and we furthermore divide them into smaller numbers. This is in contrast to n which tends to grow with the size of the dataset. Thus, we can treat l as a constant in regards to n. The complexity now can be summarized as $O(ng)$. Regarding g, let's examine the following probabilistic argument. Suppose we have k groups at some point and a new time series X comes in. There are (k+1) events that can happen, X either belongs to the one of the k groups or X starts a new group. Let‘s make the assumption that these events are all equally likely. Then the probability that X starts a new group is $1/(k+1)$. How many expected trials do we need until we get a new group? This is a geometric

distribution with expected value $1/p$ so $1/(1/(k+1)) = k+1$ [23]. Thus $\sum_{k=1}^{g}(k+1) = n$; the expected number of groups is $O(\sqrt{n})$. Then the complexity becomes $O(n^{3/2})$, which is much better than $O(n^3)$. Additional discussions related to the ONEX base can be found in our Tech Report.

4.2 Handling Multiple Similarity Parameters

As described above, our R-Space is constructed using a specific user-supplied ST. However, analysts might want to use different thresholds ST' to customize their notion of similarity based on the application. We thus extend the ONEX base to incorporate different values for ST, as desired by analysts. More precisely, during the construction of groups we identify similarity threshold values corresponding to specific lengths for which the groups of that length change significantly and incorporate them into a *Similarity Parameter Space* (*SP-Space*).

DEFINITION 11. ***The $\mathcal{SP}$-Space*** *is a conceptual space with length $\mathcal{L}$ and similarity threshold ST as its dimensions.*

We build an *SP-Space* for each specific length based on establishing relevant values for similarity thresholds. These could be ST_{half} and ST_{final} indicating when a half and respectively all precomputed groups of subsequences of a specific length merge – resulting in major differences in the result set. In general, two groups merge for a new similarity threshold ST', if $ST' \geq ST + Dc$ where Dc is defined in Def. 10. In Fig. 1 we showcase three specific lengths and the thresholds for which half and respectively all the groups of that length merge. For example, for length i we have $ST_{half} = 0.5$ and $ST_{final} = 0.78$. This means that for this specific length half of the groups merge if the analyst selects a new threshold $ST' = 0.5$ and all groups merge if $ST' = 0.78$ or higher. These "local" critical thresholds specific to each length can be combined to find the global similarity threshold values for which half and respectively all the groups across all lengths merge. We compute these "global" critical thresholds ST_{half} and ST_{final} by selecting the maximum of the local values of each length as depicted with dashed lines in Fig. 1. Choosing the maximum of all "local" ST_{final} values to be the "global" ST_{final} value ensures that all the groups of that length merge for a threshold $ST' >= ST_{final}$. We use the local and global threshold values to offer guidance in exploring similarity as shown later in Sec. 5.1. We now introduce the concept of **similarity degree** for our system:
S=Strict, where $ST \leq ST_{half}$
M=Medium, where ST in $[ST_{half}, ST_{final}]$
L=Loose, where $ST \geq ST_{final}$.

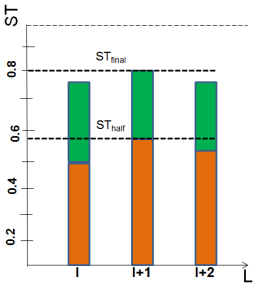


Figure 1: **Similarity Parameter Space**

For the example illustrated in Fig. 1, if an analyst asks for a recommendation for similarity thresholds ranges for ''Strict" similarity, the recommended values are in the range $[0, 0.6]$. Any value chosen by the analyst in this interval returns results with "strict similarity". Lower values chosen in this interval lead to "stricter" similarity translated in smaller distances between the similar sequences.

4.3 Storage and Indexing of ONEX Base

ONEX uses in-memory structures that achieve indexing in linear or constant time. We first construct a **Global Time Index** (GTI) as an array indexed by length for quick retrieval of the set of groups for each specific length. Each entry in GTI is composed of:

- a vector $V_i(k)$ containing the identifiers k for the groups G_k^i of a specific length i. They are used to retrieve the groups associated with that length.
- a two-dimensional array $D_i(k, j)$ containing the pairwise Inter-Representative distances (Dc) between groups G_k^i and G_j^i of length i. They help refine the results for different user-provided similarity thresholds. (Sec. 5.2)
- an array $S_i(k, sum_k)$ containing the representatives' identifiers and their associated sums of the pairwise Inter-Representative distances Dc for length i. We sort this array based on the sum of the distances to optimize the online search of representatives based on a given sample sequence.(Sec. 5.3).
- similarity threshold values ST_{half} and ST_{final} for length each i (Sec. 4.2) for local parameter recommendations.

For each group G_k^i, we maintain a **Local Sequence Index** (LSI) with the following elements:

- a two-dimensional array $ED_k(m, ED_m)$ containing the subsequence identifiers m in the group and their $\overline{ED}$ to the representative of the group. Each subsequence m in the group is a subsequence $(X_p)_j^i$ further identified by the time series identifier p and its starting position j. We sort this array based on the $\overline{ED}$ to the representative to optimize the online retrieval of sequences inside a "selected" group.
- vector containing the representative R_k^i of the group G_k^i.
- array containing the envelopes around each representative R_k^i using $LB_{(Keogh)}$ [22], which is a well-known lower bound for pruning unpromising candidates.

5. ONEX ONLINE QUERY PROCESSOR

5.1 ONEX Queries Classes

User-driven analytics in ONEX allow analysts to control the exploration by providing a target sample sequence *seq*. The **data-driven** analytics provide insights into the dataset without the user giving a sample sequence as a target match. **Class I: Similarity queries** fall into the user-driven query class. They return time series most similar to a user-supplied sample sequence *seq*. If "Match=Any" then sequences of every possible length are searched, otherwise only sequences of length l indicated by the sample are considered.

```
Q1 OUTPUT X_k
   FROM  D
   WHERE Sim <= min| ST, seq = q
   MATCH = Exact(L)|Any
```

Use Case: A financial analyst may want to retrieve the stock similar to the stock fluctuations of the Apple Stock for a specific time period. If the length is not specified, a similar stock of any length is retrieved. This illustrates the case when the sample sequence is a sequence present in the dataset. Alternatively, an analyst can "design" the desired stock fluctuation of interest and search for the best match for that sample sequence in the dataset. Such sequence is likely not to exist in the dataset, but rather the closest match sequence will be retrieved. Alternatively, referring back to our motivating example, similar economic indicators, like economic growth, can be found over specific time intervals or any time intervals.

Class II: Seasonal similarity queries allow the analyst to gain insights into the dataset by identifying similarity patterns. In the user-driven class, for example, queries like $Q2$, given a sample time series, return the "recurring" similarity by retrieving all similar subsequences of a specific length that belong to this sample time series. In the data-driven class, the analyst explores the dataset without a sample query, by just providing a specific length. The result consists of groups of similar sequences of the specified length.

```
Q2: OUTPUT SeasonalSim {X_p}
    FROM D
    WHERE seq=X_p | NULL
    MATCH = Exact(L)
```

Use Case: For the user-driven class, in the Stock Market application, an analyst can find all 30 days long subsequences of the Apple stock having similar prices. In the data-driven class an analyst can retrieve all the stocks whose prices were similar to each other over any 30 days periods.

Class III: Similarity threshold recommendations help the user better understand the dataset by transforming the intuition of "loose, medium or strict similarity" (introduced in Sec. 4.2) into actual parameter values. These terms are connected to the interpretations of similarity based on user and application domains.

```
Q3 OUTPUT ST
   FROM  D
   WHERE simDegree=NULL|L|M|S
   MATCH = Exact(L)|Any
```

Use case: Sometimes a user submits mining requests using different similarity thresholds, yet ends up receiving the same or very similar results. In such situations $Q3$ saves time and effort by allowing the user to make use of similarity thresholds that will cause a difference in the output. For example, if a user is not sure what value to use for a similarity threshold, then when prompted for the similarity strength they are interested in, they can enter "S" for strict similarity. The system will return a range of similarity thresholds for which the subsequences returned have very small similarity distances. We later explain in Sec. 5.2 how varying similarity thresholds can be managed without re-constructing the entire knowledge base.

5.2 Query Processing over ONEX Base

Based upon our solid formal foundation (Sec. 3.2), our query processor (Algo. 2) now applies time-warped strategies on the compact ED-based ONEX base. ONEX handles **similarity queries** by first exploring the R-Space using a three-step process. In Lines 4-7, we use GTI (Sec. 4.3) to retrieve groups of specific lengths. Then the processor uses LSI (Sec. 4.3) to find the representative with the minimum $\overline{DTW}$ to the sample sequence, the so-called the best matching representative. Third, we use the LSI to find the best match sequence inside the selected group.

Algorithm 2: Online Time Series Exploration

```
   2.A: Similarity Queries
   Input: Query Q,k, L, match
   Output: Sequences {X}
1  begin
2      {X} ⟵ ∅;
3      switch match do
4          case exact
5              {G_k^i} = GTI.getgroups(L);
6              minGp =
               LSI.compareRep(Q, G_k^i.Representatives);
7              X ⟵ LSI.getKSim(minGp);
8          case Any
9              for each L do
10                 {G_k^i} = GTI.getgroups(L);
11                 minGp =
                   LSI.compareRep(Q, G_k^i.Representatives);
12             {X} ⟵ LSI.getKSim(minGp);
13
14 2.B Seasonal Similarity Queries
   Input: queryType,TimeSeries (X_p, X_q), Length L
   Output: Subsequences X
15 begin
16     {G_k^i} = GTI.getgroups(L);
17     for each G ∈ G_k^i do
18         switch queryType do
19             case Single
20                 if X_p ∈ G then
21                     {X} ⟵ LSI.getSubsequences(X_p)
22             case NULL
23                 for each X_p ∈ G & X_q ∈ G do
24                     {X} ⟵ LSI.getSubsequences(X_p, X_q)
25
26 2.C Varying Similarity thresholds
   Input: Similaritythreshold ST',
          Inter − Representativedistance Dc
   Output: Groups {G'}
27 begin
28     G' ⟵ ∅;
29     if ST' == ST then
30         G' = G //return precomputed computed groups
31     else if ST' < ST then
32         split groups G
33     else
34         if ST' < Dc then
35             merge pair of groups with this condition
36         else if ST' ≥ Dc then
37             if ST' − ST ≥ Dc then
38                 merge groups with this condition
39             else if ST' < ST then
40                 G' = G //return precomputed computed groups
```

Processing costs for this class include: Cost($getGroups$),

Cost(*compareRep*) and Cost(*getKSim*). The complexity of Cost(*getGroups*) is constant for each specific length. The Cost(*compareRep*) is O(g) where g denotes the number of groups. The cost for finding the best matching representative is Cost(*getKSim*) which is O(m) where m is the number of subsequences in the best matching group. Thus the total complexity is $O(g) + O(m)$.

For **seasonal similarity queries** we distinguish two scenarios. In the first scenario (Lines 19-21), the analyst provides a sample time series *seq* and a specific length. ONEX explores the R-Space first using GTI (Sec. 4.3) to retrieve the groups of that specific length. After the groups are retrieved, they are explored using LSI (Sec. 4.3) based on the identifier of the sample time series. ONEX returns only the sequences in each group having the same sequence identifier as the sample *seq*. In the second scenario (Lines 22-24), if no sample sequence is provided, ONEX retrieves the groups of similar sequences of the length specified by the analyst.

Processing costs for this class include: Cost(*getGroups*) and Cost(*getSubsequences*). As shown in Sec. 4.3, we maintain the sequence identifiers for each group. So we can retrieve all subsequences in O(1). The complexity of Cost(*getGroups*) is constant for each specific length. The cost for finding sequences Cost(*getSubsequencess*) is O(n) where n is the number of subsequences in the group. The complexity for scenario two is $O(ng)$, where n is the number of time series in each group and g is the number of groups. The complexity for scenario one, seasonal similarity using a sample time series, is O(g). In both scenarios there is no additional storage overhead because all needed information is already contained in the index structures.

The complexity of the **similarity threshold recommendations** is highly dependent on the dataset, as the recommendations are based on the computations of the local and global similarity thresholds. We reduce this computation by re-using the pre-computed results for a specific ST and adapt them to newly provided ST' thresholds. For the construction of similarity groups (Sec. 3.1), we use a specific threshold value ST. In practice the optimal threshold varies from dataset to dataset. Such threshold offers the best trade-off between the accuracy of the results and the time response. We discuss in Sec. 6 how to empirically find these thresholds and we use them for our experiments. However, analysts might be interested in using a different domain-specific ST' in their queries. In such situations, the *R-Space* does not have to be re-constructed from scratch. Instead, we provide an efficient strategy to "refine" the similarity groups based on pre-computed information. We distinguish the following scenarios:

1. $ST' = ST$**.** In this case, the ST matches the one provided by the analyst. So the precomputed groups are used "as is" to find the best match.

2. $ST' < ST$**.** In this case, the precomputed groups contain similar sequences, but they must be refined. The intuition behind the refinement is that sequences that are similar for the threshold ST remain similar for a smaller threshold ST'. Groups are split in "smaller" similarity groups so no possible answer is missed. Assuming without loss of generality $ST/ST' = k$ where $k \in \mathbb{N}$, each precomputed group is now split into k groups. We use the same methodology for constructing these smaller groups as we originally used to construct the groups for specific lengths (Sec. 3.1).

3. $ST' > ST$**.** This is the more complex case, as some pairs of groups may merge, depending on the value of the Inter-Representative Distances (Dc) defined in Sec. 3.1.

3.1. $ST' < Dc$**.** In this case, all precomputed pairs of groups for which this condition is true are used to find the best match sequence. The intuition is that groups whose representatives are farther from each other by more than the given similarity threshold ST' can't merge. The rest of the groups will be processed as follows:

3.2. $ST' \geq Dc$**.** we distinguish two scenarios:

3.2a. $ST' - ST \geq Dc$. The pairs of precomputed groups for which this condition is true are merged and used to find the best match seq. The intuition is that a higher similarity threshold allows more sequences to be similar. The remaining groups are returned without any action. If the case that the same group could be merged with more than one group arises, we randomly choose a pair of qualifying groups and perform the merge. We then compute the Inter-Representative distance between the newly formed group and the remaining groups to determine if any additional mergers are needed. This process repeats in a cascading manner as long as the above condition holds.

3.2b. $ST' - ST < Dc$**.** In this case the precomputed groups are used "as is" to find the best match sequence.

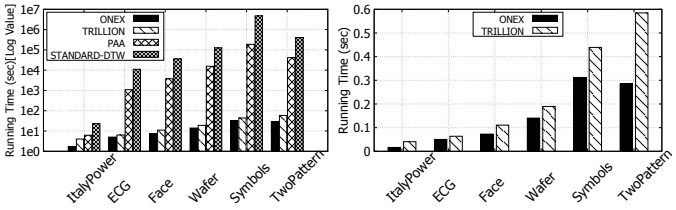
5.3 Optimizations for Processing Queries.

To efficiently retrieve the best match for a given sample sequence *seq*, we adopt optimizations for retrieving the best group representative and the best-match sequence within this group. The optimizations include the early abandoning of DTW [22] and stop the calculation if the lower bound exceeds the best-so-far distance, and pruning of the representatives. We combine these with the re-ordering of the early abandoning. Finally, we take advantage of the cascading lower bounds to prune unpromising candidates, a method also popular with DTW [11]. In addition, we develop the following optimizing strategies specific to ONEX:

- For a given sample sequence of length $\mathcal{L}$, we start the search for the best match representative with the ones of the same length as the query. If we don't find the "best match" representative for this length (the one with $\overline{DTW}$ to *seq* within ST/2), we continue by searching the representatives in decreasing order of their length until we reach the smallest length, followed by the search in increasing order of the length.

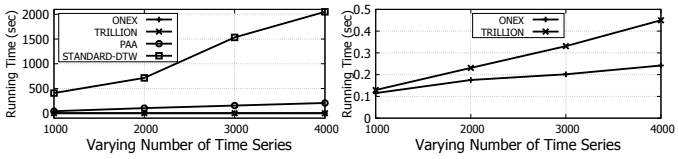
- Our strategy to find the "best match" representative of a specific length is optimized as well. We use the ordered array $S_i(k, sum_k)$ in GTI (Sec. 4.3) containing the sums of pairwise Inter-Representative distances. We find the "median representative" or the one whose sum is "in the middle" of this list and start our search with this representative. We follow up by alternately checking the closest representatives to its left and then right in the index, until we reach the representative with the min and respectively the max sum.

- The search inside the particular group with the best match representative could be optimized as follows: using the ordered array $ED_k(p, ED_p)$ containing the $\overline{ED}$ of the sequences in the group to the representatives (Sec. 4.3), we search for the "best match" sequence to *seq*. This is the sequence in the group whose $\overline{ED}$ to the representative has the closest value to the



(a) (b)

Figure 2: **Time Response for similarity queries**



(a) (b)

Figure 3: **Time Response for Similarity queries varying the number of time series in the dataset**

$\overline{DTW}$ between the *seq* and the representative. We follow up if needed by checking the neighboring sequences with smaller and respectively larger distances alternatively until we find the best match sequence.

6. EXPERIMENTAL EVALUATION

6.1 Experimental Setup

Experiments are conducted on a Windows machine with 3.35 GHz Intel Core i5 processor and 64GB GB of RAM. **Alternative State-of-the-Art Techniques.** We compare ONEX with three methods: the **Standard DTW** which is an exact brute-force method computing all pairwise distances and guaranteeing the best match, **PAA** (Piecewise Aggregate Approximation) [19] which finds an approximate solution by reducing the dimensionality of the data using an average approximation, and **Trillion**[22] which uses Lower Bounds and early abandoning steps to retrieve similar sequences. Trillion exhibits the fastest known-to-date response time. Their experiments show faster times than EBSM [2] for similarity queries [22] and many other competitors. For this and other reasons detailed in the Related Work section, we do not compare with EBSM, but rather with Trillion. The ONEX system and alternate methods PAA [19] and the standard DTW are implemented in C++ using Qt Creator with Clang 64-bit compiler. Trillion code is downloaded from the UCRsuite[4].

Datasets: We select our datasets from the UCR time-series collection[5], the largest collection of public datasets. Statistics of our datasets can be found in our Tech Report. To make meaningful comparisons between time series, they need to be normalized [22]. We normalize each sequence based on the maximum (max) and minimum (min) values in each dataset. For any sequence $X = (x_1...x_n)$, we compute the normalized values for each point x_i as $\frac{x_i - min}{max - min}$.

Performance Statistics. Our experimental evaluation is geared towards illustrating that ONEX is an interactive system offering results within less than one second for all datasets tested while yielding highly accurate results. The **online query processing time** and **accuracy** are reported for queries averaged over multiple runs (Sec.6.2). We evaluate the ONEX base by measuring the **size** and the **preprocessing time** of our pregenerated information when varying both datasets and similarity threshold ranges (Sec.6.3).

[4] *www.cs.ucr.edu/ ∼ eamonn/UCRsuite.html*
[5] *www.cs.ucr.edu/ ∼ eamonn/time_series_data*

6.2 Evaluation of Query Processing Time and Accuracy

6.2.1 Similarity Queries

This class retrieves the sequence closest to a given query sample *seq*. Standard DTW compares the query sample with all sequences in the dataset. PAA compares the query sample with the sequences in the dimensionality-reduced dataset. Trillion computes the lower bound of the query sample first and then uses the early abandoning optimization to find the best match of the same length as the query.

In our experiment we vary the length of the query sequence to cover a wide range of lengths. We compute the average **response time** for each system using 20 queries of different lengths chosen to cover a wide range from the smallest to the largest length. Further, our aim is to experiment both with query sequences are present and sequences that might not be present in the dataset. As our use cases motivate, it is important to find the best match for a query, regardless if the sequence is present or not in the dataset. In both cases we look to find the best match, namely the solution with the closest DTW to the query sequence. First we randomly select 10 subsequences of different lengths from each dataset and "promote" them to become query sequences. This is our query **"in the dataset"** part of the experiment. Then we adopt the methodology proposed in [13], where a random subsequence is chosen from each dataset to act as the query and is taken out from that dataset instead. We do this for 10 different subsequences. This is our query **"outside of the dataset"** part of the experiment. For each of them we run each individual query in the specific dataset 5 times and average the time response per query. We then compute the average time response for the 20 queries for each dataset.

As shown in Fig. 2, ONEX consistently outperforms both Standard DTW and PAA by several orders of magnitude and exhibits better running times than Trillion. The chart in Fig. 2a displays values multiplied by 100 on a logarithmic scale, while Fig. 2b zooms into the comparison of ONEX and Trillion. Although for small datasets ONEX and Trillion have fairly close response times, as the dataset size increases ONEX becomes on average 1.8 times faster than Trillion.

Furthermore, ONEX is highly accurate returning the best match for a query, regardless of the length of the match. Since Trillion only returns the best match of the same length as the query sequence, we designed an additional experiment by adjusting ONEX to only search for the best match of the same length as the query. We found that in this restricted situation the ONEX time response further improves, becoming on average 3.8 times faster than Trillion. The results for

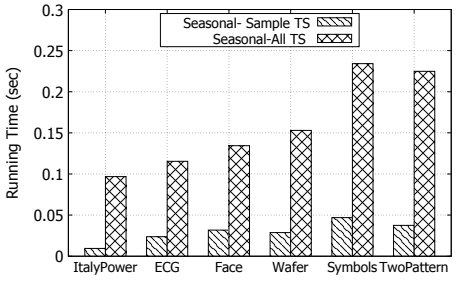


Figure 4: **Time response for Seasonal Similarity**

this particular experiment are displayed in Table 1, where ONEX-S denotes our ONEX returning the solution with the same length as the query similar to Trillion.

Next we evaluate **scalability** i.e., the impact of increas-

Table 1: **Time response similarity solution same length as query**

	Italy Power	*ECG*	*Face*	*Wafer*	*Symbols*	*Two Pattern*
ONEX-S	0.01	0.024	0.028	0.042	0.176	0.109
Trillion	0.04	0.063	0.11	0.189	0.439	0.585

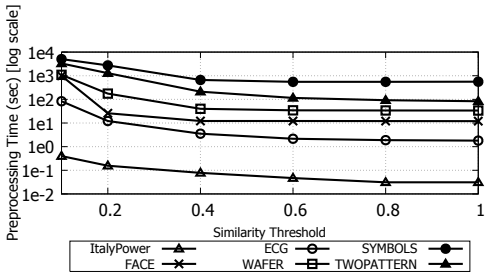


Figure 5: **Offline Construction Time Varying ST**

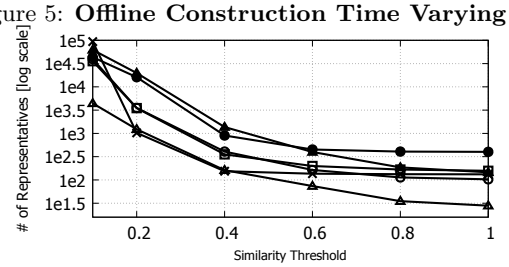


Figure 6: **Number of Representatives**

ing the number of time series on the running time for the StarLightCurves dataset. We randomly take subsets containing time series of length 100. The number of time series varies between 1000 and 5000 in increments of 1000. We employ the same methodology described above using query sequences both present and not present in the dataset. ONEX and Trillion's running times are close in magnitude and thus superimposed in Fig. 3a. As depicted in Figure 3a, the running times for Standard DTW and PAA increase drastically as the number of time series increases, while the time increase for ONEX and Trillion "seems" nearly constant. In reality, in Fig. 3b the zoomed detail shows that the Trillion

Table 2: **Accuracy for similarity solution of same length as query**

	Italy Power	*ECG*	*Face*	*Wafer*	*Symbols*	*Two Pattern*
ONEX-S	97.77	99.48	97.82	97.87	97.2	99.2
Trillion	82.97	74.58	71.87	87.67	96.99	88.04

Table 3: **Accuracy for similarity solution for any length**

	Italy Power	*ECG*	*Face*	*Wafer*	*Symbols*	*Two Pattern*
ONEX	99.47	99.81	98.74	99.48	98.28	98.54
Trillion	82.97	74.58	71.87	87.67	96.99	88.05
PAA	92.99	96.36	96.547	99.21	99.65	99.25

time response is up to 4 times slower than ONEX.

Solution Accuracy. We measure the **accuracy** of the solution as follows: we retrieve the DTW between the solution and the query sequence for ONEX, PAA and Trillion. We compute the error in retrieving each individual solution for each system as the difference between the distance computed by that system and the exact solution as provided by the brute force Standard DTW. We take the average of the error for each individual system and compute the accuracy as $(1-average(error))*100$. We use the same 20 queries as in the above experiment for similarity queries. The average accuracy for each system is displayed in Table 3. Since the brute-force always retrieves the best match possible and it is used as "accurate", we omit its results from the table.

As shown in Table 3, the ONEX accuracy is superior by an average of 19.5% to Trillion and 2% to PAA. While PAA is closer in accuracy with ONEX, the difference in the time response renders PAA impractical, that is, ONEX is 3 to 4 orders of magnitude faster than PAA. It is interesting to observe that Trillion has a high accuracy if the query sequence is "in the dataset", as it is the case for the first 10 queries used. That is due to the fact that Trillion, as described by its authors, performs an "exact search". However, the accuracy drops significantly for queries that do not exist in the dataset, i.e., the last 10 queries used. As before we also conduct a separate experiment restricting ONEX to retrieve the solution of the same length as the query sequence. In this experiment shown in Table 2 ONEX shows increased accuracy by an average of 12.6% compared to Trillion.

6.2.2 Seasonal Similarity Queries

This class returns the similar subsequences of a specific length that belong to a sample time series. Neither Standard DTW or PAA have been designed to search for this kind of similarity, thus we omit them from this experiment. Trillion is omitted because its optimizations solve the problem of finding only one closest sequence of the same length as the sample and cannot be applied to find subsequences whose similarity recurs. Since the length is specified by the analyst, we cover a wide range of lengths from the smallest to the largest. For the user-driven case we randomly select 5 time series from each dataset and we use 5 different lengths for each sample. For each sample time series, we run 5 times the seasonal similarity query for each chosen specific length and compute the average response time per length. We then average those response times for each sample time series. We repeat the experiment for each dataset and report the

average running times in Fig. 4. For the data-driven case we select 5 random lengths for each dataset and retrieve the groups with similar sequences of that length. We run 5 times for each length and compute the average time for retrieving the groups for that specific length. The average response times are reported in Fig. 4. We display an additional visual results in our Tech Report.

6.3 Evaluation of Preprocessing Performance

Since Standard DTW, PAA and Trillion don't involve a preprocessing phase, we display the preprocessing times and the size of the ONEX base for all datasets and for varying similarity thresholds. Fig. 5 displays the ONEX **offline construction time** for our datasets while varying the similarity threshold ST. As expected, for low similarity thresholds the construction time is higher because many groups are created. As the similarity threshold increases, fewer groups are created and thus more subsequences are grouped together. After a certain threshold the construction time remains constant. In Fig. 6 we display the **size of the pregenerated information** in ONEX in terms of number of representatives for varying similarity thresholds. Table 4 displays the number of representatives, the total number of subsequences highlighting the reduction in data cardinalty for all datasets and the index sizes. The index sizes (in MB) correspond to the number of representatives with the average space required per representative. The larger the similarity threshold, the smaller the number of representatives stored in the representative space (as depicted in Fig. 6). For example in the ItalyPower dataset the total number of representatives is 1228 for ST=0.2. The average space for the indexes in ItalyPower dataset is 1.14 MB. GTI uses 0.83 MB (0.11 MB for the group identifiers vector, 0.718 MB for the array containing inter-representative distances and 0.0004 MB for ST_{final} and ST_{half}). LSI needs 0.305 MB (0.28 MB for the sequence identifiers, 0.0046 MB for the representative vectors and 0.018 MB for the upper and lower bound envelopes for each representative). As observed

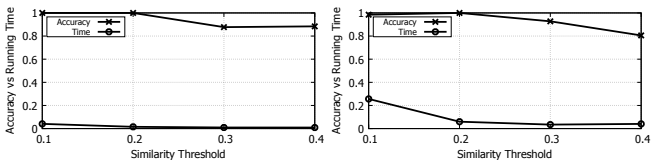
Table 4: **No. of Representatives, total no. of subsequences and size in MB for specific datasets**

DataSet	*Representatives*	*Subsequences*	*Size in MB*
ItalyPower	1228	18492	1.14
ECG	3532	931200	21.53
Face	4896	4768400	86.75
Wafer	3489	11476000	183.02
Symbols	3424	78607985	1210.32
Two Pattern	3961	33024000	513.41

in the four charts in Figures 7 & 8, each dataset has a particular similarity threshold for which the size of the pregenerated information and the construction time are best "balanced". We use this in combination with the observation of the best "trade-off" between accuracy and time to choose the most appropriate ST for specific datasets. For example, for most of our datasets these similarity thresholds are around 0.2. We indeed used these thresholds for our experiments reported in this paper. Due to space constraints we only showcase the results for four datasets.

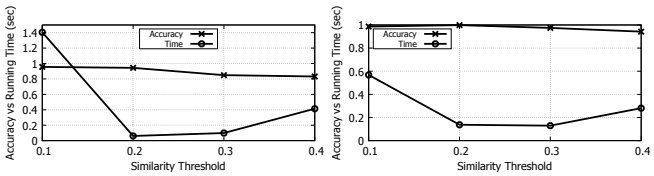
7. RELATED WORK

Many **data representation** techniques have been used for time series mining. Although the Euclidean Distance is



(a) ItalyPower (b) ECG

Figure 7: **Tradeoff accuracy vs time varying ST**



(a) Face (b) Wafer

Figure 8: **Tradeoff accuracy vs time varying ST**

known to be sensitive to distortion along the time axis[7], [19], it remains the most frequently used distances [12],[16],[31]. Its ubiquity is due to its ease of implementation and its time and space efficiency. The problem of time distortion is addressed by other distances including Dynamic Time Warping (DTW)[5], Longest Common Subsequence (LCSS)[15], Edit Distance with real penalty (ERP) [6].

To reduce the DTW time response, **indexing** techniques [12], [18], [31] and other optimizations like early abandoning of DTW [22], cascading lower bounds to prune unpromising candidates [11], and reversing the query/data role by creating an envelope around the query sequence instead of the data [22] were developed. These techniques are orthogonal to our work, and we indeed leverage them in our system.

Dimensionality reduction techniques include Discrete Fourier Transformation (DFT) [1], Single Value Decomposition (SVD) [30] and Piece Aggregate Approximation (PAA) [17]. Such techniques typically focus on guaranteeing no false dismissal and precision rather than tackling efficiency as their main goal. Some [1] are indexable, while others [17] are not, making them not scalable for large datasets.

Like ONEX, [2] uses a **preprocessing** phase for converting subsequence matching to vector matching using an embedding based on DTW. However, numerous parameters have to be supplied, such as the number of reference sequences and the number of split points.

Range searches and nearest neighbor searches [1] remain target queries for time series data. [11] answers such queries by finding representatives for groups of objects. Methods like the nearest centroid classifier [28] and k-means clustering generalize the nearest neighbor classifier by replacing the set of neighbors with their centroid. Conceptually similar, [4] and [14] **reduce the data cardinality** by grouping similar time series. [14] represents time series as trajectories in a multidimensional space and compares their structural similarity using multiscale comparison. A more recent warping-invariant-averaging condensation framework [21] creates a classifier based on DTW leading to improve-

ments in accuracy and reduced computation at run time. This method uses cluster centers based on 'DTW-average' of the sequences, while ONEX uses ED for clustering and a point-wise average sequence for representatives. [21] does data editing to make its classifier faster and more accurate, while we have to keep the original data to support accurate similarity searches and return actual sequence results. [3] introduces clusters constructed using DTW and a modified Density Peaks algorithm to improve the clustering performance. Using upper and lower bounds of DTW, TADPole can prune unnecessary computations. However its goal is clustering, not similarity search, which is our task at hand.

While ONEX adopts the general idea of finding representatives for groups of objects from [14] and [9], its solution for exploring similarity groups is unique, combining two well-known distances and showing to be highly effective compared to the state-of-the-art techniques.

Due to space constraints we provide additional general background information and notations, as well as discussions about alternative clustering methods, maintenance of ONEX base, general query syntax and seasonal similarity experiments at:
https : //github.com/onexAdditional/ONEX_Additional

8. CONCLUSION

ONEX emerges as a truly interactive time series exploration system. Our unique approach based on the combination of two similarity distances leads to improvements in accuracy of up to 19% and up to 3.8 times shorter time responses compared to the fastest known state-of-the-art method. ONEX renders more practical the exploration of large time series datasets and helps analysts better understand similarity by supporting novel classes of operations.

Acknowledgments *We thank Dr. Eamonn Keogh from UC Riverside for his invaluable and always prompt feedback, detailed suggestions on this paper, and for sharing datasets and code. We thank M. Andrews and S. Swartz from WPI for help with the implementation, and Dr. Athitsos from Univ. of Texas at Arlington for sharing materials. We thank Fulbright and the CS Dept. at WPI for financial support, and NSF for grants IIS-1018443 and CRI-1305258.*

9. REFERENCES

[1] R. R. Agrawal, Faloutsos, et al. *Efficient similarity search in sequence databases*. Springer, 1993.

[2] V. Athitsos, P. Papapetrou, M. Potamias, G. Kollios, and D. Gunopulos. Approximate embedding-based subsequence matching of time series. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 365–378. ACM, 2008.

[3] N. Begum, L. Ulanova, J. Wang, and E. Keogh. Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 49–58. ACM, 2015.

[4] L. Belbin. The use of non-hierarchical allocation methods for clustering large sets of data. *Australian Computer Journal*, 19(1):32–41, 1987.

[5] D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, 1994.

[6] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *VLDB*. VLDB Endowment, 2004.

[7] S. Chu, E. J. Keogh, D. M. Hart, M. J. Pazzani, et al. Iterative deepening dynamic time warping for time series. In *SDM*, pages 195–212. SIAM, 2002.

[8] T. H. Cormen. *Introduction to algorithms*. MIT press, 2009.

[9] G. Das et al. Rule discovery from time series. In *KDD*, 1998.

[10] E. Dimitriadou, A. Weingessel, and K. Hornik. A combination scheme for fuzzy clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(07):901–912, 2002.

[11] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E. Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

[12] C. C. Faloutsos, Ranganathan, et al. *Fast subsequence matching in time-series databases*. ACM, 1994.

[13] A. W.-C. Fu, E. Keogh, L. Y. Lau, C. A. Ratanamahatana, and R. C.-W. Wong. Scaling and time warping in time series querying. *The VLDB JournalThe International Journal on Very Large Data Bases*, 17(4):899–921, 2008.

[14] S. Hirano et al. Cluster analysis of time-series medical data based on the trajectory representation and multiscale comparison techniques. In *ICDM*. IEEE, 2006.

[15] J. W. Hunt and T. G. Szymanski. A fast algorithm for computing longest common subsequences. *Communications of the ACM*, 20(5):350–353, 1977.

[16] E. Keogh, K. Chakrabarti, et al. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM SIGMOD Record*, 30(2):151–162, 2001.

[17] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and information Systems*, 3(3):263–286, 2001.

[18] E. Keogh and C. A. Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

[19] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289. ACM, 2000.

[20] J. Kruskall and M. Liberman. The symmetric time warping algorithm: From continuous to discrete. time warps, string edits and macromolecules, 1983.

[21] F. Petitjean, G. Forestier, G. I. Webb, A. E. Nicholson, Y. Chen, and E. Keogh. Faster and more accurate classification of time series by exploiting a novel dynamic time warping averaging algorithm. *Knowledge and Information Systems*, 47(1):1–26, 2016.

[22] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh. 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*, pages 262–270. ACM, 2012.

[23] R. Rivest. The md5 message-digest algorithm. 1992.

[24] E. J. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes. Correlating financial time series with micro-blogging activity. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 513–522. ACM, 2012.

[25] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[26] Y. Sakurai, C. Faloutsos, and M. Yamamuro. Stream monitoring under the time warping distance. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 1046–1055. IEEE, 2007.

[27] J. M. Steele. *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press, 2004.

[28] R. Tibshirani, T. Hastie, B. Narasimhan, and G. Chu. Diagnosis of multiple cancer types by shrunken centroids of gene expression. *Proceedings of the National Academy of Sciences*, 99(10):6567–6572, 2002.

[29] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684. IEEE, 2002.

[30] D. Wu, A. Singh, D. Agrawal, A. El Abbadi, and T. R. Smith. Efficient retrieval for browsing large image databases. In *Fifth international conference on Information and knowledge management*, pages 11–18. ACM, 1996.

[31] B.-K. Yi and C. Faloutsos. Fast time sequence indexing for arbitrary lp norms. VLDB, 2000.