Interactive Time Series Exploration Powered by the Marriage of Similarity Distances

Rodica Neamtu, Ramoza Ahsan, Elke Rundensteiner and Gabor Sarkozy

Computer Science Department, Worcester Polytechnic Institute, Worcester MA, USA
rneamtu, rahsan, rundenst, gsarkozy @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) that employs 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 Dynamic Time Warping distance by applying it over the surprisingly compact knowledge base instead of the raw data. Our comparative study reveals that ONEX is up to 20% 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
In applications ranging from finance, business, and medicine to meteorology [14],[25], time series data is prevalent, presented as stock fluctuations, ECG, rainfall amounts, etc. 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. Then they 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. If only close matches are found, they indicate slightly different impacts in specific states. We conclude that 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. Analysts also searched for 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 parameter 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 N(n – 1)/2. For example, a benchmark dataset like StarLightCurves from the UCR
Time Series Data Mining Archive\textsuperscript{1} with 9236 time series, each of length 1024, is composed of 4.83e\textsuperscript{9} subsequences. Real-world datasets tend to be orders of magnitude larger than this\textsuperscript{2}. 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], [27] 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 timeliness. However, their requirement for setting many different parameters limits their efficiency [23].

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], [17], [31] to achieve 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 [23], [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], [23], [27] 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], [16] 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.

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 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 subsequences and preserve 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. 3.2).

4. Our experimental performance shows that ONEX is several times faster than the fastest known method [23] and many orders of magnitude faster than [19] and [27]. In addition, ONEX is up to 20% more accurate than [23] 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 $(X_p)_j$, is a time series of length $i$ starting at position $j$ where $1 \leq j \leq n$ and $0 \leq j \leq n - 1$.

We discuss in this paper the Euclidean Distance (ED) and Dynamic Time Warping (DTW)\textsuperscript{[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:

$$ED(X, Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$ (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.** For below, 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 \( P = (p_1, p_2, ..., p_t, ..., p_T) \), where \( n \leq T \leq 2n-1 \), \( p_1 = (1,1) \) and \( pr = (n,m) \).

**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:

\[
w(P) = \sum_{t=1}^{T} w_{i_t,j_t} \tag{2}\]

The **DTW distance** between \( X \) and \( Y \), denoted by \( DTW(X,Y) \), is defined to be the weight of the path \( P \) of the path that has the minimum weight among all possible warping paths, i.e., \( \min_{\forall P} (w(P)) \).

This path can be calculated using dynamic programming [26] (See Fig 1). More details can be found in [20, 23].

![Figure 1: Warping Path](image)

**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 \( ST \); namely, \( Dist(X,Y) \leq ST \) 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:

\[
ED(X,Y) = ED(X,Y)/\sqrt{n} = \sqrt{1/n \sum_{i=1}^{n} (x_i - y_i)^2} \tag{3}
\]

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** \( DTW \) is defined as

\[
DTW(X,Y) = DTW(X,Y)/2n \quad \tag{4}
\]

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

In this section, 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 \( R^*_i \) of \( S \) is defined as the point-wise average of the sequences in the set \( S \) [10]. That is, \( R^*_i = \text{avg}(X_{ij}) \), for all \( (X_{ij}) \in S \).

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 \( \{X_{ij}\} \) of the time series of dataset \( D \), assume these subsequences \( \{X_{ij}\} \in T \) are grouped into similarity groups with their respective representatives \( R^*_i \), such that all subsequences \( \{X_{ij}\} \in T \) are in one and only one group \( G^*_i \). These similarity groups are defined to be **ONEX similarity groups**, denoted by \( G^*_i \), if the following three properties hold:

1. all subsequences \( \{X_{ij}\} \) in a group \( G^*_i \) must have the same length \( i \).
2. \( ED \) between any subsequence \( \{X_{ij}\} \) in \( G^*_i \) and the representative \( R^*_i \) of this group \( G^*_i \) is smaller than half of the similarity threshold \( ST \) used by the system, that is \( ED((X_{ij}), R^*_i)) \leq ST/2 \), \forall j \in [1,m] \forall p \in [1,n].
3. \( ED \) between the subsequence \( \{X_{ij}\} \) and the representative \( R^*_k \) of the group \( G^*_k \) is the smallest compared to \( ED \) of \( \{X_{ij}\} \) and all other representatives \( R^*_i \) of the same length \( i \) defined over \( D \), or \( ED((X_{ij}), R^*_i)) \leq ED((X_{ij}), R^*_j)) \| \forall i, j \in [1,m] \forall p \in [1,n] \forall l \in [1,q] \), where \( q \) denotes the number of representatives of length \( i \).

The key requirements for placing sequences into the same ONEX similarity group are two-fold. First, \( ED \) of the sequences to the representative of the group must be the smallest compared to the \( ED \) 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 \( G^*_i \), with \( G^*_i \) defined in Def. 8, the \( ED(X,Y) \) defined in Def. 5 is within the threshold \( ST \), that is, \( ED(X,Y) \leq ST \), for all \( X, Y \in G^*_i \).
Let $X = (x_i...x_j)$ and $Y = (y_i...y_j)$ be two time series in the same similarity group $G_k$ and $R = (r_i...r_j)$ be the representative of that group. According to Def. 8, we have: $ED(X, R) \leq ST/2$ and $ED(Y, R) \leq ST/2$

Using Def. 5, this means:

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

*Equation (5)*

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

*Equation (6)*

We want to prove that $ED(X, Y) \leq ST$ when means:

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

*Equation (7)*

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

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

*Equation (8)*

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

*Equation (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 [28], we get:

$$\sum_{k=i}^{j} (x_k - y_k)^2 \leq 2 \sum_{k=i}^{j} (x_k - r_k)^2 + 2 \sum_{k=i}^{j} (r_k - y_k)^2.$$  

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

$$\sum_{k=i}^{j} (x_k - y_k)^2 \leq 2 \sum_{k=i}^{j} (x_k - r_k)^2 + 2 \sum_{k=i}^{j} (y_k - r_k)^2$$  

*Equation (10)*

$$\leq 2(\frac{ST^2}{4}) + 2(\frac{ST^2}{4}) = ST^2.$$  

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

As suggested in Fig. 2, we "represent" each group constructed over a data set 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** ( R-Space).

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

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

This distance plays a central role in dealing with variable similarity thresholds, as shown in Sec. 4.3.

### 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 $DTW$ between the seq and the representative $R_k$ is smaller than $ST/2$, then we can guarantee that all subsequences in that group $G_k$ are similar to this sequence seq and that $DTW$ between seq and any of these sequences is within the similarity threshold $ST$.

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

**Proof. (Case: subsequences of same length):**

We know from Def. 1, Lemma 1 and assumptions of Lemma 2:

$$ED(Y, Y^r) = \sqrt{\sum_{i=1}^{n} (y_i - y_i^r)^2} \leq \sqrt{n \frac{ST^2}{4}}.$$  

Squaring this we get:

$$ED^2(Y, Y^r) = \sum_{i=1}^{n} (y_i - y_i^r)^2 \leq n \frac{ST^2}{4}.$$  

*Equation (11)*

We define matrices $M(X, Y)$ and $M(X, Y^r)$ 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 $M(X, Y^r)$ with weight at most $2nST$. In fact we will show that the same warping path $P$ from $(M, Y^r)$ will be good. Let $P$ be $P = (p_1, p_2, . . . , p_T)$, where $n \leq T \leq 2n - 1$, $p_1 = (1, 1)$, $p_T = (n, n)$, $p_i = (i, j^r)$. Then from the assumptions of Lemma 2, we have:

$$\sum_{t=1}^{T} (x_i - y_i) \leq 2n \frac{ST}{2} = nST.$$  

Squaring this, we get:

$$\sum_{t=1}^{T} (x_i - y_i)^2 \leq 2n \frac{ST^2}{2} = nST^2.$$  

*Equation (12)*

We want to prove that:

$$\sqrt{\sum_{t=1}^{T} (x_i - y_i^r)^2} \leq 2nST.$$  

This is the same as:

$$\sum_{t=1}^{T} (x_i - y_i^r)^2 \leq 4n^2ST^2.$$  

*Equation (13)*

We have:

$$x_i - y_i^r = x_i - y_i + y_i - y_i^r = (x_i - y_i) + (y_i - y_i^r).$$

Using the Cauchy-Schwarz inequality [28], we get:

$$\sum_{i=1}^{T} (x_i - y_i^r)^2 \leq 2 \sum_{i=1}^{T} (x_i - y_i)^2 + 2 \sum_{i=1}^{T} (y_i - y_i^r)^2.$$  

Then using this and (12), we get:

$$\sum_{t=1}^{T} (x_i - y_i)^2 \leq 2 \sum_{t=1}^{T} (x_i - y_i)^2 + 2 \sum_{t=1}^{T} (y_i - y_i^r)^2$$  

*Equation (14)*

$$\leq 2n^2ST^2 + 2 \sum_{t=1}^{T} (y_i - y_i^r)^2.$$  

In order to estimate the second term, note that it is $\sum_{i=1}^{n} (y_i - y_i^r)^2$ 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:

\[
\sum_{t=1}^{T} (x_{t}-y_{t})^2 \leq 2n^2 ST^2 + 2n \sum_{i=1}^{n} (y_{i}-y_{i}^*)^2 \leq 2n^2 ST^2 + \frac{2nnST^2}{4}.
\]

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

\[
\frac{nnST^2}{2} \leq 2n^2 ST^2.
\]

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

**Proof sketch (Case: subsequences of different lengths)**

Let \( Y \) and \( Y^* \) be 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 \( m < n \) but the proof is very similar for \( n \leq m \). In the \( D^2TW \) 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. We note that Lemma 2 guarantees that our solution is always within \( ST \) of the exact solution.

**Discussion: DTW Clustering in ONEX framework.**

We now show that using DTW for ONEX similarity group formation would require us to guarantee the triangle inequality for DTW, which is still an open research problem [6], [21] due to the non-metric nature of DTW. This, along with the efficiency of ED for clustering compared to DTW, confirms our ED-DTW design choice of the framework.

Let’s assume that some clustering method were to exist that would create clusters as in our Def. 8, i.e. it is guaranteed to place all subsequences in similarity groups for which the following conditions hold:

(1) \( D^2TW(X,Y) \leq ST \), where \( X \) and \( Y \) are any sequences in the same group and \( ST \) is the given similarity threshold.

(2) \( D^2TW(X,R) \leq ST/2 \), where \( R \) is the representative of a group and \( X \) is any sequence in that group.

In other words, our ONEX framework would have to work now with an “adjusted” Lemma 2, where \( ED \) would be replaced by DTW. For this important foundation of our ONEX methodology to still be applicable, we would have to prove the following:

If \( D^2TW(Y,Y^*) \leq ST/2 \) and \( D^2TW(X,Y) \leq ST/2 \) then we have \( D^2TW(X,Y^*) \leq ST \), which unfortunately corresponds to the triangle inequality for DTW. □

In conclusion, using a clustering methodology based on DTW would impede our framework’s functionality, rendering its core formal foundation unproven.

### 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 \( ED \) 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 \( G_{k} \) and their representatives \( R_{k} \) for all possible lengths.

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

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{Similarity threshold } ST ), TimeSeries ( {X} ), Length ( \mathcal{L} )</td>
<td>( {\text{Representatives } {R}, \text{Groups } {G}} )</td>
</tr>
</tbody>
</table>

\[
\text{begin}
\begin{align*}
\{G\} &= \emptyset, \{R\} = \emptyset \\
\text{RandomizedINPLACE}(); \\
\{X\} &= \text{all subsequences of length } \mathcal{L} \\
\text{minSM}=0, \text{mink}=0 \\
\text{for } X_{k} \in \{X\} \text{ do} \\
& \quad \text{if } \{G\} = \emptyset \text{ then} \\
& \quad \quad G_{k} \leftarrow X_{k} \\
& \quad \quad G_{k} \leftarrow X_{k} \\
& \quad \quad R_{k} \leftarrow X_{p} \\
& \quad \quad \text{else} \\
& \quad \quad \text{for } k=1 \text{ to Representative.count do} \\
& \quad \quad \quad \text{mink}=\text{closest Representative distance} \\
& \quad \quad \quad \text{if } (\text{minSM} \leq (\text{sqrt} \mathcal{L} \times ST/2)) \text{ then} \\
& \quad \quad \quad \quad G_{\text{mink}} \leftarrow X_{p} \\
& \quad \quad \quad \quad \text{update } R_{\text{mink}} \\
& \quad \quad \text{else} \\
& \quad \quad \quad G_{k} \leftarrow G_{K+1} \text{ /new Group} \\
& \quad \quad \quad R_{k+1} \leftarrow X_{p} \\
\end{align*}
\text{end}
\]

The complexity of the R-Space construction is \( O(nl^{2}g) \) 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 \( O(n^{3/2}) \). 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 \( 1/(1/(k+1)) = k+1 \) [24]. Thus \( \sum_{l=1}^{\infty} (k+1) = n \); the expected number of groups is \( O(\sqrt{n}) \). Then the complexity becomes \( O(n^{3/2}) \), which is much better than \( O(n^{3}) \).

### 4.2 Discussion of ONEX Base

#### 4.2.1 ONEX Base Construction

While above we have introduced a viable and robust solution for ONEX base construction, the question arises if alternate clustering methods based on ED could equally be utilized in the context of our framework. The key observation here is that any clustering solution we employ must observe our core group requirements, namely, they must produce clusters with a maximum diameter equal to \( ST \) and have centers (our representatives) whose \( ED \) to any sequence in
the group is less than \(ST/2\). While the well-known \textit{kmeans} algorithm appears to be a contender on first sight, we note that there are key differences: (1) \textit{K-means} must know the fixed number of clusters in advance, while we give a lower bound to the number of groups, but we expect it to grow. (2) \textit{K-means} is batch oriented, while Algorithm 1, after a small setup batch mode, is online. In a sense, Algorithm 1 is similar with online nearest neighbor clustering or the on-line \textit{k}-center problem.

\subsection{ONEX Base Maintenance under Updates}

Next, we sketch strategies for accommodating insertions and deletions of time series. If a time series is deleted, the groups can be updated to remove any sequences with that specific time series id. These groups don’t need to be re-constructed. However, the representatives (and their envelopes) have to be re-computed according to the sequences that remain in the group. We keep the locations of all objects in an index, thus if we delete an object, we can find it in constant time and set it to NULL. In short, deletions, so long as they are relatively rare (say less than 10\% of all objects), have almost no cost, and almost no effect on speed or accuracy. If a time series is inserted into the dataset, some groups have to be updated. When possible we use strategies that avoid re-construction of the groups. One possibility is to use the original ONEX ED based methodology to place the subsequences of the new time series into groups. The groups into which the new subsequences are placed are updated to include the new sequence, and their representatives are recomputed. Another option is to use each subsequence of the inserted time series as a query sample and to find its best match representative using the ONEX approach (See Section 5.2). Then we insert the subsequence into this group and recompute the representative.

\subsection{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 \textit{Similarity Parameter Space} (SP-Space).

**Definition 11.** The \textit{SP-Space} is a conceptual space with length \(L\) and similarity threshold \(ST\) as its dimensions.

We build an \textit{SP-Space} for each specific length based on establishing relevant values for similarity thresholds. These could be \(ST_{\text{half}}\) and \(ST_{\text{final}}\) indicating when a half and respectively all precomputed groups of subsequences of a specific length merge – resulting in major differences in the result set. Alternatively, the ranges of relative similarity thresholds might be “customized” by imposing a certain granularity of the ranges, i.e., by splitting the overall range in 10 equal sized ranges. 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. 3 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 \(ST_{\text{half}} = 0.5\) and \(ST_{\text{final}} = 0.78\). 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

---

3\text{http://cseweb.ucsd.edu/diasgupta/291geom/streaming.pdf}
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 provides insights into the dataset without the user giving a sample sequence as a target match.

Q: OUTPUT \( X_p \) 
FROM D 
WHERE Sim <= ST, [seq = q | NULL] 
MATCH = Exact(L) | Any

The above general ONEX query syntax uses clauses described in Table 2.

Table 2: List of clauses

<table>
<thead>
<tr>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATCH</td>
<td>Exact(L) refers to a specific length L. Any refers to any length.</td>
</tr>
<tr>
<td>( X_p )</td>
<td>Subsequence of the time series p</td>
</tr>
<tr>
<td>seq</td>
<td>NULL means no sample sequence is given. ( X_p, q ) - samples sequences provided by user</td>
</tr>
<tr>
<td>Sim</td>
<td>Similarity Distance</td>
</tr>
<tr>
<td>ST</td>
<td>Similarity Threshold</td>
</tr>
<tr>
<td>L</td>
<td>Length</td>
</tr>
</tbody>
</table>

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. 4.3) 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 | LIMIS 
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. 4.4) to retrieve groups of specific lengths. Then the processor uses LSI (Sec. 4.4) to find the representative with the minimum 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.

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. 4.4) to retrieve the groups of that specific length. After the groups are retrieved, they are explored using LSI (Sec. 4.4) 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 subsequences of the length specified by the analyst. We retrieve the groups using GTI and then we find the sequences in each group using LSI.

The processing costs for this class include: Cost(getGroups) and Cost(getSubsequences). As shown in Sec. 4.4, 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(getSubsequences) is \( O(n) \) where \( n \) is the number of subsequences in the group. The
Algorithm 2: Online Time Series Exploration

2.A. Similarity Queries
Input: Query Q,k, L, match
Output: Sequences X

begin
\(X\) \(\leftarrow\) \(\emptyset\);
switch match do
\{G\}_k = GT1.getgroups(L);
minGp = LSI.compareRep(Q, \{G\}_k, Representatives);
\(X\) \(\leftarrow\) LSI.getKSim(minGp);
\}
case exact
\{
\}
case Any
\{for each \(L\) do
\{\(G\}_L = GT1.getgroups(L);
minGp = LSI.compareRep(Q, \{G\}_L, Representatives);
\(X\) \(\leftarrow\) LSI.getKSim(minGp);
\}
\}
end

2.B Seasonal Similarity Queries
Input: queryType, TimeSeries \((X_p, X_q)\), Length \(L\)
Output: Subsequences \(X\)

begin
\{\(G\}_L = GT1.getgroups(L);
for each \(G \in \{G\}_L\) do
switch queryType do
\{if \(X_p \in G\) then
\{\(X\) \(\leftarrow\) LSI.getSubquences\((X_p)\)
\}
\}
case NULL
\{for each \(X_p \in G \& X_q \in G\) do
\{\(X\) \(\leftarrow\) LSI.getSubquences\((X_p, X_q)\)
\}
\}
end

2.C Varying Similarity thresholds
Input: SimilarityThreshold ST, Inter-Representative Distance \(Dc\)
Output: Groups \(\{G\}\)

begin
\(G\) \(\leftarrow\) \(\emptyset\);
if \(ST = ST\) then
\{\(G\) \(= G\) /return precomputed computed groups
\}
else if \(ST < ST\) then
\{split groups \(G\)
\}
else if \(ST < Dc\) then
\{merge pair of groups with this condition
\}
else if \(ST > Dc\) then
\{merge groups with this condition
\}
else if \(ST < ST\) then
\{\(G\) \(= G\) /return precomputed computed groups
\}
end

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 [23] 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 \(L\), 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 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 \(S(4, sum)\) in GTI (Sec. 4.4) containing the sums of pairwise Inter-Representative distances.

complexity for scenario two is \(O(n g)\), 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 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. \(ST > ST\). 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. \(ST > ST\). 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.
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 $ED$ of the sequences in the group to the representatives (Sec. 4.4), we search for the “best match” sequence to $seq$. This is the sequence in the group whose $ED$ to the representative has the closest value to the $DTW$ 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 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 [23] 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 [23] 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.

Datasets: We select our datasets from the UCR time-series collection\(^5\), the largest collection of public datasets. We display their statistics in Table 3. To make meaningful comparisons between time series, they need to be normalized [23]. We normalize each sequence based on the maximum (max) and minimum (min) values in each dataset. For any sequence $X = \left\{ x_1, \ldots, x_n \right\}$, we compute the normalized values for each point $x_i$ as $\frac{x_i - \text{min}}{\text{max} - \text{min}}$.

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).

6.2 Evaluation of Query Processing Time and Accuracy

\(^5\) www.cs.ucr.edu/~eamonn/UCRsuite.html

\(^6\) www.cs.ucr.edu/~eamonn/time_series_data

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 that 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. 4, ONEX consistently outperforms both Standard DTW and PAA by several orders of magnitude and exhibits better running times than Trillion. The chart in Fig. 4a displays values multiplied by 100 on a logarithmic scale, while Fig. 4b 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.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Number</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItalyPower</td>
<td>200</td>
<td>20</td>
</tr>
<tr>
<td>ECG</td>
<td>560</td>
<td>10</td>
</tr>
<tr>
<td>Face</td>
<td>1060</td>
<td>131</td>
</tr>
<tr>
<td>Waseda</td>
<td>995</td>
<td>152</td>
</tr>
<tr>
<td>Symbol</td>
<td>4000</td>
<td>129</td>
</tr>
<tr>
<td>Two Patterns</td>
<td>9236</td>
<td>1024</td>
</tr>
<tr>
<td>Star</td>
<td>4</td>
<td>1e0</td>
</tr>
<tr>
<td>Light</td>
<td>560</td>
<td>1</td>
</tr>
<tr>
<td>Curves</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3: Datasets Statistics
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 3.7 times faster than Trillion. The results for this particular experiment are displayed in Table 4, 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 increasing the number of time series on the running time for the similarity queries using 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. 5a. As depicted in Figure 5a, 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. 5b the zoomed detail shows that the Trillion 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 6. 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 6, the ONEX accuracy is superior by an average of 20% to Trillion and 6.5% 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 5 ONEX shows increased accuracy by an average of 14.79% 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. We explore seasonal similarity based on 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. 6. The average response times, regardless of the length of the sample query or the size of the dataset are very small, less than 0.04 seconds for any sample sequence.

For the data-driven case we select 5 random lengths for

<table>
<thead>
<tr>
<th>Similarity Threshold</th>
<th>similarity solution same length as query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Italy Power</td>
<td>ECG</td>
</tr>
<tr>
<td>ONEX-S</td>
<td>0.01</td>
</tr>
<tr>
<td>Trillion</td>
<td>0.04</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Similarity Threshold</th>
<th>offline Construction Time Varying ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Italy Power</td>
<td>ECG</td>
</tr>
<tr>
<td>Preprocessing Time (sec) [log scale]</td>
<td></td>
</tr>
</tbody>
</table>

| # of Representatives [log scale] |

| # of Representatives [log scale] |

| Table 5: Accuracy for similarity solution of same length as query |
|----------------------|----------------------------------------|
| Italy Power | ECG | Face | Wafer | Symbols | Two Pattern |
| ONEX-S | 97.74 | 99.48 | 97.82 | 97.87 | 97.2 | 99.2 |
| Trillion | 82.97 | 74.58 | 71.87 | 87.07 | 96.99 | 88.04 |

| Table 6: 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 | 98.74 |
| Trillion | 82.97 | 74.58 | 71.87 | 87.07 | 96.99 | 88.05 |
| PAA | 92.99 | 96.36 | 96.64 | 99.21 | 99.66 | 99.29 |
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 data mining. Although the Euclidean Distance metric is known to be sensitive to distortion along the time axis[7], [19], it remains one of the most frequently used distances [12],[17],[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 the Dynamic Time Warping (DTW)[5], Longest Common Subsequence (LCSS)[15], Edit Distance with real penalty [23], [31] and other optimizations like early abandoning candidates [11], and reversing the query/data role by creating an envelope around the query sequence instead of the data [23] 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) [16]. The basic idea is to approximate the original sequences by mapping them into low-dimensional points. 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 [16] are not, making them not scalable for large datasets.

Like ONEX, [2] uses a preprocessorizing 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.

that the average of a set of objects is more representative than any object from that set is popular. For instance, methods like the nearest centroid classifier [29] 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 multiscalar comparison. A more recent warping-invariant-averaging condensation framework [22] creates a classifier based on DTW – leading to improvements 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. [22] 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 support of 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.

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 20% and up to 4 times shorter time responses compared to the fastest known state-of-the-art method. Our system renders more practical the exploration of large time series datasets and helps analysts better understand the similarity of time lines 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 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 HS-1018443 and CRI-1305258.

9. REFERENCES