Data Min Knowl Disc (2017) 31:809–850
DOI 10.1007/s10618-016-0489-3

On searching and indexing sequences of temporal intervals

Orestis Kostakis[1] · **Panagotis Papapetrou**[2]

Received: 28 December 2015 / Accepted: 28 November 2016 / Published online: 16 January 2017

Abstract In several application domains, including sign language, sensor networks, and medicine, events are not necessarily instantaneous but they may have a time duration. Such events build sequences of temporal intervals, which may convey useful domain knowledge; thus, searching and indexing these sequences is crucial. We formulate the problem of comparing sequences of labeled temporal intervals and present a distance measure that can be computed in polynomial time. We prove that the distance measure is metric and satisfies the triangle inequality. For speeding up search in large databases of sequences of temporal intervals, we propose an approximate indexing method that is based on embeddings. The proposed indexing framework is shown to be contractive and can guarantee no false dismissal. The distance measure is tested and benchmarked through rigorous experimentation on real data taken from several application domains, including: American Sign Language annotated video recordings, robot sensor data, and Hepatitis patient data. In addition, the indexing scheme is tested on a large synthetic dataset. Our experiments show that speedups of over an order of magnitude can be achieved while maintaining high levels of accuracy. As a result of our work, it becomes possible to implement recommender systems, search engines and assistive applications for the fields that employ sequences of temporal intervals.

Responsible editor: Eamonn Keogh.

✉ Orestis Kostakis
orestis.kostakis@aalto.fi

Panagotis Papapetrou
panagiotis@dsv.su.se

[1] Aalto University, Espoo, Finland

[2] Stockholm University, Stockholm, Sweden

Keywords Temporal intervals · Event-interval sequences · Indexing temporal interval sequences · Embeddings

1 Introduction

Sequences of event-intervals exist in many application domains, such as human motion databases, sign language data, human activity monitoring, and medical data. Their main advantage over traditional sequences, which model series of instantaneous events, is that they incorporate the notion of duration in their event representation. Due to this, they are used in a broad range of research fields such as geo-informatics (Pissinou et al. 2001), cognitive science (Berendt 1996), linguistic analysis (Bergen and Chang 2005), music informatics (Pachet et al. 1996), and medicine (Kosara and Miksch 2001). Essentially, sequences of temporal intervals can be encoded as a collection of labeled events accompanied by their start and end time values. An example of a sequence of five labeled temporal intervals is presented in Fig. 1.

So far, most studies on sequences of temporal intervals have been focusing on the aspect of knowledge discovery, such as mining patterns and association rules that might be of interest to domain experts (Kam and Fu 2000; Ale and Rossi 2000). Surprisingly, very limited work has been performed on comparing event-interval sequences (Kostakis et al. 2011a, b; Kotsifakos et al. 2013).

Many data mining algorithms and application domains require similarity comparisons as a subroutine (Rakthanmanon et al. 2012), and the need for fast and accurate similarity search becomes more imminent when the data is characterized by more complex structure, such as for the case of sequences of temporal intervals. Once a framework for comparing temporal-interval sequences is defined, it will enable many types of clustering and classification algorithms, and will facilitate the implementation of crucial solutions, such as recommender systems, phylogenies, and assistive applications.

Examples In the biomedical domain, e-sequences are used to encode the health records of patients; the duration of their symptoms and prescribed medication (Patel et al. 2008). A direct application of distance functions of event-interval sequences is the search of similar patient histories among large databases of health records. The aim is to facilitate physicians and medical researchers. Employing an interval sequence-based temporal abstraction on multi-variate time series representing the medical history of patients (Moskovitch and Shahar 2014a) has shown substantial improvement in

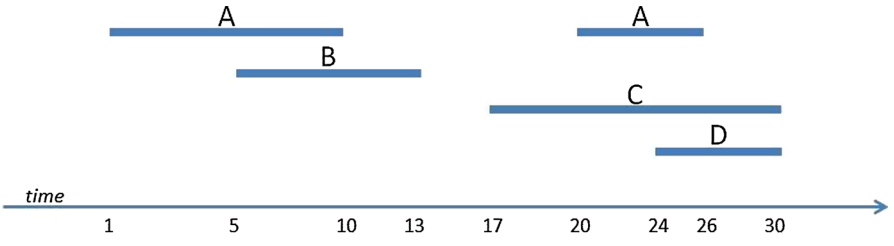

Fig. 1 Example of a sequence of 5 temporal intervals

terms of predictive performance against traditional classifiers on real datasets from the domains of diabetes, intensive care, and infectious hepatitis. Similar temporal abstractions have been employed for the exploration and classification of renal-damage risk factors in patients with diabetes type II (Klimov et al. 2015). Furthermore, consider the domain of Sign Language (Papapetrou et al. 2009b), where sequences of temporal intervals can be formed by events corresponding to facial gestures, e.g., "raised eyebrows", syntactic notations, e.g., "wh-question", or part-of-speech identifications of manual signs, e.g., "wh-word". Since events may occur concurrently, pairwise temporal relations can emerge. Similarity search in this domain is critical for supporting (deaf) people who use Sign Language and for creating relevant assistive applications. Subsequence search has proven useful for detecting specific language expresssions (Kostakis and Papapetrou 2015).

Several distance functions for computing the dissimilarity of pairs of instances have been proposed for most common data structures; Dynamic Time Warping (DTW) (Itakura 1975) and Euclidean distance (Faloutsos et al. 1994) for time series, variants of graph matching for graphs (Umeyama 1988; Bunke 2000), etc. Our work attempts to address similar needs in the field of sequences of temporal intervals.

Converting event-interval sequences to symbolic sequences (strings) and applying existing similarity measures is not an applicable approach; especially when temporal relations such as *overlaps* or *contains* occur between event-intervals of the *same* label. This has also has been demonstrated and argued extensively in several existing works, e.g., by Kostakis et al. (2011a) and Papapetrou et al. (2009b). In such setting, converting these sequences to a symbolic sequence representation may result in loss of temporal information.

In the simplest example, one could convert a sequence of event-intervals to a sequence of instantaneous events by only considering the start and end points of each event-interval, and associating each of the two points with the event label that corresponds to that event-interval. This would result in a simplification of the representation of these sequences; they would be mapped to traditional sequences of instantaneous events without the need for keeping track of the pair-wise interval relations. Nonetheless, the size of the alphabet (i.e., set of all possible event labels) would double since each interval label would be mapped to two labels, each corresponding to an instantaneous event. Then, the solution to the problem would reduce to applying an existing distance or similarity measure for sequence matching, such as the Levenshtein distance (Levenshtein 1966). The aforementioned solution, although simple, is not correct, as it may lead to a large number of false matches and can produce arbitrarily bad scores irrespective of the similarity measure used.

To illustrate why a mapping to symbolic sequences and the application of a string-matching algorithm would introduce ambiguities when event-intervals of the same label are present, consider the two examples shown in Fig. 2. In these examples, each event-interval sequence consists of three intervals with the same label. In the first case (Fig. 2a), each event-interval is fully contained within the other (in terms of duration), while in the second case (Fig. 2b) each event-interval overlaps with all the previous. Obviously, the mapping for both sequences is the same, i.e., $\{A_s, A_s, A_s, A_e, A_e, A_e\}$. This suggests that traditional methods for discrete event sequences may fail to capture

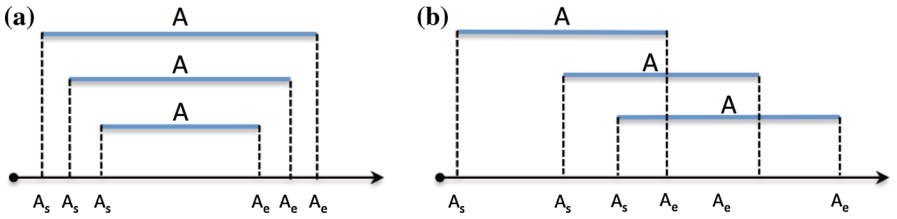

Fig. 2 An example where mapping event-interval sequences to discrete event sequences may cause ambiguities

the inherent temporal structure of such sequences, and more importantly, they may produce arbitrarily bad results especially as the number of event labels increases.

Other approaches for mapping event-interval sequences to strings augment the original alphabet with symbols, such as brackets, plus and minus signs, or numbers corresponding to counts of interval-relations [(we refer the readers to the discussion by Patel et al. (2008)]. Again, if we would apply string algorithms out-of-the-box to such strings the results would not be valid. In most cases, the matchings would not even be valid in the intervals domain (brackets would remain open, intervals would begin but not end, etc.).

In this paper we study the problem of efficient similarity search under full-sequence matching in databases containing sequences of event-intervals. We first present `Artemis`, a robust distance metric for sequences of event-intervals. `Artemis` has appeared in our preliminary work (Kostakis et al. 2011a), while in this work we extend `Artemis` by providing additional theoretical properties and bounds. Our main focus in this paper is, however, to enable practical and industrial-scale applications of similarity search under `Artemis`. It is, hence, imperative to be able to quickly and efficiently search through very large datasets of temporal interval sequences. This can be achieved by devising efficient indexing schemes that can provide significant speedups against brute-for-search without sacrificing nearest-neighbor retrieval accuracy.[1] A suitable indexing scheme should also take into consideration the underlying properties and peculiarities of the studied data types. Thus, we propose `EBESM` (shorthand for Embedding-based E-sequence Matching), an approximate embedding-based indexing scheme. This is the first indexing scheme for sequences of event-intervals under a distance function; recently several methods such as *Relation Index* (Kostakis and Gionis 2015) and *KarmaLego* (Moskovitch and Shahar 2015) have been proposed for speeding up the task of subsequence search.

Hence, the main **contributions** of this line of work are summarized as follows:

- We formulate the problem of assessing the dissimilarity of sequences of event-intervals and present a method to solve it. The method, called `Artemis` (Kostakis et al. 2011a), takes into account the temporal relations that may occur between the events in the sequence and employs bipartite maximal matching to compute their

[1] For the remainder of this paper we will refer to nearest-neighbor retrieval accuracy (Papapetrou et al. 2011) as "retrieval accuracy". For clarity, we note that in this paper this term is used within the context of nearest neighbor similarity search and not in the context of information retrieval. A formal definition is provided in Sect. 6.

distance. We extend our previous work (Kostakis et al. 2011a) by proving that `Artemis` is a metric under our problem setting and propose a linear-time lower bound for `Artemis` that achieves significant speedups during similarity search.
- The performance of `Artemis` is tested and benchmarked through rigorous experimentation on eight real datasets, including American Sign Language annotated video recordings, robot sensor data, and Hepatitis patient data.
- We further propose an approximate embedding-based indexing method, called `EBESM` for approximate similarity search under `Artemis`, and prove that `EBESM` is an embedding that satisfies the contractiveness property.
- The performance of `EBESM` is evaluated on a large synthetic dataset, where it is shown that speedups of over an order of magnitude can be achieved. In addition, we confirm empirically that vector-based indexing methods are inefficient for searching in databases of event-interval sequences.

2 Problem setting

Let $\Sigma = \{E_1, \ldots, E_m\}$ be an alphabet of m event labels. An event that occurs over a time interval defines an *event-interval* and an ordered multiset of event-intervals defines an *event-interval sequence*. Next, we provide a more formal definition for these two concepts.

Definition 1 *(event-interval)* An *event-interval* is defined as a triple $S = (E, t_{start}, t_{end})$, where $S.E \in \Sigma$ and $S.t_{start}$, $S.t_{end}$ correspond to the start and end time of S, respectively. In general, $S.t_{start} \leq S.t_{end}$, where the equality holds when the event is instantaneous.

Definition 2 *(e-sequence)* A sequence of event-intervals, or *event-interval sequence*, or *e-sequence*, $\mathcal{S} = \{S_1, \ldots, S_n\}$ is an ordered multiset of n event-intervals. The temporal order of the event-intervals in $\mathcal{S}$ is ascending based on their start time and in the case of ties it is descending based on their end time. If ties still exist, the event-intervals are sorted alphabetically.

The *length* of an e-sequence $\mathcal{S}$ is defined as the number of event-intervals in the e-sequence (denoted as $|\mathcal{S}|$), while its *size* is the number of temporal relations in $\mathcal{S}$.

For example, the event-interval sequence shown in Fig. 1 has length $|\mathcal{S}| = 5$, and is encoded as:

$$\mathcal{S} = \{(A, 1, 10), (B, 5, 13), (C, 17, 30), (A, 20, 26), (D, 24, 30)\}.$$

It becomes apparent that in an e-sequence there exist temporal relations between the event-intervals. We base our work on Allen's model for event-interval relations (Allen 1983). Allen's algebra defined 13 relations, that form 6 pairs of inverse relations, and the remaining is the 'equal' relation. We keep only one relation out of every pair. Hence, given two event-intervals A and B, we consider the following seven relations (shown in Fig. 3): *before*(A,B), *meets*(A,B), *equal*(A,B), *overlaps*(A,B), *contains*(A,B), *startedby*(A,B), *finishedby*(A,B). For the remainder of this paper, we denote as $R(A, B)$ the

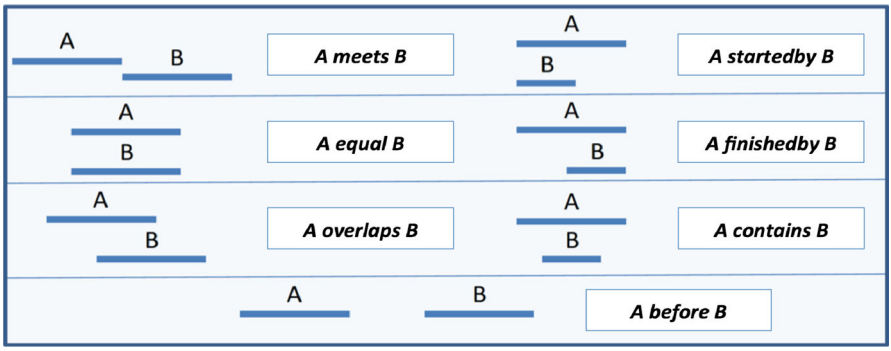

Fig. 3 The seven temporal relations between two event-intervals that are considered in this paper

temporal relation between A and B. More details about these relations can be found in Papapetrou et al. (2009b). We define the set of valid temporal relations between two event-intervals as $\mathcal{I} = \{meets, equal, overlaps, contains, started\text{-}by, finished\text{-}by, before\}$, with $|\mathcal{I}| = 7$.

Definition 3 *(labeled relation)* Given two event-intervals S and T, a labeled relation $LR(S, T)$ denotes the combination of both the temporal relation $R(S, T)$ between S and T and the ordered pair of their labels $(S.E, T.E)$; for example, $follow(E_2, E_1)$, with $E_1, E_2 \in \Sigma$. For a pair of labeled relations $LR(S, T)$ and $LR(U, V)$, it holds that $LR(S, T) = LR(U, V)$ if and only if $R(S, T) = R(U, V)$, $S.E = U.E$, and $T.E = V.E$.

In this paper, we study two problems:

Problem 1 Define a distance measure D for e-sequences, that satisfies the following two properties:

- *Metric property* D should be a metric. That is: (i) $D(\mathcal{S}, \mathcal{T}) \geq 0$, $\forall \mathcal{S}, \mathcal{T}$, (ii) $D(\mathcal{S}, \mathcal{T}) = 0$, iff $\mathcal{S} = \mathcal{T}$, $\forall \mathcal{S}, \mathcal{T}$, (iii) $D(\mathcal{S}, \mathcal{T}) = D(\mathcal{T}, \mathcal{S})$, $\forall \mathcal{S}, \mathcal{T}$, and (iv) $D(\mathcal{S}_1, \mathcal{S}_3) \leq D(\mathcal{S}_1, \mathcal{S}_2) + D(\mathcal{S}_2, \mathcal{S}_3)$, $\forall \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$, where $\mathcal{S}, \mathcal{T}, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ are e-sequences.
- *Temporal relations* D should capture and reflect combinations of event-intervals and their temporal relations. Hence, the distance between two e-sequences should depend on the number of event-interval relations in which they agree, or disagree.

Problem 2 Devise an indexing method for efficient e-sequence similarity search under distance measure D. More particularly, we are interested in developing an index structure that will exploit the metric property of a given measure D, and hence achieve search speedups of at least an order of magnitude compared to brute-force search.

We explain the rationale behind the two properties of Problem 1. First, being metric is a highly favorable property for any distance measure, since this can facilitate several data mining tasks efficiently, such as clustering, or indexing (Hjaltason and Samet 2003). In addition, it can be used to provide exact indexing with theoretical guarantees on the trade-offs between nearest-neighbor retrieval accuracy and efficiency

(see for example RBSA(Papapetrou et al. 2009a)) as opposed to accurate though only empirically efficient speedup techniques (Keogh 2002; Papapetrou et al. 2011). Secondly, taking into account the types of temporal relations is in line with previous work (Kostakis et al. 2011a; Papapetrou et al. 2009b; Patel et al. 2008) and based on the motivating example applications and discussion presented in Sect. 1.

Finally, in our model, we are not concerned with the actual duration of the event-intervals nor with the time separating any pair of event-intervals. This makes our measure robust to any time warping. So, the two sequences $\{(A, 1, 2), (B, 3, 4)\}$ and $\{(A, 10, 20), (B, 50, 100)\}$ are considered the same. The motivation and rationale for this is that we should expect from people who practice Sign Language, when repeating a phrase, to consume different amounts of time for each word and the whole utterance between different attempts; similarly when pronouncing a long sentence in spoken language. In other words, the semantic information is contained in the relations among the event-intervals. In the same manner, we assume that in robot sensor data a high-level description of a situation is derived from the combination of underlying events. For example, the gripping mechanism was enabled throughout the whole time the robot's wheels were active so the object was transferred successfully to the target location, in contrast to releasing at any time point half-way through which would indicate a drop.

2.1 Prior distance functions for e-sequences

In this Section we provide a brief description of the existing methods for comparing e-sequences. None of the existing methods solve Problem 1; they all fail to satisfy the required property: $D(\mathcal{S}, \mathcal{T}) = 0 \iff \mathcal{S} = \mathcal{T}$.

Relation Matrix (Kostakis et al. 2011b) For each of the two e-sequences given as input, `Matrix` enumerates their $\binom{|S|}{2}$ labelled relations. The counts of each labelled relation are stored in a matrix with dimensions $7 \times |\Sigma|^2$; note that $|\mathcal{I}| = 7$. Rows correspond to relation types (i.e. *equal*, *overlaps*, *before*), while columns correspond to ordered pairs of event-interval labels. Each matrix cell contains the count of each labeled relation in a given e-sequence. Comparing e-sequences reduces to comparing the matrices. The distance of two matrices is the $\mathcal{L}_1$ norm over all cells, normalized by the sum of the e-sequences' sizes; $d(S, T) = \sum_i^{|\mathcal{I}|} \sum_j^{|\Sigma|^2} \frac{|M_S(i,j) - M_T(i,j)|}{M_S(i,j) + M_T(i,j)}$, where M_S and M_T are the matrices of e-sequences S and T respectively. Populating the matrices requires $O(n^2 + m^2)$ time, where $n = |S|$ and $m = |T|$, for enumerating the relations. Comparing the two matrices requires $O(|\Sigma|^2)$ time; hence inducing a total complexity of $O(n^2 + m^2 + |\Sigma|^2)$.

This is a straightforward measure that simply compares the sets of all event-interval relations between two e-sequences. Nonetheless, this approach is not at all sufficient, since it fails to solve Problem 1. In particular, there exist infinitely many pairs of e-sequences that are different but `Matrix` returns zero distance score. Figure 4 depicts such examples of movements in a network of binary proximity sensors; in each case `Matrix` is unable to distinguish between the scenarios of an object following different paths; the corresponding e-sequences are depicted in Fig. 5 and the instance of `Relation Matrix` in Table 1.

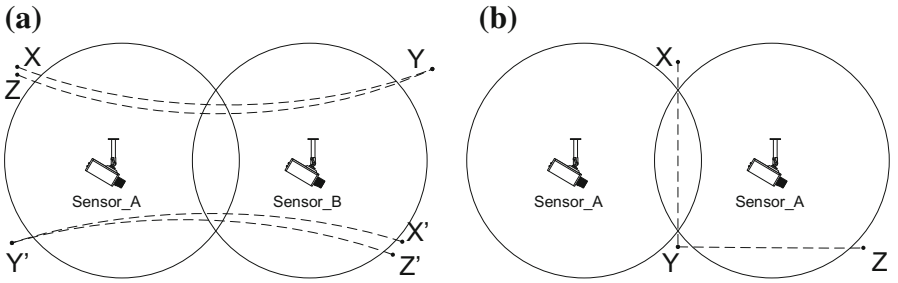

Fig. 4 Two examples for which `Matrix` is unable to distinguish between scenarios of an object following different paths; **a** XYZ versus X′Y′Z′ (different label sensors), **b** XYZ versus ZYX (same label sensors)

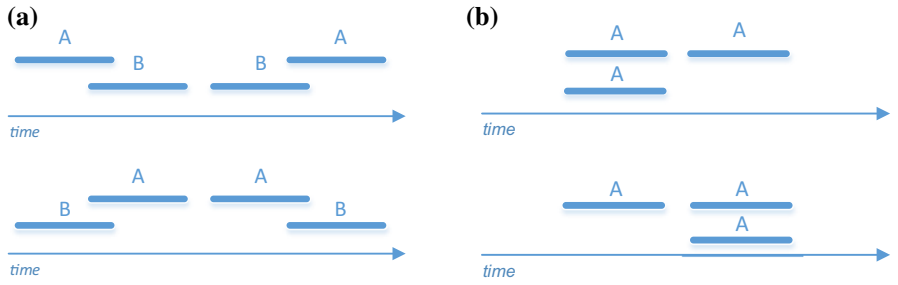

Fig. 5 The corresponding e-sequences for the scenarios of Fig. 4. `Matrix` is unable to identify the differences between each pair, and returns a zero distance score. There is an infinite number of e-sequence pairs that `Matrix` is unable to distinguish. **a** Different label sensors, **b** same label sensors

Table 1 The `Relation Matrix` instance for the e-sequences in Fig. 5a

	(A,A)	(A,B)	(B,A)	(B,B)
Before	1	1	1	1
Overlap	0	1	1	0
Equal	0	0	0	0
Overlaps	0	0	0	0
Started-by	0	0	0	0
Finished-by	0	0	0	0
Contains	0	0	0	0

Vector-DTW (Kostakis et al. 2011a) As discussed already, it is impossible to map e-sequences to strings of symbols without introducing ambiguity. For example, how should we order the symbols to differentiate the relative position of the end-points of two event-intervals, when their relation is 'contains', 'equal', or 'overlaps'? In other words, the ambiguity is created when we attempt to define an order for symbolic events that happen concurrently (e.g. both event-intervals begin at the same time). No matter the ordering we decide, there will exist another event-interval relation that cannot be represented correctly. A work-around approach is to map e-sequences to vectors based on the number of event-intervals that are active at certain time points. `Vector-DTW`, or simply `DTW`, creates a vector for each start- or end-point of any

$$\left(\begin{pmatrix}1\\0\\0\\0\end{pmatrix},\begin{pmatrix}1\\1\\0\\0\end{pmatrix},\begin{pmatrix}0\\1\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\end{pmatrix},\begin{pmatrix}0\\0\\1\\0\end{pmatrix},\begin{pmatrix}1\\0\\1\\0\end{pmatrix},\begin{pmatrix}1\\0\\1\\1\end{pmatrix},\begin{pmatrix}0\\0\\1\\1\end{pmatrix},\begin{pmatrix}0\\0\\0\\0\end{pmatrix}\right)$$

Fig. 6 The sequence of vectors created by `Vector-DTW` for the e-sequence in Fig. 1; the coordinates correspond to labels A, B, C, and D respectively. `IBSM` would create similar vectors for each time-point of the sequence; 30 vectors

event-interval. For the example in Fig. 1, it would create 9 vectors, one for each of the indicated time-points on the image; these are depicted in Fig. 6.

Then, the problem of comparing e-sequences reduces to comparing the sequences of vectors. The sequences of vectors are compared using the multi-variate extension of the Dynamic Time Warping distance function; requiring $O(m \cdot n \cdot |\Sigma|)$ time, where $n = |S|$ and $m = |T|$. While the use of vectors enables to avoid certain types of ambiguity, it still suffers in cases like the one in Fig. 2.

IBSM (Kotsifakos et al. 2013) The rationale is similar to that of `Vector-DTW`, but instead of creating a vector each time an event begins or ends, in IBSM a vector is created for each time point. Then, all the vectors are concatenated to create a matrix. Similar to `Matrix`, the `IBSM` distance is computed over the matrices. $IBSM(S,T) = \sqrt{\sum_t \sum_i^{|\Sigma|} (V_S(t,i) - V_T(t,i))^2}$, where $V_S(t,i)$ and $V_T(t,i)$ denote the number of active events with label $j \in \Sigma$ at time point t for e-esquences S and T respectively. This requires e-sequences to have the same time-duration. To account for e-sequences that do not have the same time-duration, all e-sequences are scaled using linear interpolation to match the duration of the longest e-sequence in the dataset. The running time of IBSM is $O(|\Sigma| \cdot \gamma)$, where γ is the maximum time duration of an e-sequence in the whole dataset.

`IBSM`, like `DTW`, does not consider the relations between event-intervals. As a result, it cannot differentiate between the two e-sequences of Fig. 2. This is the only method in this paper where the final result is affected by the "sampling rate"; all else remaining the same, if we simply have more time-points due to increased time granularity, the `IBSM` distance of two e-sequences will increase linearly with respect to the increase in the number of time-points. Furthermore, due to the temporal scaling of the e-sequences to match the maximum duration of an e-sequence γ in the whole dataset, this results in various inconsistencies: the distance of two e-sequences, under `IBSM`, changes as a function of γ, even if none of the two e-sequences' duration is equal to γ.

3 A bipartite matching-based approach

We present a distance measure, called `Artemis` (Kostakis et al. 2011a), for comparing e-sequences, as well as a method to compute it. We theoretically (in this Section) and experimentally (in Sect. 6) demonstrate the superiority of `Artemis` against the three existing state-of-the-art measures. In summary, `Artemis` is superior with respect to the following three aspects:

- `Artemis` **is a metric** One major shortcoming of these three competitors is that they are not metric. More importantly they all violate the identity of indiscernibles (or Leibniz's Law). This suggests that sequences that highly differ in terms of temporal relations may be assigned with a distance of 0. Hence, competitors are not guaranteed to always provide the correct result as opposed to `Artemis`. Furthermore, as described by Fig. 2, `DTW` and `IBSM` may provide arbitrarily bad results; there is no bound on the number of relations that may differ, between two sequences, while the two methods identify them as identical. Similarly, `Matrix` is unable to distinguish between pairs of sequences that may contain even very few event-intervals. It becomes, hence, a necessity when using `DTW`, `Matrix`, and `IBSM` even under linear scan, to post-filter the results for removing false matches.
- `Artemis` **is robust to noise** In modern database systems, noise-tolerant retrieval of structured data is a necessity; the queries are usually noisy versions of the objects contained in the database. Hence, our retrieval method should tolerate noise. This is clearly demonstrated in the noise robustness experiments (Sect. 6.2.2), where `Artemis` outperforms all the other methods.
- **The run-time of** `Artemis` **depends only on the size of the e-sequences** `Artemis` is the only method whose run-time complexity is a function only of the size of the two e-sequences and does not depend on the size of the alphabet of the whole dataset or other factors. The other methods become slow when the alphabet size increases. Our experimental evaluation shows that `Artemis` is faster than the competitors in 3 out of 8 real datasets and better than `Matrix` and `IBSM` in 5 out of 8 datasets. Nonetheless, when applying `EBESM` and scaling the database size in a synthetic experiment (Sect. 6.3), speedups of over an order of magnitude can be achieved against brute-force serial scan using `Artemis`.

3.1 Defining Artemis

Given two e-sequences $\mathcal{S}$ and $\mathcal{T}$, the requirement for `Artemis` is to define their distance by taking into account the temporal relations among the event-intervals while satisfying the metric property. The overall similarity of two e-sequences is inferred by determining an equivalence between pairs of event-intervals. The equivalence of a pair of event-intervals belonging to different e-sequences is determined by the fraction of common event-interval relations they are involved in. We use this to quantify the distance between event-intervals. The goal, then, is to find the matching that yields the minimum distance over all paired intervals. Thus, the objective of `Artemis` reduces to finding the matching that induces the minimum sum of scores, which is the optimal perfect matching. That score is the `Artemis` distance of the two e-sequences.

3.2 Computing Artemis

The computation of `Artemis` can be broken down to two steps: (a) the mapping step and (b) the matching step. The procedure for calculating the Artemis distance of two e-sequences is described in Algorithm 1.

The mapping step The first step is to map each e-sequence $\mathcal{S}$ to a sequence of multi-sets of temporal relations between event-intervals. More specifically, for each event-interval $S_i \in \mathcal{S}$ we record the set of labeled relations of S_i with $S_j \in \mathcal{S}, \forall j \neq i$ in the same e-sequence. Three multi-sets of labeled relations are computed as follows:

- $\mathcal{R}_l(S_i) = \{LR(S_j, S_i) | 1 \leq j < i\}$, which contains the labeled temporal relations of S_i with all event-intervals preceding S_i in $\mathcal{S}$.
- $\mathcal{R}_r(S_i) = \{LR(S_i, S_j) | i < j \leq |\mathcal{S}|\}$, which contains the labeled temporal relations of S_i with all event-intervals succeeding S_i in $\mathcal{S}$, and
- $\mathcal{R}_\varnothing(S_i) = \{LR(\varnothing, S_i)\}$, which is a singleton with a labeled *follow* relation between S_i and $\varnothing$—an extra symbol such that $\varnothing \notin \Sigma$.

We additionally denote: $\mathcal{R}_{\varnothing l}(S_i) = \mathcal{R}_l(S_i) \cup \mathcal{R}_\varnothing(S_i)$.

Definition 4 *(event-interval relation set)* Given an e-sequence $\mathcal{S}$, for each event-interval $S_i \in \mathcal{S}$, the event-interval relation set of S_i is defined as follows:

$$\mathcal{R}(S_i) = \mathcal{R}_l(S_i) \cup \mathcal{R}_r(S_i) \cup \mathcal{R}_\varnothing(S_i). \tag{1}$$

Note that $\varnothing$ is introduced so that event-interval labels are also taken into account. Specifically, e-sequences that differ in event-interval relations but share similar event labels will be assigned with smaller distance values than e-sequences that differ in both event labels and event-interval relations.

The matching step Given two e-sequences $\mathcal{S}$ and $\mathcal{T}$, the matching step of `Artemis` computes a distance value $d_m(S_i, T_j)$ for each pair of event-intervals $S_i \in \mathcal{S}$ and $T_j \in \mathcal{T}$ as defined in Eq. 2.

$$d_m(S_i, T_j) = \begin{cases} \dfrac{\max\{|\mathcal{S}|, |\mathcal{T}|\} - |\mathcal{R}_{\varnothing l}(S_i) \cap \mathcal{R}_{\varnothing l}(T_j)| - |\mathcal{R}_r(S_i) \cap \mathcal{R}_r(T_j)|}{\max\{|\mathcal{S}|, |\mathcal{T}|\}}, & \text{if } S_i.E = T_j.E \\ 1, & \text{if } S_i.E \neq T_j.E \end{cases} \tag{2}$$

If in Eq. 2 we would subtract $|\mathcal{R}(S_i) \cap \mathcal{R}(T_j)|$ instead of the two intersections, `Artemis` would violate the metric property in some instances; such a case are the pair of Fig. 5b.

To handle the case of e-sequences of different size, "dummy" event-intervals, with distance 1 from all other event-intervals, are added to the smaller e-sequence. We denote the potentially augmented e-sequences by $\mathcal{S}'$ and $\mathcal{T}'$. In addition, $|\mathcal{S}'| = |\mathcal{T}'| = \max\{|\mathcal{S}|, |\mathcal{T}|\}$.

Let $D_{\mathcal{S}',\mathcal{T}'}$ be a $|\mathcal{S}'| \times |\mathcal{T}'|$ matrix, with $D_{\mathcal{S}',\mathcal{T}'}(i, j) = d_m(S'_i, T'_j)$, $S'_i \in \mathcal{S}'$ and $T'_j \in \mathcal{T}'$. We call $D_{\mathcal{S}',\mathcal{T}'}$ the *event-interval distance matrix* of $\mathcal{S}'$ and $\mathcal{T}'$. Given $D_{\mathcal{S}',\mathcal{T}'}$, `Artemis` reduces to computing the minimum score induced over all perfect matchings, i.e., matchings that assign each and every event-interval in $\mathcal{S}'$ to exactly one event-interval in $\mathcal{T}'$.

The optimal perfect matching can be found by the Hungarian algorithm (Munkres 1957); this is denoted by $MinCostMaxBipartiteMatching(DistM)$ in Algorithm 1. The Hungarian algorithm operates on complete bipartite graphs with weighted edges; in our case the vertices correspond to intervals in each e-sequence and the weights on the edges are their interval-distances $d_m(S'_i, T'_j)$. The algorithm returns a

Algorithm 1 Computing `Artemis`

Input: Two Event-Interval Sequences $\mathcal{S}, \mathcal{T}$
Output: The distance score: $Artemis(\mathcal{S}, \mathcal{T})$
for all intervals $S_i \in \mathcal{S}$ **do**
compute $R_{\varnothing}(S_i), \mathcal{R}_l(S_i), \mathcal{R}_r(S_i)$
end for
for all intervals $T_i \in \mathcal{T}$ **do**
compute $R_{\varnothing}(T_i), \mathcal{R}_l(T_i), \mathcal{R}_r(T_i)$
end for
$\mathcal{S}', \mathcal{T}' =$ Augment($\mathcal{S}$,$\mathcal{T}$)
// Compute all event-interval distances $d_m(S'_i, T'_j)$
for all $S'_i \in \mathcal{S}'$ **do**
for all $T'_j \in \mathcal{T}'$ **do**
//update distance matrix
DistM$(i, j) = d_m(S'_i, T'_j)$
end for
end for
$\mathcal{H}(\mathcal{S}', \mathcal{T}')$ = MinCostMaxBipartiteMatching(DistM)
return $\sum_{i=1}^{|\mathcal{S}'|} d_m(S'_i, h(S'_i))$

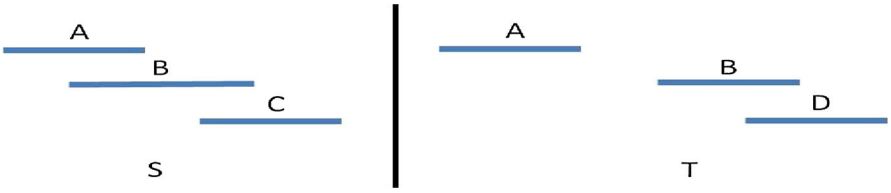

Fig. 7 Two e-sequences (time points are omitted) $\mathcal{S}$ and $\mathcal{T}$ used as an example for `Artemis`

set of edges so that each vertex is uniquely connected to another vertex, and the sum of the edges' weight is minimised.

The Hungarian algorithm takes as input the event-interval distance matrix $D_{\mathcal{S}',\mathcal{T}'}$. Let the output of the algorithm be the following matching $\mathcal{H}(\mathcal{S}', \mathcal{T}') = (h(S'_1), \ldots, h(S'_{|\mathcal{S}'|}))$ with an assignment cost $C(\mathcal{S}', \mathcal{T}')$. By $h(S'_i) \in \mathcal{H}(\mathcal{S}', \mathcal{T}')$ we simply denote the event-interval in $\mathcal{T}'$ that $S'_i \in \mathcal{S}'$ is matched to by the Hungarian algorithm.

Hence, the assignment cost $C(\mathcal{S}', \mathcal{T}')$ corresponds to the *Artemis distance* between the original e-sequences $\mathcal{S}$ and $\mathcal{T}$. Based on the above, given $\mathcal{S}'$, $\mathcal{T}'$, and $\mathcal{H}(\mathcal{S}', \mathcal{T}')$, the *Artemis distance* of $\mathcal{S}$ and $\mathcal{T}$ is computed as follows:

$$Artemis(\mathcal{S}, \mathcal{T}) = \sum_{i=1}^{|\mathcal{S}'|} d_m(S'_i, h(S'_i)). \tag{3}$$

Example Fig. 7 shows two e-sequences $\mathcal{S}$ and $\mathcal{T}$. The *mapping step* first computes the event-interval relation multi-sets. For $\mathcal{S}$, these are:
$\mathcal{R}_{\varnothing l}(S_1) = \{before(\varnothing, A)\}$ and $\mathcal{R}_r(S_1) = \{overlaps(A, B), before(A, C)\}$
$\mathcal{R}_{\varnothing l}(S_2) = \{before(\varnothing, B), overlaps(A, B)\}$, and $\mathcal{R}_r(S_2) = \{overlaps(B, C)\}$
$\mathcal{R}_{\varnothing l}(S_3) = \{before(\varnothing, C), before(A, C), overlap(B, C)\}$, and $\mathcal{R}_r(S_3) = \emptyset$.

For $\mathcal{T}$, these sets are:
$\mathcal{R}_{\varnothing l}(T_1) = \{before(\varnothing, A)\}$ and $\mathcal{R}_r(T_1) = \{before(A, B), before(A, D)\}$
$\mathcal{R}_{\varnothing l}(T_2) = \{before(\varnothing, B), before(A, B)\}$, and $\mathcal{R}_r(T_2) = \{overlaps(B, D)\}$
$\mathcal{R}_{\varnothing l}(T_3) = \{before(\varnothing, D), before(A, D), overlaps(B, D)\}$, and $\mathcal{R}_r(T_3) = \emptyset$.
At the *matching step* the Hungarian algorithm would return $\mathcal{H}(\mathcal{S}, \mathcal{T}) = (\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3)$, hence the pairs, based on their labels, are (A, A), (B, B), and (C, D). Finally, $Artemis(\mathcal{S}, \mathcal{T}) = (2/3 + 2/3 + 1) = 7/3$.

If $\mathcal{S}$ contained an additional event, then $\mathcal{R}_{\varnothing l}(S_i)$ and $\mathcal{R}_r(S_i), \forall i$, would be appropriatelly augmented, while $\mathcal{R}_{\varnothing l}(T_j)$ and $\mathcal{R}_r(T_j), \forall j$, would remain the same. A dummy interval would be added in $\mathcal{T}$ for the purpose of the Hungarian algorithm.

Complexity Let $m = max(|\mathcal{S}|, |\mathcal{T}|)$. Then, at the mapping step, $O(m^2)$ relations are enumerated, while the complexity of computing $D(\mathcal{S}, \mathcal{T})$ using hashing is $O(m^3)$. The cost of applying the Hungarian algorithm to the two event-interval relation sets results in a total time complexity of $O(m^3)$.

3.3 Lower bounding Artemis

We propose a lower bound to speed up similarity search using `Artemis`. The lower bound can be computed in linear time and is based on the comparison of event label counts. By knowing the number of labels in which two e-sequences differ, we can determine a lower bound for their `Artemis` distance.

Given an e-sequence $\mathcal{S}$, we define a $|\Sigma|$-dimensional vector $v^{\mathcal{S}}$, that stores, for each event label in Σ, the count of event-intervals in $\mathcal{S}$ that share that label.

Theorem 1 *Given $\mathcal{S}$,$\mathcal{T}$,a lower bound for $Artemis(\mathcal{S}, \mathcal{T})$ is given by:*

$$Artemis_{LB}(\mathcal{S}, \mathcal{T}) = \frac{k}{2} + \left(m - \frac{k}{2}\right)\left(\frac{k}{2m}\right) = k - \frac{k^2}{4m}, \tag{4}$$

where $k = ||v^{\mathcal{S}} - v^{\mathcal{T}}||_1$ and $m = max(|\mathcal{S}|, |\mathcal{T}|)$.

Proof Under `Artemis`, when an interval is matched to another interval with a different label, or to a dummy interval when $|\mathcal{S}| \neq |\mathcal{T}|$, it induces a partial score of 1. Knowing that $||v^{\mathcal{S}} - v^{\mathcal{T}}||_1 = k$ we can be sure that $Artemis(\mathcal{S}, \mathcal{T}) \geq k/2$; in the worst case, those k event-intervals are divided into $k/2$ in each e-sequence, and matched with each other contributing a score of $d_m(S_i, T_j) = 1$ for each of the $k/2$ pairs.

If $|\mathcal{S}| = |\mathcal{T}| = \frac{k}{2}$, then $Artemis(\mathcal{S}, \mathcal{T}) = k/2$. If not, then the fact that $||v^{\mathcal{S}} - v^{\mathcal{T}}||_1 = k$ is reflected in the matching scores of the rest of the event-intervals (due to differences in $\mathcal{R}_l(S_i)$, $\mathcal{R}_l(T_j)$, $\mathcal{R}_r(S_i)$, and $\mathcal{R}_r(T_j)$). So, given $m = max(|\mathcal{S}|, |\mathcal{T}|)$, with $m > \frac{k}{2}$, the rest of the $m - k/2$ event-intervals of each e-sequence would have at least $k/2$ non-common relations with their matched counterparts. Thus, increasing $Artemis(\mathcal{S}, \mathcal{T})$ by an additional $(m - k/2) \cdot (k/2m)$. □

The proposed lower bound focuses on label counts and not on relations of event-intervals. When the differences of two e-sequences are restricted to event-interval

labels, the lower bound is equal to the distance obtained by *Artemis*. On the other hand, when the e-sequences share the same event labels and differ only in the type of event-interval relations, then the lower bound is inefficient and yields zero score. The tightness and pruning power of the lower bound is studied on eight datasets in Sect. 6.

3.4 Metric property

In this section we show that `Artemis` is a metric. Specifically, we prove that it satisfies the triangle inequality.

Theorem 2 `Artemis` *satisfies the triangle inequality.*

Proof For any three e-sequences $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ we will show that

$$Artemis(\mathcal{A}, \mathcal{C}) \leq Artemis(\mathcal{A}, \mathcal{B}) + Artemis(\mathcal{B}, \mathcal{C}).$$

We prove this by showing a stronger statement. Consider the following three matchings: $h_1 = H(\mathcal{A}, \mathcal{B})$,with $h_1(A_i) = B_j$, $A_i \in \mathcal{A}$, $B_j \in \mathcal{B}$ and $h_2 = H(\mathcal{B}, \mathcal{C})$, with $h_2(B_j) = C_k$, with $B_j \in \mathcal{B}$, $C_k \in \mathcal{C}$, given by the Hungarian Algorithm for the two pairs of e-sequences, we construct a matching h_3 so that $h_3 = h_2 \circ h_1$. In other words, if $A_i \in \mathcal{A}$ is matched to $B_j \in \mathcal{B}$ by h_1, and B_j is matched to $C_k \in \mathcal{C}$ by h_2, then h_3 matches A_i to C_k.

We prove that even under these matchings, the cost induced by h_3 is always less than or equal to the sum of the costs induced by h_1 and h_2. Since h_3 induces a score greater or equal to that of the matching returned by the Hungarian algorithm the triangle inequality holds for Artemis.

We can formulate $Artemis(\mathcal{A}, \mathcal{B})$ as

$$\sum_{i=1}^{|\mathcal{B}'|} d_m(A_i, h(A_i)) = \sum_i \left(1 - \frac{(p_{ij} + q_{ij})}{\max\{|\mathcal{A}|, |\mathcal{B}|\}}\right),$$

with $p_{ij} = |\mathcal{R}_{\varnothing l}(A_i) \cap \mathcal{R}_{\varnothing l}(B_j)|$ and $q_{ij} = |\mathcal{R}_r(A_i) \cap \mathcal{R}_r(B_j)|$.

Our proof is based on showing that the inequality holds on an interval-pair level ($d_m(A_i, C_k) \leq d_m(A_i, B_j) + d_m(B_j, C_k)$) for all triples A_i, B_j, C_k where $h_1(A_i) = B_j$ and $h_2(B_j) = C_k$. Thus, the inequality will also hold under the summation. We have two cases:

Case I Not all three event labels are the same: In this case the triangle inequality holds, trivially. For example, if $A_i.E = B_j.E$ and $A_i.E \neq C_k.E$, then we have

$$d_m(A_i, C_k) = 1 \leq d_m(A_i, B_j) + 1 = d_m(A_i, B_j) + d_m(B_j, C_k). \tag{5}$$

Case II All three event labels are the same: In this case we first have to prove that the inequality holds when $|\mathcal{A}| = |\mathcal{B}| = |\mathcal{C}|$.

Suppose that the triangle inequality does not hold for a triple of matched intervals. Then, we have:

$$\begin{aligned} & d_m(A_i, B_j) + d_m(B_j, C_k) < d_m(A_i, C_k) \\ & \Rightarrow 1 - (p_{ij} + q_{ij})/|\mathcal{A}| + 1 - (p_{jk} + q_{jk})/|\mathcal{A}| < 1 - (p_{ik} + q_{ik})/|\mathcal{A}| \\ & \Rightarrow |\mathcal{A}| + (p_{ik} + q_{ik}) < (p_{ij} + q_{ij}) + (p_{jk} + q_{jk}) \end{aligned} \tag{6}$$

Since all the three intervals have the same labels, $p_{ik}, p_{ij}, p_{jk} \geq 1$ due to the $\mathcal{R}_\varnothing$ relations. The minimum possible value for $p_{ik} + q_{ik}$ is 1, and in that case the maximum possible value of $(p_{ij} + q_{ij}) + (p_{jk} + q_{jk})$ is $|\mathcal{A}| + 1$, i.e. the $|\mathcal{A}| - 1$ common labeled relations excluding the two $\mathcal{R}_\varnothing$, can be divided in all possible combinations between $(p_{ij} + q_{ij})$ and $(p_{jk} + q_{jk})$ so that none appears in both of the sets. This last part is important since if a labeled relation appears in both $(p_{ij} + q_{ij})$ and $(p_{jk} + q_{jk})$, that would imply that it also appears between A_i and C_k, so it would be $p_{ik} + q_{ik} > 1$. Hence, even in this extreme case, we get that $|\mathcal{A}| + 1 < |\mathcal{A}| + 1$, Inequality 6 does not hold since the inequality should be strict. For the general case, due to the pigeonhole principle, each additional common labeled relation in $(p_{ij} + q_{ij}) + (p_{jk} + q_{jk})$ would be identical to an existing common labeled relation and thus, signifying the existence of this labeled relation in $(p_{ik} + q_{ik})$. So, the above inequality would never hold and the triangle inequality is always satisfied.

The above proof addressed the case where $|\mathcal{A}| = |\mathcal{B}| = |\mathcal{C}|$. For example if $|\mathcal{A}| \leq |\mathcal{B}| < |\mathcal{C}|$, the added dummy intervals for the matching between $|\mathcal{B}|$ and $|\mathcal{C}|$ are not paired with any interval, or exist at all, in the matching between $\mathcal{A}$ and $\mathcal{B}$. For the general case, the above proof is not sufficient, since it can be applied to only $\min\{|\mathcal{A}|, |\mathcal{B}|, |\mathcal{C}|\}$ intervals of each e-sequence. Thus we apply the technique to only the possible subset of intervals. The generalization uses the same technique.

We identify three cases: (a) $|\mathcal{A}| \leq |\mathcal{B}| < |\mathcal{C}|$, (b) $|\mathcal{B}| < |\mathcal{A}| < |\mathcal{C}|$, and (c) $|\mathcal{A}| < |\mathcal{C}| < |\mathcal{B}|$. The remaining may be derived by swapping $\mathcal{A}$ and $\mathcal{C}$.

For this, the Artemis distance can be broken down into two coefficients; one for the cost induced by the intervals participating in all three matchings and the other for the cost induced by the extra (the rest of the) intervals. However, for costs induced by the extra intervals the triangle inequality holds, i.e. the sums of unit costs of interval additions and deletions satisfy the triangle inequality since they correspond to the space of non-negative integers which is metric.

We still have to prove that for the first coefficient the triangle inequality still holds. The logic remains the same as before but the equations change slightly.

Case a The triangle inequality on the interval-pair level can be written as follows:

$$\begin{aligned} & d_m(A_i, C_k) \leq d_m(A_i, B_j) + d_m(B_j, C_k) \\ & \Rightarrow 1 - (p_{ik} + q_{ik})/|\mathcal{C}| \leq 1 - (p_{ij} + q_{ij})/|\mathcal{B}| + 1 - (p_{jk} + q_{jk})/|C| \\ & \Rightarrow |\mathcal{C}|(p_{ij} + q_{ij})/|\mathcal{B}| + (p_{jk} + q_{jk}) \leq |\mathcal{C}| + (p_{ik} + q_{ik}). \end{aligned}$$

Again, using the same reasoning as before, for $(p_{ik} + q_{ik})$ to be equal to 1, $((p_{ij} + q_{ij}) + (p_{jk} + q_{jk}))$ can be at most $|\mathcal{B}| + 1$. Even when $(p_{ij} + q_{ij}) = |\mathcal{B}|$ (so that the $|\mathcal{C}|/|\mathcal{B}|$ factor is preferred) the triangle inequality is not violated since both parts are

equal. From the pigeonhole principle, additional common relations on the lhs would be common on the rhs too. Thus the triangle inequality could not be violated.
Case b Similarly, supposing the triangle inequality does not hold, this case can be reduced to: $|\mathcal{C}|(p_{ij}+q_{ij})/|\mathcal{A}|+(p_{jk}+q_{jk}) > |\mathcal{C}|+(p_{ik}+q_{ik})$ but $(p_{ij}+q_{ij})$,$(p_{ij}+q_{ij})$, $(p_{jk}+q_{jk})$ are greater or equal to 1 and less than or equal to $|\mathcal{A}|-1$. The same approach is applicable.
Case c Same as above.
Hence, `Artemis` satisfies the triangle inequality. □

In addition, we can show that `Artemis` satisfies Leibniz's law. Assume $d(\mathcal{S},\mathcal{T}) = 0$, we provide an overview of the proof that $\mathcal{S}$ is identical to $\mathcal{T}$. First, for a zero-distance, it means that $|\mathcal{S}| = |\mathcal{T}|$, otherwise the dummy vertices would induce positive score. In addition, the labels of the i-th intervals should be the same; $\mathcal{S}_i.E = \mathcal{T}_i.E, \forall i$, otherwise $\mathcal{R}_{\varnothing}(S_i) \neq \mathcal{R}_{\varnothing}(T_i)$. Starting from left to right, the first two intervals should have the same interval-relation in both e-sequences, $RL(\mathcal{S}_1,\mathcal{S}_2) = RL(\mathcal{T}_1,\mathcal{T}_2)$; otherwise positive distance would be induced by the clause $|\mathcal{R}_{\varnothing l}(S_i) \cap \mathcal{R}_{\varnothing l}(T_j)|$. Inductively, due to the same reason, all corresponding relations in $\mathcal{S}$ and $\mathcal{T}$ should be the same.

Since the maximum matching produced by the Hungarian algorithm is symmetric, for any two e-sequences $\mathcal{S}$ and $\mathcal{T}$, it holds that $Artemis(\mathcal{S},\mathcal{T}) = Artemis(\mathcal{T},\mathcal{S})$. Finally, by definition $Artemis(\mathcal{S},\mathcal{S}) = 0$, for any e-sequence, and $Artemis(\mathcal{S},\mathcal{T}) \geq 0$, for any pair of e-sequences. Consequently, `Artemis` is a metric.

4 Indexing Artemis

Let $D = (X_1,\ldots,X_{|D|})$ be an e-sequence database, where each X_i is an e-sequence. Note that each X_i can be of arbitrary size and length. Given a query e-sequence Q, we want to find the e-sequence X_i in D, that is closer to Q under `Artemis`.

One way of solving the above problem is to apply `Artemis` in a brute-force manner. That is, we compute the `Artemis`(X_i, Q) distance score for each e-sequence $X_i \in D$ against Q, and report that e-sequence X_i with the smallest score, which effectively is the nearest neighbor of Q. We may additionally speedup this brute-force search by applying $\texttt{Artemis}_{LB}$ at each step during the search. It is apparent that the brute-force approach can be prohibitive for very large e-sequence databases. So, an indexing scheme is needed for fast e-sequence retrieval.

We propose an embedding-based framework for indexing e-sequences under the `Artemis` distance measure. We call this framework `EBESM`; an acronym for *Embedding-Based E-Sequence Matching*. Our approach has many similarities with existing embedding-based methods for indexing large database sequences under non-metric and metric spaces (Papapetrou et al. 2009a, 2011; Venkateswaran et al. 2006; Athitsos et al. 2007).

The key differences of `EBESM` against embedding-based indexing methods are:

- It does not require any training as opposed to `EBSM` (Papapetrou et al. 2011), `RBSA` (Papapetrou et al. 2009a), Athitsos et al. (2007), and Venkateswaran et al. (2006), where the pre-processing time is prohibitive for very large datasets. The first three employ time consuming cross-validation based on query samples, while the last

one selects reference sequences based on two heuristics, but still is cubic to the number of database sequences.
- It is query-sensitive, i.e., reference sequences are chosen based on the query. The key difference from the query-sensitive embeddings proposed by Athitsos et al. (2007) is that in our case the reference sequence selection is performed in an online manner based on the query size, without requiring any training, such as boosting.
- Furthermore, in Athitsos et al. (2007) prior knowledge of the query domain is required in order to perform the training step, as opposed to `EBSM`, which is query-independent.

Compared to alternative existing indexing structures and techniques for multi-dimensional data, EBESM is the only robust index applicable to event-interval sequences. More specifically:

- Existing multi-dimensional vector-indexing structures (Gaede and Günther 1998), such as R-trees (Guttman 1984), KD-trees (Bentley and Friedman 1979), or Quad-trees (Finkel and Bentley 1974)) are not directly applicable to our setting, since they require the data to be defined in a vector space, which is not the case for event-interval sequences. Even if we considered a vector-based representation similar to the one used by Artemis_LB, the dimensionality would be based on the alphabet size, which can be arbitrarily large (e.g., 254 for ASL2); hence yielding these techniques inapplicable, since it has been shown that their performance rapidly deteriorates as the data dimensionality increases (Orlandic and Yu 2002). In particular, it has been shown that when the vector dimensionality becomes higher than 15–20, sequential scan becomes faster than using any vector indexing structure to answer most of the queries (Weber et al. 1998). We confirm this in our experiments, in Sect. 6.3.3.
- Locality Sensitive Hashing (LSH) (Gionis et al. 1999) is a hash-based indexing technique typically applicable to high-dimensional data. A major requirement for the LSH framework to work is that the underlying distance function (in our case, `Artemis`) should be *LSH-able*, i.e., locality sensitive. While being a metric is a required condition for this property to hold, it is not sufficient. Hence, deriving an LSH scheme for `Artemis` is still a non-trivial task. Furthermore, and to the best of our knowledge, even for simpler distance functions such as the string edit distance, there is no LSH indexing method with theoretical guarantees.
- Recently presented indexing methods, Relation Index (Kostakis and Gionis 2015) and Karmalego (Moskovitch and Shahar 2015) were devised for subsequence matching in databases of event-interval sequences. Similarly to `Artemis`, they both focus on the relations between intervals. However, subsequence matching is orthogonal to full-sequence matching, which is the focus of this paper. More importantly, they have been devised for exact subsequence matching, and cannot be retrofitted for solving our problem.

4.1 Defining the embedding function

The embedding function $\mathcal{F}$ used by our proposed indexing scheme is defined as a mapping of an e-sequence X to a real vector space. Each dimension of the vector space is linked to a specific e-sequence, which we call *reference e-sequence*.

Given a reference e-sequence R and a target e-sequence X, we define a 1-dimensional embedding F^R, that maps X into a real number $F^R(X)$, such that

$$F^R(X) = \texttt{Artemis}(R, X)\,. \tag{7}$$

The proposed embedding function $\mathcal{F}$ is based on the extension of the above formulation by using a set of reference e-sequences $\mathcal{R} = \{R_1, \ldots, R_d\}$ instead of one. Effectively, this constructs a d-dimensional embedding $\mathcal{F}$ such that:

$$\mathcal{F}(Q, \mathcal{R}) = (F^{R_1}(Q)/d, \ldots, F^{R_d}(Q)/d)\,. \tag{8}$$

$$\mathcal{F}(X_i, \mathcal{R}) = (F^{R_1}(X_i)/d, \ldots, F^{R_d}(X_i)/d)\,. \tag{9}$$

This embedding can be seen as a variant of the basic Lipschitz embedding. The set of reference e-sequences $\mathcal{R}$ is crucial to the performance of $\mathcal{F}$ in terms of nearest neighbor retrieval, and hence it needs to be properly selected.

4.2 Selecting the reference e-sequences

We initially select a set $\mathcal{U} \subseteq D$ of k e-sequences from D, where k is relatively larger than the target size d of $\mathcal{R}$. Set $\mathcal{U}$ contains those k e-sequences in D that have the maximum variance in their pairwise `Artemis` distance, and will be used to construct set $\mathcal{R}$ during the query-embedding step. This guarantees that the selected reference e-sequences will be as different from each other as possible, which strengthens the quality of the embedding index: if two reference e-sequences R_1 and R_2 are highly similar to each other, then their embedding values $F^{R_1}(X_i)$ and $F^{R_2}(X_i)$ for any database e-sequence X_i will also be highly similar. Hence, using both R_1 and R_2 for the construction of $\mathcal{F}$ will not be that beneficial.

After selecting $\mathcal{U}$, we record the sizes of all e-sequences in $\mathcal{U}$ and insert them into a $B+$-tree. This is used later during the online construction of the query embedding (Sect. 4.4) to facilitate efficient nearest neighbor queries on the e-sequence sizes in $\mathcal{U}$.

4.3 Constructing the database embedding

Using the above formulation, we can now construct the database embedding index, given a set $\mathcal{U}$ of k reference e-sequences and an e-sequence database D.

More concretely, each e-sequence $X_i \in D$ is mapped to an embedding vector $\mathcal{F}(X_i, \mathcal{U})$ using all reference e-sequences in $\mathcal{U}$. Note that this is done by applying `Artemis` for each pair $(X_i, R_j), \forall j \in [1, k]$. This results in a set of $|D|$ k-dimensional vectors that constitute the database embedding index.

Complexity The offline computation of the embedding index $\mathcal{F}$ takes

$$O\left(\sum_{i=1}^{|D|}\sum_{j=1}^{k} C(\texttt{Artemis}(X_i, R_j))\right),$$

where $C(\texttt{Artemis}(X_i, R_j))$ is the computational cost of `Artemis` between X_i and R_j. Hence, the total offline computational cost is

$$O\left(\sum_{i=1}^{|D|}\sum_{j=1}^{k}(\max\{|X_i|, |R_j|\})^3\right) = O(|D|km^3),$$

where $m = max(|X_i|, |R_j|)$.

4.4 Constructing the query-sensitive embedding

The construction of the query embedding is an online process and its key novelty is that it is "query-sensitive". Effectively, this means that the selection of reference e-sequences for creating the query embedding depends on the query's size and it is performed in an online manner. The intuition behind this approach is that reference e-sequences of size "closer" to that of the query will provide more "informative" `Artemis` distance values. Hence, we want to construct $\mathcal{R}$ so that it includes the *top* d e-sequences ($d \leq k$) of $\mathcal{U}$ that are most similar to the query in terms of size.

Specifically, given a query e-sequence Q, we perform a d-nearest neighbor query on the $B+$-tree used to index the sizes of the e-sequences in $\mathcal{U}$, and retrieve the set of d most similar, in terms of size, e-sequences to $|Q|$. This set of d reference e-sequences will be considered as our $\mathcal{R}$ for the filter-and-refine search process.

Finally, the query-sensitive embedding vector $\mathcal{F}(Q, \mathcal{R})$ is computed online, by applying `Artemis` d times, for each $R_i \in \mathcal{R}$.

Complexity The computational time for the query-sensitive embedding construction is

$$O\left(logk + O\left(\sum_{i=1}^{d} C(Artemis(R_i, Q))\right)\right),$$

as it depends on the NN search on the $B+$-tree index and the embedding construction. This time is typically negligible compared to the total running time of `Artemis` between Q and all e-sequences in D.

4.5 Contractiveness

Contractiveness is an important property of some types of embeddings. When it holds, contractiveness can be used to guarantee that filter-and-refine retrieval will always

return the true k-NNs, for any query (Hjaltason and Samet 2003; Athitsos et al. 2007). We show that embedding $\mathcal{F}$ is contractive, which is a result of the metric property of `Artemis`.

Consider two e-sequences X_1, X_2 and a set of d reference sequences $\mathcal{R} = \{R_1, \ldots, R_d\}$. Let $\mathcal{D}(X_1, X_2)$ denote the `Artemis` distance between X_1 and X_2, and $\mathcal{D}^E(X_1, X_2)$ denote their distance in the embedding space. By the triangle inequality it holds that:

$$|\mathcal{D}(X_1, R_j) - \mathcal{D}(X_2, R_j)| \leq \mathcal{D}(X_1, X_2), \forall R_j \in \mathcal{R} .$$

Or, equivalently,

$$|F^{R_j}(X_1) - F^{R_j}(X_2)| \leq \mathcal{D}(X_1, X_2), \forall R_j \in \mathcal{R} .$$

Hence, computing the sum for all reference objects $R_j \in \mathcal{R}$, we get

$$\sum_{j=1}^{d} |F^{R_j}(X_1) - F^{R_j}(X_2)| \leq d\mathcal{D}(X_1, X_2) \tag{10}$$

$$\Rightarrow \sum_{j=1}^{d} \frac{|F^{R_j}(X_1) - F^{R_j}(X_2)|}{d} \leq \mathcal{D}(X_1, X_2) \tag{11}$$

$$\Rightarrow |\mathcal{F}(X_1, \mathcal{R}) - \mathcal{F}(X_2, \mathcal{R})| \leq \mathcal{D}(X_1, X_2) \tag{12}$$

$$\Rightarrow \mathcal{D}^E(X_1, X_2) \leq \mathcal{D}(X_1, X_2) . \tag{13}$$

Hence, $\mathcal{F}$ is contractive.

5 Filter-and-refine retrieval

5.1 The filter step

Given a query Q we identify the d closest in size to the query reference e-sequences. These e-sequences form the final set of reference e-sequences $\mathcal{R}$. We then use $\mathcal{R}$ to construct the query embedding vector $\mathcal{F}(Q, \mathcal{R})$ as described in Sect. 4.4.

We need to identify the part of the database embedding that will be used for the given Q. Recall that the database embedding was constructed using the initial set of k reference e-sequences, thus resulting in a set of $|D|$ k-dimensional vectors. For each vector $\mathcal{F}(X_i, \mathcal{U})$ we use $\mathcal{R}$ to convert it to $\mathcal{F}(X_i, \mathcal{R})$. Since $\mathcal{R} \subseteq \mathcal{U}$, this can be done simply by selecting the coordinates of $\mathcal{F}(X_i, \mathcal{U})$ that correspond to $\mathcal{R}$.

Next, we identify the l most similar database embedding vectors $\mathcal{F}(X_j, \mathcal{R})$ by performing an l-nearest neighbor search in the vector space of $\mathcal{F}$. Equivalently, the set of l e-sequences $\mathcal{L} \subseteq D$ for which $\mathcal{F}(X_j, \mathcal{R})$ has the smallest distance to $\mathcal{F}(Q, \mathcal{R})$ will be identified. During this step, we exploit the metric property of `Artemis` in order to decrease the number of comparisons of $\mathcal{F}(Q, \mathcal{R})$ against the database embedding.

Note that l is a parameter that is set by the user. Different values for l can provide trade-offs between accuracy and runtime cost.

Given a query Q, an e-sequence $X_i \in D$, and a reference e-sequence $R_j \in \mathcal{R}$, based on the triangle inequality, $\forall Q, X_i, R_j$ it holds that

$$|\texttt{Artemis}(X_i, R_j) - \texttt{Artemis}(Q, R_j)| \leq \texttt{Artemis}(Q, X_i) \, .$$

Or, equivalently,

$$|F^{R_j}(X_i) - F^{R_j}(Q)| \leq \texttt{Artemis}(Q, X_i) \, . \tag{14}$$

Hence, Eq. 14 can be used to "skip" parts of the vector comparisons in the embedding space. We achieve that by performing an additional step during the construction of the embedding index. Specifically, after computing each vector $\mathcal{F}(X_i, \mathcal{R})$ we sort the values of the coordinates in ascending order.

Then, at query time, we start by computing $\mathcal{F}(Q, \mathcal{R})$ as follows: each time we compute $F(Q, R_j)$, we apply the inequality of Eq. 14. We prune X_i from the set of candidate nearest neighbour solutions if for some coordinate j it holds that

$$|F^{R_j}(X_i) - F^{R_j}(Q)| > upper_{NN}.$$

Note that $upper_{NN}$ denotes the l-th largest distance of the e-sequences in $\mathcal{L}$ to the query Q. In other words, if X_i is guaranteed not to be in $\mathcal{L}$ we eliminate it from the filter step.

5.2 The refine step

The goal is to find the nearest neighbor of Q among the e-sequences in $\mathcal{L}$. This is done via standard lower-bound pre-filtering search(Keogh 2002). Alternatively, speedups can be achieved at this step by parallellizing the individual distance computations.

We note that selecting appropriate values for l increases the chance that the true nearest neighbor, i.e., the one given by the brute-force `Artemis` search, would be among the l selected candidates of the filter step. In other words, the accuracy of `EBESM` highly depends on l. On the other hand, a greater value for l imposes a higher computation cost since the number of candidate e-sequences, for which `Artemis` will be computed, increases. Hence, there is a trade-off between accuracy and retrieval run-time, which is tuned by the value of l.

To recap, the following three parameters are involved in the index construction and in the filter-and-refine framework (Fig. 8):

- k number of reference e-sequences used to construct the database embedding index, and are chosen from D using the maximum variance heuristic; these e-sequences form set $\mathcal{U}$;
- d number of reference e-sequences used to construct the query embedding index, and are chosen from $\mathcal{U}$ based on how similar they are to the query in terms of length; these e-sequences form set $\mathcal{R}$;

- l number of database embedding vectors that are chosen to be compared to the query embedding vector during the filter step; the e-sequences corresponding to these l vectors form set $\mathcal{L}$.

6 Experiments

We have benchmarked and evaluated the methods presented in this paper. First, we provide an overview of the used datasets in Sect. 6.1. Section 6.2 contains the evaluation of `Artemis`, while Sect. 6.3 contains the experimental evaluation of EBESM.

In summary, we benchmarked the performance of `Artemis` in terms of classification, clustering purity, noise robustness, and scalability, on eight real datasets, against three state-of-the-art methods: `Matrix` (Kostakis et al. 2011b), `DTW` (Kostakis et al. 2011a), and `IBSM` (Kotsifakos et al. 2013). The implementations of these methods have been made publicly available.[2] In addition, we studied the performance of `EBESM` on a large synthetic dataset as well as on a real dataset. We also practically demonstrate how other indexing methods, such as R-trees, are unable to provide any benefits.

6.1 Datasets

We used the following eight real datasets:

- *ASL* (Papapetrou et al. 2009b) Event labels correspond to grammatical or syntactic forms (e.g., wh-word, wh-question, verb, noun, etc.) as well as facial or gestural expressions (e.g., head tilt right, rapid head shake, eyebrow raise, etc.). An e-sequence is an expression of a sentence using sign language.
- *ASL2* This is a new dataset of American Sign Language, with a larger number of e-sequences and additional event labels.
- *Auslan* (Mörchen and Fradkin 2010) The e-sequences were derived from the Australian Sign Language dataset available in the UCI repository.[3] Each event-interval represents a word like *girl* or *right*.
- *Blocks* (Mörchen and Fradkin 2010) Event labels correspond to visual primitives obtained from videos of a human hand stacking colored blocks describing which blocks are touched and the actions of the hand (e.g., contacts blue or red, attached hand red, etc.). E-sequences represent scenarios such as atomic actions (*pickup*) or complete scenarios (*assemble*).
- *Context* (Mörchen and Fradkin 2010) Event labels were derived from categorical and numerical data describing the context of a mobile device carried by humans in different situations. Each e-sequence represents one of five different scenarios such as *street* or *meeting*.
- *Hepatitis* (Patel et al. 2008) The dataset contains information about patients who have either Hepatitis B or Hepatitis C. The event-intervals represent the results of 63 regular tests. Each e-sequence describes a series of tests taken by a patient.

[2] http://users.ics.aalto.fi/kostakis/software/ArtemisJournal.

[3] http://www.ics.uci.edu/~mlearn/MLRepository.html.

Table 2 Dataset summary

Dataset	# of e-seq.	Max e-seq time	e-seq. length			# of labels
			Min	Max	Mean	
ASL	873	5957	4	41	18	216
ASL2	1839	14968	4	93	23	254
Auslan2	200	30	9	20	12	12
Blocks	210	123	3	12	6	8
Context	240	284	47	149	81	54
Hepatitis	498	7555	15	592	108	147
Pioneer	160	80	36	89	56	92
Skating	530	6829	27	143	44	41

- *Pioneer* (Mörchen and Fradkin 2010) This dataset was constructed from the Pioneer-1 dataset available in the UCI repository. event-intervals correspond to the input provided by the robot sensors. Each e-sequence in the dataset describes one of three moving scenarios of the robot: *gripper*, *move*, *turn*.
- *Skating* (Mörchen and Fradkin 2010) event-intervals describe muscle activity and leg position of 6 professional In-Line Speed Skaters during controlled tests at 7 different speeds on a treadmill. Each e-sequence represents a complete movement cycle.

Further details regarding these datasets are displayed in Table 2.

In order to acquire large enough datasets, we have implemented a generator, called `INT-GEN`, of artificial e-sequences. The main characteristic of our generator is to produce large datasets with statistical properties similar to those of the real datasets.

Data Generator INT-GEN The generator takes the following seven inputs:

- M the number of e-sequences to be produced,
- $[N_{min}, N_{max}]$ a lower and upper limit for the number of event-intervals per e-sequence,
- max_{time} the maximum length of a sequence, which corresponds to the maximum possible start point of an event-interval in the e-sequence,
- $|\Sigma|$ the alphabet size,
- $\{\mu, std\}$ the mean and standard deviation of the length of the event-intervals.

We select the number n of temporal intervals with uniform probability within $[N_{min}, N_{max}]$. We then proceed to select an equal number of event-interval starting points. The points are selected uniformly from $[0, max_{time}]$. By selecting the duration of each interval from a Gaussian distribution and adding it to the starting point, we acquire the ending point. Finally, the label of each interval is also chosen with uniform probability from the alphabet.

A Python implementation of `INT-GEN` can be found online.[4]

[4] http://users.ics.aalto.fi/kostakis/software/INT-GEN.zip.

6.2 Benchmarking `Artemis`

In this section we benchmark `Artemis` against the other methods; `VectorDTW`, `Relation Matrix`, and `IBSM`. In Sect. 6.2.1 we examine the performance of the methods in the task of NN classiciation, while in Sect. 6.2.2 we evaluate their robustness to noise, under 1-NN queries. In Sect. 6.2.3 we investigate the run-time and scalability of the methods, and finally in Sect. 6.2.4 we examine the achieved speedups when using $Artemis_{LB}$.

6.2.1 k-NN classification and clustering purity

We explored the applicability of `Artemis` against state-of-the-art measures for the tasks of classification and clustering. More precisely, we studied the performance of `Artemis` in terms of the following two metrics:

- *k-NN classification accuracy* corresponding to the fraction of correctly classified samples using the k-NN classifier under `Artemis`;
- *Clustering purity* defined as the ratio of all samples that agree in class label with the majority element of their cluster under k-medoids.

In terms of k-NN classification, previous work (Kotsifakos et al. 2013) has demonstrated that `IBSM` performs better for most of the datasets used in this paper. However, evaluating full-matching distance measures based on their k-NN classification performance is not meaningful for all of the datasets. For example, the class-labels for the ASL and ASL2 datasets are determined by the presence of single interval-labels that attribute a particular label to a sequence (e.g., wh-word). Thus, one should just examine the presence of any of those intervals in the e-sequence, instead of performing k-NN classification using a full-sequence distance measure. Similar arguments hold for cases where a class-label is attributed by a small combination of intervals, where subsequence methods (Kostakis and Papapetrou 2015; Kostakis and Gionis 2015) are suitable.

The only datasets used in this paper and in (Kotsifakos et al. 2013), for which k-NN classification is meaningful using full-sequence distance measures, are the Blocks and Pioneer datasets. The class-labels are derived by describing high-level actions. For the Blocks dataset, all methods provide classification accuracy of more than 99%, and `Artemis` is close second in purity. For the Pioneer dataset, `Artemis` achieves the best performance together with `Matrix`. However, `Artemis` provides significantly higher clustering purity. The results are depicted in Table 3; the purity is derived from executing 100 times the k-medoids algorithm and taking the mean, while the k-NN classification results are derived by applying tenfold cross validation.

6.2.2 Noise robustness

In modern database systems, noise-tolerant retrieval of structured data is a necessity; the queries are usually noisy versions of the objects contained in the database. Hence, our retrieval method should tolerate noise. `Artemis` has been shown to be highly robust to noise, superior to `Matrix` and `DTW` when noise is present in the form of

Table 3 1-NN classification and clustering purity

Dataset	Artemis	DTW	IBSM	Matrix
Blocks 1NN	99	100	100	99
Blocks Purity	98.3	84.9	100	92
Pioneer 1NN	97.2	93.5	93.5	97.2
Pioneer Purity	79.4	65	63.8	63.8

temporally shifted intervals; detailed results can be found in our earlier work (Kostakis et al. 2011a). Here, we demonstrate the superiority of Artemis against all discussed methods, including the recently published competitor method IBSM. The methods are benchmarked against two types of artificial noise; the first one is induced by temporally shifting the intervals by an offset, while the second is induced by swapping their labels. In all cases, the noise was added off-line and all methods were tested on exactly the same distorted e-sequences, in order to rule out differences in score caused by the randomness factor.

For the first type of noise, each event in an e-sequence has probability p of being shifted; with equal probability of being shifted ahead or back in time. An additional *distortion* parameter d, as a ratio of the total time-duration of the e-sequence, denotes maximum allowed temporal shift of the intervals. The actual offset values are selected with uniform probability from $(0, d \cdot l]$, where l is the duration of e-sequence S.

The second type of noise is based on swaps of event-interval labels, while maintaining the original durations and the relations of the event-intervals. Given an e-sequence, the *swap probability* parameter determines if an event-interval will have its label swapped with that of another event-interval; every interval in the e-sequence is considered independently. The other event-interval is chosen uniformly at random from the whole e-sequence. The tested swap probability parameters values were 0.2 to 1, with step 0.2.

Noise was imputed into e-sequences that then served as the queries in 1-NN search tasks. Given a database of e-sequences, a copy of an e-sequence is distorted, based on the parameter values, and then its nearest neighbor is found by scanning the database and using a distance function to calculate the distances. This is performed for each and every e-sequence in the database. Ideally, we would like each noisy e-sequence (query) to be matched to the e-sequence from which it originated. This procedure is evaluated by using two metrics: *retrieval accuracy* and *rank of nearest neighbour*.

Before presenting the results, we provide a more general definition of the term retrieval accuracy in the context of similarity search, which is used in this paper. Consider an e-sequence database $\mathcal{D}$ and a set of query e-sequences $\mathcal{Q}$. Let $\mathcal{I}$ define a set of indices $\mathcal{I}$, such that there is a one-to-one mapping between $\mathcal{Q}$ and $\mathcal{I}$, i.e., each query $Q_j \in \mathcal{Q}$ is associated with an index in $I_j \in \mathcal{I}$ and vice-versa. In addition, each index $I_j\mathcal{I}$ is pointing to a target database e-sequence in $\mathcal{D}$. Given a distance function D and a query Q_j we denote as $NN_D(Q_j)$ the nearest neighbour e-sequence found by function D. Using the above, we define **nearest neighbour retrieval accuracy** or simply **retrieval accuracy** as the fraction of queries in $\mathcal{Q}$ for which their corresponding nearest neighbour e-sequences determined by function D, are the same as the target e-sequences defined in the set of indices $\mathcal{I}$. More formally:

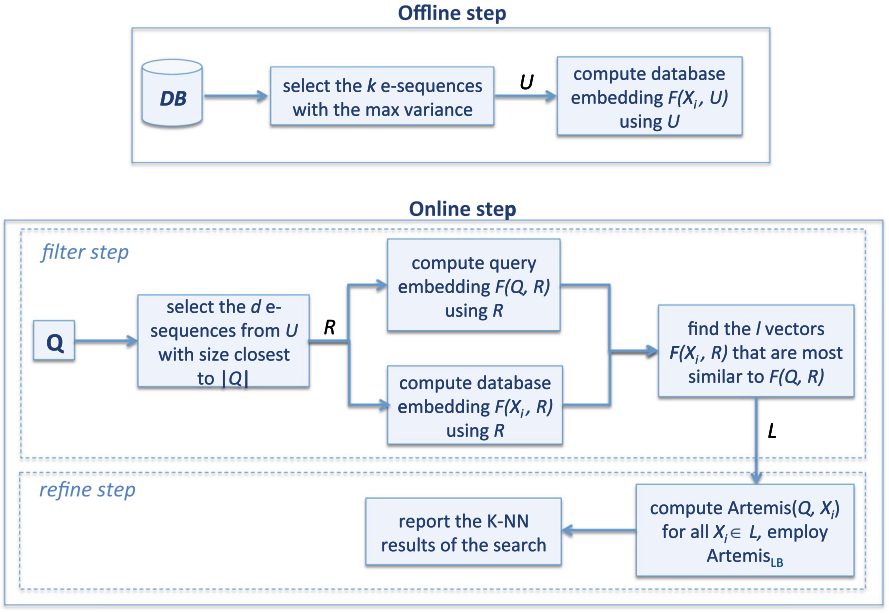

Fig. 8 The main steps of the EBESM framework

$$retrieval\ accuracy = \frac{|\{j \in [1, |\mathcal{Q}|] : NN_D(Q_j) = I_j\}|}{|\mathcal{Q}|} \tag{15}$$

Hence, the noise robustness benchmark is evaluated using the following metrics:

- *Retrieval accuracy for noise robustness* defined by Eq. 15, where $|\mathcal{Q}|$ is the set of noisy queries and $\mathcal{I}$ is the set of indices, such that I_j is referring to the originating counterpart of Q_j in $\mathcal{D}$, for each $Q_j \in \mathcal{Q}$ and $I_j \in \mathcal{I}$. In other words, for this experiment, it is the fraction of noisy queries for which the originating e-sequence is retrieved;
- *Rank of nearest neighbor* for each query, the number of database e-sequences with distance less than or equal to that of the originating counterpart.

Figures 9 and 10 show indicative results for the label-swap type of noise the performance of `Artemis` against `IBSM`, `DTW` and `Matrix` in terms of *retrieval accuracy* (Figs. 9a, 10a) and *rank of nearest neighbor*. In 6 out of 8 datasets, the order of performance is similar to that in Fig. 9. `Artemis` is superior against all other methods. `Matrix` came second and `IBSM` only performed better than `DTW`. Figure 9b denotes the ranks of the nearest neighbor.

For the cases of the Auslan2 (Fig. 10) and Blocks datasets, `IBSM` provides better accuracy than the other methods. This is due to the fact that these datasets cotain multiple e-sequences that describe the same scenarios, but the durations of the events differ. Due to that, `Artemis`, `DTW` and `Matrix` correctly identify those scenarios as identical, which distorts their scores in this experiment, while `IBSM` identifies the correct counterpart simply due to the absolute time durations of each interval. Instead,

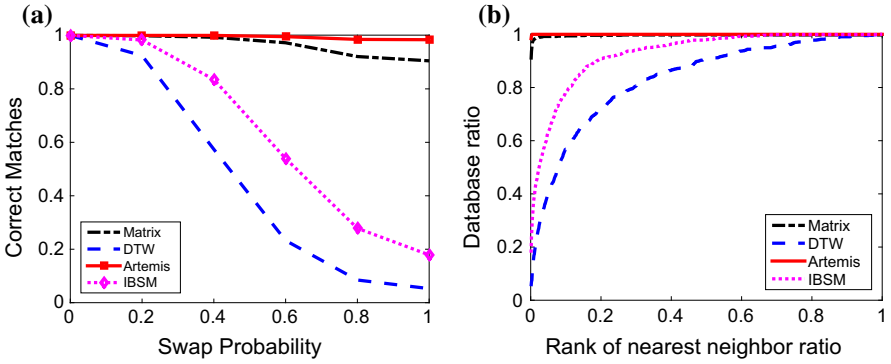

Fig. 9 Swaps-noise robustness, ASL. **a** ASL: retrieval accuracy, **b** ASL: ranks of NN. Swap probability 1.0.

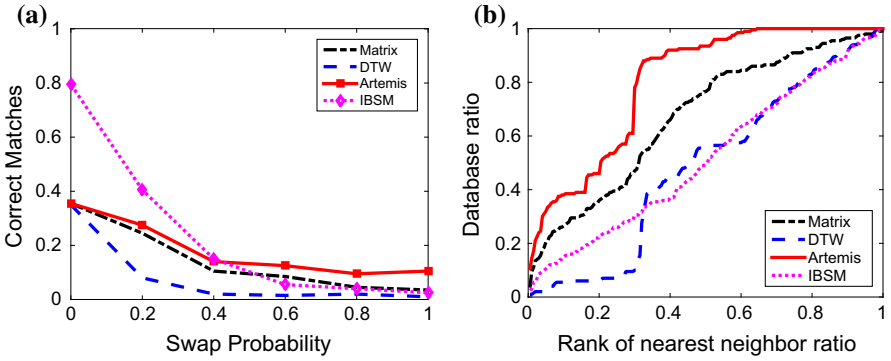

Fig. 10 Swaps-noise robustness, Auslan2. **a** Auslan2: retrieval accuracy, **b** Auslan2: ranks of NN. Swap probability 1.0.

in the presence of noise, the performance of `IBSM` deteriorates to the same or worse levels compared to the other methods. However, we see in Fig. 10b that even for those datasets `Artemis` still provides better overall Nearest Neighbor ranks, as it does for all datasets.

Similar findings apply to the interval-offset type of noise. Figure 11a depicts the retrieval accuracy for the four methods over the ASL dataset, as a function of maximum allowed distortion d, for $p = 0.6$. Figure 11b depicts the ranks of the nearest neighbors, for $d = 0.6$ and $p = 0.6$. Overall, the two methods that rely on the interval-relations are more robust than the two other methods that rely on transforming e-sequences to strings. `Artemis` performs best, and `Matrix` is marginally wosre.

6.2.3 Scalability

We benchmark the four methods in terms of scalability. For each method, we count the total time it takes to compute the whole $|D| \times |D|$ distance matrix of each real dataset. The results are depicted in Table 4. No speedup technique has been applied for any

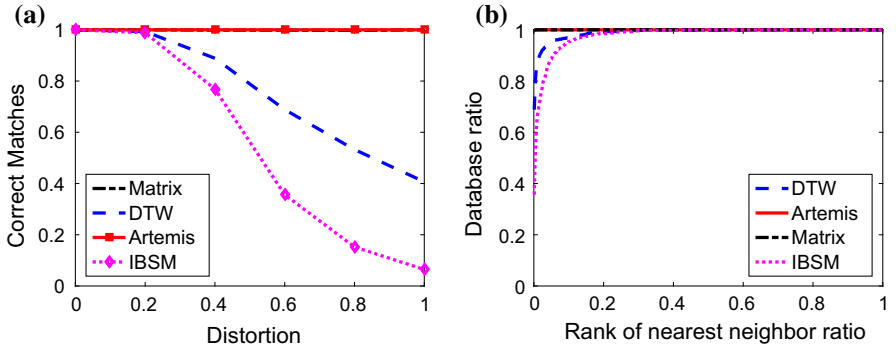

Fig. 11 Offset-noise robustness, ASL. **a** ASL: retrieval accuracy, offset probability $p = 0.6$, **b** ASL: ranks of NN. Offset probability $p = 0.6$, max distortion $d = 0.6$

Table 4 Total run-time for computing distance matrices

Dataset	Artemis	DTW	Matrix	IBSM
ASL	**24″**	1′21″	18′57″	103′55″
ASL2	**3′49″**	10′2″	126′32″	1263′23″
Auslan2	2″	**0.9″**	2.8″	**0.9″**
Blocks	**0.4″**	**0.4″**	0.9″	3.8″
Context	3′40″	**9″**	1′53″	25″
Hepatitis	156′57″	**2′27″**	31′29″	13′55″
Pioneer	14″	**3″**	45″	20″
Skating	3′37″	**51″**	2′21″	9′37″

The values in bold are the ones that are better for each experiment (for each row)

of the methods. `Artemis`, `DTW`, and `Relation Matrix` have been implemented in Java. The code of `IBSM`, provided by the authors of (Kotsifakos et al. 2013), is in Matlab, hence direct comparison of wall-clock run-times may not be applicable between `IBSM` and the other three methods. Nevertheless, we witness explicit trends in the run-times that are in accordance with the asymptotic complexity. We witness that `Atremis` is faster than all other methods on the ASL and ASL2 datasets, and ties in first place with `DTW` for the Blocks dataset. `Artemis` compared to `IBSM` and `Matrix` is better on two more datasets; 5 out of 8 in total. `DTW` performs faster for the rest of the datasets, which is expected since it solves an easier problem. Our experiments were performed on a PC running Ubuntu Linux, equipped with Intel Core i5-3470 CPU (3.20GHz).

Additionally, we study the run-times of our methods as a function of the e-sequences' size (number of event intervals), and the size of the dataset's alphabet Σ. We are only interested in how the methods scale as a function of each parameter. We normalize the results for each method based on its own run-time value when the parameter we examine has the lowest value; i.e. for sequence sizes equal to 10, and alphabet size $|\Sigma| = 10$, respectively. We do this in order to eliminate any discrimination of any method, positive or negative, due to specific implementations. Furthermore, as explained already, the run-time of `IBSM` also changes linearly as function of the

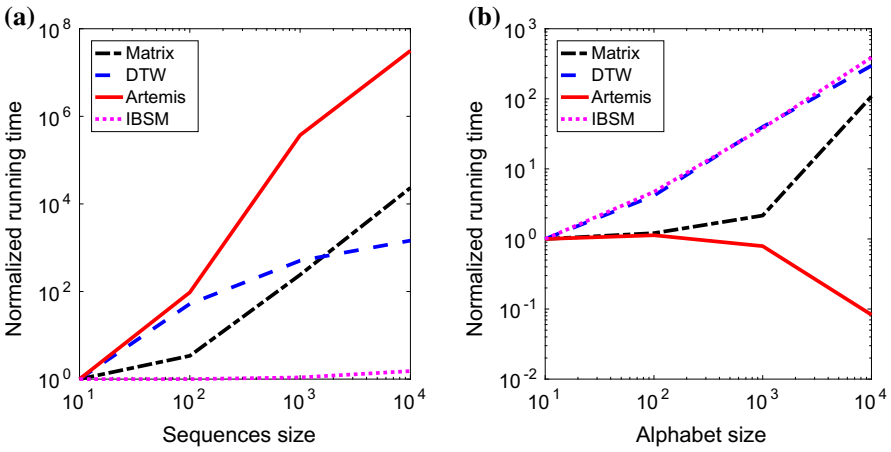

Fig. 12 Scalability experiments over synthetic datasets. **a** (log–log) varying e-sequence size $|S|$, **b** (log–log) varying alphabet size $|\Sigma|$

"sampling rate" and the maximum time-duration of any e-sequence in the dataset; the rest of the methods are immune to any scaling of the time axis. As a result, there is no fair way to compare absolute run-time values. We used `INT-GEN` to create datasets of 200 e-sequences (hence we perform 40000 comparisons), each containing 1000 intervals, and the alphabet size is $|\Sigma| = 1000$. The maximum time-duration of the e-sequences was restricted to 400.

Overall, the results are in line with each method's asymptotic complexity. The run-times as a function of the e-sequences' size are depicted in Fig. 12a. `IBSM` is affected by the number of event intervals in an e-sequence only when processing the e-sequence in order to create the vectors for the appropriate time-points; hence the increase in running-time is sub-linear. For `DTW`, the running time depends on the number of time-points that an event starts or ends; the duration of the e-sequences is an upper bound of that. Hence, `DTW`'s quadratic complexity is evident only when the ammount of distinct points increases significantly (from 10 to 10^2 and from 10^2 to 10^3 intervals). As expected, for `Matrix` the increase in running time is almost quadratic due to the enumeration of all the relations, when creating the matrices. For `Artemis` the running time increases almost with cubic rate.

Figure 12b depicts the results for varying alphabet size. `IBSM` and `DTW` scale linearly, as the size of the vectors at each time point increases linearly. `Matrix` scales quadratically for large sizes of $|\Sigma|$. Finally, for `Artemis` we witness that the total running time decreases as a function of the alphabet-size. This is an implementation side-effect when computing the intersections in $d_m(S_i, T_j)$ (Eq. 2); we can avoid computing the set-intersections when then labels are different and, instead, return '1' in constant time. Thus speeding up the comparison of e-sequences when two intervals have low probability of having the same label. The naive implementation of simply computing the intersections achieves constant running-time as a function of $|\Sigma|$.

Table 5 $Artemis_{LB}$ tightness and pruning power

Dataset	LB Tightness	1-NN pruning power
ASL	0.8837	0.7931
ASL2	0.8653	0.7393
Auslan2	0.5216	0.8619
Blocks	0.5513	0.8835
Context	0.6444	0.6370
Hepatitis	0.7166	0.4599
Pioneer	0.6189	0.4855
Skating	0.6202	0.6106

6.2.4 Lower-bounding

To assess the quality of $Artemis_{LB}$, we compute its pruning power for 1-NN queries in the database and its tightness. The pruning power is defined as the ratio of pruned comparisons, using `Artemis`, over the total number of comparisons that would have been required if the database were serially scanned. The tightness is defined as the average ratio of the lower bound distance over the distance given by `Artemis`.

The assessment of $Artemis_{LB}$ is summarized in Table 5. The higher values are observed on ASL and Blocks, for tightness and pruning power respectively, contrary to Auslan2 and Hepatitis which yield the lowest scores for tightness and pruning power respectively.

6.3 Benchmarking `EBESM`

The performance of `EBESM` has been evaluated both on synthetic and real data. In Sect. 6.3.1 we benchmark the performance of `EBESM` in terms of accuracy and pruning power using synthetic data, varying the index size (number of reference sequences). In Sect. 6.3.2 we demonstrate the applicability of `EBESM` on the largest of our real datasets, `ASL2`, and show that similar trade-offs between accuracy and pruning power can be achieved even for smaller index sizes. Finally, in Sect. 6.3.3 we show the inferiority of existing vector-indexing structures, as the number of dimensions increases.

For the synthetic data experiment, we used `INT-GEN` to create a database of 100, 000 e-sequences. The rest of the input parameters were set as follows: $N_{min} = 50$, $N_{max} = 100$, $max_{time} = 1000$, $|\Sigma| = 50$, $\mu = 500$, $std = 500$. These parameters were set based on the corresponding statistics of the real datasets used in Sect. 6.1. For the real data experiment, we used the largest real dataset of our collection, i.e., `ASL2`. For both experiments, we used the brute-force linear scan under `Artemis`, which is described in Sect. 4, as our baseline method. Lower-bounding using $\texttt{Artemis}_{LB}$ was employed for pruning. We evaluated `EBESM` in terms of the following two metrics:

- *Retrieval accuracy for NN search* for this experiment, it is the fraction of queries for which the nearest neighbor retrieved by `EBESM` agrees with the true nearest neighbor (given by linear scan). More formally, defined by Eq. 15, $|\mathcal{Q}|$ is the set

of queries and, in this case, each index $I_j \in \mathcal{I}$ is the true nearest neighbour of Q_j in $\mathcal{D}$.

- *Retrieval efficiency* which is quantified in two ways: (1) retrieval run-time, defined for each query as the ratio of the run-time of EBESM over the run-time of the brute-force approach, i.e., linear scan using Artemis:

$$\text{retrieval run-time} = \frac{\text{run-time of EBESM}}{\text{run-time of brute-force linear scan}},$$

and (2) pruning power, which, similar to lower-bounding, corresponds to the percentage l of e-sequences that have been filtered by the embedding index of EBESM.

6.3.1 Evaluation on synthetic data

We first studied the performance of EBESM on a synthetic dataset. More precisely, we benchmarked the method on a synthetic dataset two orders of magnitude larger than our largest real dataset (i.e., ASL2). For our experiments, we selected 100 queries, uniformly at random, from the database, and repeated the experiment 10 times. In our results we report the average retrieval accuracy and efficiency.

We should note that the tightness of $\texttt{Artemis}_{LB}$ on the synthetic dataset was 0.71 and the pruning power was 0.60. Effectively, this suggests that for performing a nearest neighbor query, we should expect (on average) to compute the actual Artemis distance for approximately 40% of the database, hence achieving a speedup of up to a factor of 2.5.

For determining the collection of reference e-sequences, we initially selected the 1000 e-sequences with the highest variance, i.e., we set k=1000. Note that this set was disjoint from the query set. We then experimented with different sizes of $\mathcal{R}$ and studied trade-offs between l and $|\mathcal{R}|$. Note that $|\mathcal{R}|$ defines the size of the embedding index, while l regulates the fraction of the e-sequence database that will be passed to the refine-step for computing the actual Artemis distance. Hence, trade-offs between accuracy and retrieval efficiency can be obtained while varying l, for different embedding index sizes.

Table 6 depicts the performance in terms of retrieval run-time of EBESM for different number of reference e-sequences in $\mathcal{R}$. The best performance was obtained when 20 e-sequences were chosen. We can see that EBESM can achieve an accuracy of 100% with a retrieval run-time of 8.91%. This means that by using 20 reference e-sequences, EBESM retrieves the correct nearest neighbour with 100% probability, while requiring only 8.91% of the brute-force run-time. The actual run-time when using 20 reference e-sequences was on average 34.03 s per query, using a single CPU.

As the number of reference e-sequences increases, the retrieval run-time also increases due to the required time for the embedding step, while still maintaining at least an order of magnitude speedup with respect to the brute-force.

In addition, Table 7 depicts the pruning power in terms of percentages of database e-sequences, l, that were filtered during the filter step of EBESM. As expected, when 20 reference e-sequences are selected, the filter step of EBESM can eliminate over 93% of the database e-sequences, while maintaing an accuracy of 100%, and finally

Table 6 Trade-offs between retrieval runtime versus accuracy of EBESM on the synthetic dataset: 100 queries, 100,000 e-sequences, $N_{min} = 50$, $N_{max} = 100$, $max_{time} = 1000$, $|\Sigma| = 50$, $\mu = 500$, and $std = 500$

Accuracy	20	10	40	80	160
100%	**8.91**%	12.23%	13.32%	14.40%	15.57%
99%	**5.89**%	9.57%	9.89%	10.12%	11.42%
98%	**4.21**%	8.25%	8.12%	8.83%	9.32%
95%	**3.16**%	7.56%	7.68%	7.18%	8.18%
90%	**2.89**%	6.19%	7.08%	6.32%	7.92%
85%	**2.58**%	5.57%	6.33%	5.45%	6.67%
80%	**2.05**%	4.53%	4.96%	5.02%	5.79%

Each column corresponds to parameter d, i.e., the number of reference e-sequences used by EBESM
The values in bold are the ones that are better for each experiment (for each row)

Table 7 Trade-offs between pruning power (percentage of filtered e-sequences l), versus accuracy of EBESM on the synthetic dataset: 100 queries, 100,000 e-sequences, $N_{min} = 50$, $N_{max} = 100$, $max_{time} = 1000$, $|\Sigma| = 50$, $\mu = 500$, and $std = 500$

Accuracy	20	10	40	80	160
100%	**6.98**%	10.12%	11.22%	12.15%	12.79%
99%	**4.23**%	7.17%	7.45%	8.87%	9.19%
98%	**3.56**%	5.78%	6.12%	6.98%	7.87%
95%	**2.25**%	5.12%	5.57%	5.96%	6.85%
90%	**2.02**%	4.01%	5.17%	5.75%	6.12%
85%	**1.87**%	3.37%	4.02%	5.21%	5.89%
80%	**1.18**%	2.44%	2.58%	2.87%	3.88%

Each column corresponds to parameter d, i.e., the number of reference e-sequences used by EBESM
The values in bold are the ones that are better for each experiment (for each row)

pass over to the computationally expensive refine step only a small fraction (6.98%) of the database e-sequences.

The reader may wonder whether these parameters are optimal for every dataset. The answer to this question is not that straightforward. As it has been shown in similar approaches, e.g., by Papapetrou et al. (2011), the size and contents (reference e-sequences) of the embedding highly depend on the database to be indexed. Hence, in order to achieve the best trade-offs between accuracy and retrieval run-time or l, the proper reference e-sequences should be chosen. This can be achieved by using a sample of the expected queries and determining the appropriate set of reference e-sequences, just as demonstrated in this paper.

6.3.2 Evaluation on real data

Next, we evaluated EBESM on real data. For this experiment we used the largest real dataset of our collection, i.e., ASL2. In addition, we used a set of 100 queries selected

Table 8 Trade-offs between pruning power (percentage of filtered e-sequences l) versus accuracy of EBESM on ASL2

Accuracy	5	10	20	40
100%	**11.86**%	13.98%	15.50%	17.71%
99%	**7.99**%	10.46%	11.56%	12.11%
98%	**6.78**%	8.90%	9.89%	11.55%
95%	**5.55**%	8.10%	8.87%	9.56%
90%	**3.61**%	5.56%	6.88%	8.12%
85%	**2.91**%	4.47%	5.50%	6.88%
80%	**2.62**%	4.11%	4.89%	5.81%

Each column corresponds to parameter d, i.e., the number of reference e-sequences used by EBESM
The values in bold are the ones that are better for each experiment (for each row)

uniformly at random from the database. Again, we repeated the experiments for 10 random query sets.

The selection of reference e-sequences for EBESM was performed by first selecting a set of 100 e-sequences with the highest variance, i.e., we set k=100. Table 8 depicts the performance in terms of pruning power of EBESM on ASL2, for different number of reference e-sequences in $\mathcal{R}$. In this case, the best performance was obtained when 5 e-sequences were chosen.

We can see that EBESM can achieve an accuracy of 100% when the percentage of filtered e-sequences is 11.86% of the database size. Again, this suggests that by using 5 reference e-sequences, EBESM retrieves the correct nearest neighbour with 100% probability, while requiring only 11.86% of the brute-force. The actual runtime when using 5 reference e-sequences was on average 10m s per query. As opposed to the synthetic dataset case, the best performance was achieved at a lower number of reference e-sequences, and it monotonically deteriorates as the number of reference e-sequences increases.

6.3.3 Benchmarking against R-trees

As discussed earlier, multi-dimensional vector-indexing structures are hampered by the curse of dimensionality, and hence are unable to handle efficiently vector spaces with more than 15–20 dimensions (Orlandic and Yu 2002). To experimentally confirm this claim, we benchmarked the performance of an R-tree (Guttman 1984) on all eight real datasets in terms of 1-NN search. Since R-trees (like all vector-indexing techniques) require the data to be in a vector form, loading the e-sequences directly to the R-tree at their original form was not possible and had to be converted to vectors. We considered two vector-based representations. In the first case (we call this CASE I), each e-sequence was converted to a vector of size equal to the alphabet size of the dataset. Each value in the vector was equal to the number of occurrences of the corresponding alphabet label in that e-sequence. In the second case (CASE II), we considered all pairs of relation types that can occur between the event labels in the dataset; hence, producing a total of $7 \times |\Sigma|^2$ dimensions ($|\Sigma|$ is the alphabet's size),

Table 9 R-tree benchmark; average ratio of pruned leaf-nodes

Dataset	CASE I (%)	CASE II (%)
ASL	0	0
ASL2	0	0
Auslan2	28	0
Blocks	25	0
Context	0	0
Hepatitis	0	0
Pioneer	0	0
Skating	0	0

since there are seven possible relations that can occur between any pair of event labels. The reader might notice that these two representations are already employed by Artemis_LB and `Matrix`, respectively.

We quantify the performance (pruning power) of an R-tree as the average ratio of pruned visits to the R-tree leaf-nodes during the queries. The leaf-nodes store data elements, i.e., e-sequences; visiting all leaf-nodes corresponds to linear scan. Table 9 depicts the results over all the datasets. For CASE I, the R-tree avoids visiting some leaf-nodes only for Blocks and Auslan2, since their dimensionality is rather low, 12 and 8 respectively. The performance for all other datasets is equivalent to linear scan (in practical run-time much worse), due to their very high dimensionality. For CASE II, all datasets performed similar to linear scan (again in practice much worse) due to the high number of dimensions.

In practice the actual performance values in terms of retrieval runtime is much worse than linear scan. In our experiments, we witnessed instances performing up to 100 times slower than linear scan, which is in accordance with the results presented in Orlandic and Yu (2002). In addition, we need to point out that when considering vector-based indexing methods (and the transformation of e-sequences to vectors happens like in cases I or II), one should expect to experience one or more of the limitations exhibited by Artemis_LB and `Matrix`.

6.4 Lessons learned

Solution to Problem 1 `Artemis` is the only one that satisfies the metric property, and solves Problem 1. `IBSM` and `DTW` do not consider interval relations. Hence, as described using Fig. 2, pair-comparison results by `IBSM` and `DTW` can be arbitrarily bad. While `Relation Matrix` enumerates interval relations, it fails to distinguish scenarios such as those in Fig. 4. Since `DTW`, `IBSM` and `Matrix` violate the metric property, and in particular sub-property (ii) as described in Problem 1, even under linear-scan the returned results would require a post-query filtering step to verify them against false matches. For the same reason, when devising indexing methods, theoretical guarantees for trade-offs between retrieval accuracy and efficiency cannot be easily provided, while in some cases approximate solutions are inevitable.

Noise robustness In our evaluation, Artemis proved to be the most robust method to two types of noise. The reason is not only the focus on the relations of the event-intervals, which is common with Relation Matrix, but also the underlying nature of the method that attempts to find correspondence between event-intervals. The later allows Artemis to easily identify the originating counterpart of noisy e-sequences; e-sequences with the same number of each label yield overall lower distance scores than others with different labels and size. In addition, Artemis provides better NN rank values. Furthermore, Artemis is the only one of the 4 methods that does not produce false matches. On the other hand, DTW examines the e-sequences point-by-point and the event-intervals out of context. This makes it more sensitive to minor edit operations. Consequently, DTW displayed a decline in performance in accordance with the increase of noise. IBSM suffers from the same problem as DTW, since it also ignores the temporal relations between events.

The examined types of artificial noise retained the same number of event intervals and labels. If the noise is in the form of removing intervals, the problem becomes that of subsequence search. In this case, existing work (Kostakis and Gionis 2015) has demonstrated that methods focusing on the relations between intervals provide better search accuracy and recall.

Time complexity Artemis is the only measure whose complexity is a function of only the number of event-intervals. The other methods explicitly depend also on the size of the dataset alphabet. In addition, IBSM depends on the absolute time duration of e-sequences as well as on the time duration of the largest e-sequence of the dataset. We noticed that DTW was in the majority of cases faster than the rest of the methods. This is expected since DTW solves an easier problem. Still, it performed slower than Artemis for ASL and ASL2, due to the alphabet size; for those datasets Artemis performs best. For Artemis, IBSM, and Matrix, Artemis performs better in 5 out 8 datasets compared to the other two methods.

For the scenario of querying a database, Artemis has an important advantage. IBSM requires pre-processing the whole dataset and normalizing the e-sequences based on the time duration of the longest one. As a result, the score of any pair of e-sequences changes depending on the length of the longest one in the dataset. More importantly, IBSM requires a-priori knowledge of the length of the longest possible time duration of a query sequence, otherwise the whole database needs to be pre-processed. Consider a scenario where we receive consecutive queries of increasing length and all of them are longer than any sequence in the database. In this case, we would need to pre-process the whole database once for each query. The same applies to IBSM, DTW, and Matrix for the case of the alphabet. If the whole alphabet of the dataset is not a-priori known, then no vectors or matrices can be precomputed, and hence the whole database would need to be processed for every query. Instead, for Artemis, it is possible to perform the mapping step (described in Sect. 3.2) offline for the e-sequences of the database.

Lower-bounding To speed up search using Artemis, our lower bound technique $Artemis_{LB}$ proved to be significantly tight; the average tightness ranged from 52.1% (Auslan2 dataset) up to 88% (ASL dataset). This translates to a pruning power of 45.9% to 88.3% over the brute-force sequential search of the database. Overall, the lower bound is tight enough to filter out most irrelevant candidates, still not tight enough

to render useless the information provided by Artemis. For searching through large databases, an indexing scheme is still needed.

Indexing In terms of indexing, EBESM can achieve a speedup of over an order of magnitude compared to the brute-force approach. The synthetic data experiment suggests that using only 20 reference e-sequences for the embedding construction, EBESM retrieves the correct nearest neighbour with 100% probability, while requiring only 8.91% of the brute-force run-time. Similar conclusions are drawn from the real data experiment. The only requirement of EBESM is to have prior knowledge of the expected query sizes in order to optimize the performance of the query-sensitive embedding. Finally, we confirmed the inefficiency of vector-based indexing methods, since they are suffer from the curse of dimensionality.

Overall Artemis is highly robust versus the two types of noise and that allows to find the originating counterparts of noisy queries. In certain cases Artemis can be used out-of-the-box for classification purposes. While Artemis' expected run-time might appear to be the highest, it performs better than the baseline methods when the alphabet size grows large. Its asymptotic complexity can be circumvented by the use of lower bounding and indexing techniques such as the ones we presented. For identifying exact similarity of e-sequences, Artemis is the only one that does not violate the identity of indiscernibles. As a result, Artemis is the only reliable method for searching and retrieving similar event-interval sequences.

7 Related work

The vast amount of existing work on sequences of event-intervals has been focusing on pattern and association rule mining, while limited attention has been given to similarity and indexing.

Within the area of mining sequences of event-intervals, several approaches (Lin 2003; Villafane et al. 2000) consider the extraction of patterns, where event-intervals appear sequentially and are not labeled, while others (Giannotti et al. 2006) consider temporally annotated sequential patterns where transitions from one event to another have a time duration. A graph-based approach (Hwang et al. 2004) represents each temporal pattern by considering only two types of relations between event-intervals (*follow* and *overlap*), while in (Ale and Rossi 2000), the lifetime of an item is defined as the time between its first and the last occurrence and the temporal support is calculated with respect to this event-interval.

A large variety of Apriori-based techniques (Kam and Fu 2000; Abraham and Roddick 1999; Chen and Petrounias 1999; Höppner 2001; Höppner and Klawonn 2001; Mooney and Roddick 2004; Laxman et al. 2007) for finding temporal patterns, episodes, and association rules on interval-based event sequences have been proposed. In addition, more advanced candidate generation techniques and tree-based structures have been employed by various methods (Sacchi et al. 2007; Moskovitch and Shahar 2015; Winarko and Roddick 2007; Papapetrou et al. 2005, 2006, 2009b; Moskovitch and Shahar 2009) which apply efficient pruning techniques, thus reducing the inherent exponential complexity of the mining problem, while a non-ambiguous event-interval representation is defined by Wu and Chen (2007) that considers start

and end points of event sequences and converts them to a sequential representation. The main weakness of performing such mapping is the fact that the candidate generation process becomes more cumbersome while introducing redundant patterns. An alternative approach (Chen et al. 2010) that extends Allen's relations and employs a sequence-based representation that takes into account the co-occurrences of labels over time (referred to as "co-incidence representation"). This representation results in a more compact mapping. Nonetheless, as we also demonstrated in Sect. 1, such sequential representations cannot handle temporal relations between event-intervals of the same label.

Another line of research has studied the problem of mining semi-partial orders of time intervals (Mörchen and Fradkin 2010). Such patterns have been shown to be more flexible than patterns over full intervals, as well as more useful as features for classification. Moreover, several pattern reduction approaches have been developed for event-interval sequences. One approach is to look for *margin-closed* patterns (Mörchen 2010; Mörchen and Fradkin 2010), which can significantly reduce the number of reported patterns by favoring longer patterns and suppressing shorter patterns with similar frequencies. Furthermore, an efficient method for mining closed patterns of interval-based events has been proposed (Chen et al. 2011). For a more thorough review of recent work in this particular area the reader may refer to Chen et al. (2015). Finally, a unifying view of temporal concepts and data models has been formulated in (Mörchen 2007) to enable categorization of existing approaches to unsupervised pattern mining from symbolic temporal data; time point-based methods and interval-based methods as well as univariate and multivariate methods are considered.

As far as supervised learning methods are concerned, several methods have been proposed within the context of multi-variate time series classification (Batal et al. 2009, 2012; Moskovitch and Shahar 2014b; Fradkin and Mörchen 2015). These approaches first map each time series channel to a set of event-intervals, and then extract patterns (e.g., frequent or closed) of event-intervals. The intuition is that such patterns convey the dominant temporal dependencies across the time series channels. Next, the time series are mapped to feature vectors using the most important features given by, e.g., the chi-square test. These vectors are then passed to traditional classifiers, such as an SVM. The usefulness of these methods has been demonstrated extensively within healthcare and more particularly for classifying electronic health records (Batal et al. 2013) and for prediction of renal damage in patients with diabetes (Klimov et al. 2015). A novel multi-variate time series discretization method has been proposed for identifying the cutoffs that will best discriminate among classes using the interval distribution (Moskovitch and Shahar 2014a). Despite their applicability and efficiency the problems studied by these methods are orthogonal to ours.

Similarity matching for sequences of event-intervals has been given limited attention in the literature. Several metrics have been proposed, such as Relation Matrix (Kostakis et al. 2011b), Artemis (Kostakis et al. 2011a), and IBSM (Kotsifakos et al. 2013). The first method simply enumerates all event-interval relations in the two e-sequences and compares them as bags-of-words, the second one attempts to find corresponding pairs between the two e-sequences, while the third maps the e-sequences to vectors and employs the Euclidean distance.

Intervals provide greater encoding power over distinct values. As a result, they may be employed in different scenarios, which however are orthogonal to ours. For example, Yi and Roh (2004) examine events where the observations are intervals and the observation values have upper and lower limits; despite that, the events are instantaneous. In our case, the observations are 'precise' in the sense that the labels are clearly defined, but the duration of the events are time intervals.

Similarity matching is crucial for other types of sequences and data structures. For the case of time series, standard measures include the Euclidean distance (Faloutsos et al. 1994) and several dynamic programming-based methods, such as Dynamic Time Warping (DTW) (Kruskall and Liberman 1983) and variants (e.g., cDTW (Sakoe and Chiba 1978), EDR (Chen and Özsu 2005), ERP (Chen and Ng 2004)) that are robust to misalignments and warps in the time axis. Alternative methods allowing for gaps in the alignment include, e.g., LCSS (Maier 1978) and variants (Han et al. 2007). Moreover, similar methods have been proposed for strings, such as the Edit distance and the Needleman-Wunch method (Needleman and Wunsch 1970), methods based on q-grams (Yang et al. 2008; Li et al. 2008), as well as methods for subsequence matching and local alignment, such as Smith-Waterman (Smith and Waterman 1981), the Burrows-Wheeler Transforms (BWT) (Burrows and Wheeler 1994), or WHAM (Li et al. 2012), which have highly applicability in biological sequences. Finally, variants of graph matching for graphs exist in the literature, such as those of Umeyama (1988); Bunke (2000); Kostakis (2014). An extensive discussion on these methods is far beyond the scope of this paper. The competitor methods of `VectorDTW` and `IBSM` are adaptations of some of the above methods in the context of sequences of temporal intervals.

Finally, several indexing methods have been proposed for sequential data. One family of methods includes embedding-based schemes where reference objects are used for mapping the original complex space to a vector space. Such indexing mechanisms have been proposed for time series (Papapetrou et al. 2011) and biological sequences (Venkateswaran et al. 2006; Papapetrou et al. 2009a). Alternative vector indexing techniques, such as R-trees (Guttman 1984), KD-trees (Bentley and Friedman 1979), Quad-trees (Finkel and Bentley 1974), and LSH (Gionis et al. 1999) are not directly applicable to our problem setting. The diversion ad advantages of `EBESM` compared to these schemes are extensively discussed in Sect. 4.

8 Conclusions

We studied the problem of comparing sequences of event-intervals and presented `Artemis`, a robust and efficient distance measure for solving the problem. We proved that `Artemis` is metric under our problem-setting and provided a linear-time lower-bound for speeding up its computation. In addition, we introduced an indexing method, `EBESM`, for nearest neighbour search under `Artemis`. The method is based on embeddings and it can achieve speedups of over an order of magnitude compared to brute-force search. The proposed methods were tested on real datasets from multiple different domains as well as a large synthetic dataset.

Directions for future work include the study of the problem of subsequence matching in sequences of interval-based events as well as devising on-line algorithms, where

the active intervals are given in a streaming or bursty fashion. Furthermore, the distance functions that we presented required at least quadratic time with respect to the the length of the e-sequences. It would be interesting to study the use of randomized algorithms in an effort to provide faster solutions.

Acknowledgements The work of Orestis Kostakis was supported in party by the Helsinki Doctoral Education Network in Information and Communications Technology (HICT). The work of Panagiotis Papapetrou was supported in part by the Stockholm City Council (Stockholms Läns Landsting).

References

Abraham T, Roddick JF (1999) Incremental meta-mining from large temporal data sets. In: ER '98: Proceedings of the Workshops on Data Warehousing and Data Mining, pp 1–37

Ale JM, Rossi GH (2000) An approach to discovering temporal association rules. In: Proceedings of the ACM Symposium On Applied Computing, pp 294–300

Allen JF (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843

Athitsos V, Hadjieleftheriou M, Kollios G, Sclaroff S (2007) Query-sensitive embeddings. ACM Trans Database Syst 32(2). doi:10.1145/1242524.1242525

Batal I, Sacchi L, Bellazzi R, Hauskrecht M (2009) Multivariate time series classification with temporal abstractions. In: FLAIRS

Batal I, Fradkin D, Harrison J, Moerchen F, Hauskrecht M (2012) Mining recent temporal patterns for event detection in multivariate time series data. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, pp 280–288

Batal I, Valizadegan H, Cooper GF, Hauskrecht M (2013) A temporal pattern mining approach for classifying electronic health record data. ACM Trans Intell Syst Technol 4(4):63:1–63:22

Bentley JL, Friedman JH (1979) Data structures for range searching. ACM Comput Surv 11(4):397–409. doi:10.1145/356789.356797

Berendt B (1996) Explaining preferred mental models in Allen inferences with a metrical model of imagery. In: Proceedings of the Conference of the Cognitive Science Society, pp 489–494

Bergen B, Chang N (2005) Embodied construction grammar in simulation-based language understanding. In: Construction grammars: cognitive grounding and theoretical extensions, vol 3, pp 147–190

Bunke H (2000) Recent developments in graph matching. In: IEEE 15th International Conference on Pattern Recognition, vol 2, pp 117–124

Burrows M, Wheeler DJ (1994) A block-sorting lossless data compression algorithm. Tech. Rep. 124, Systems Research Center, Palo Alto. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.37.6774

Chen X, Petrounias I (1999) Mining temporal features in association rules. In: Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases, Springer, pp 295–300

Chen L, Ng R (2004) On the marriage of l_p-norms and edit distance. In: VLDB, pp 792–803

Chen L, Özsu MT (2005) Robust and fast similarity search for moving object trajectories. In: SIGMOD, pp 491–502

Chen YC, Jiang JC, Peng WC, Lee SY (2010) An efficient algorithm for mining time interval-based patterns in large database. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10, pp 49–58

Chen YC, Peng WC, Le SY (2011) CEMiner- an effcient algorithms for mining closed patterns from interval-based data. In: Proceedings of the IEEE International Conference on Data Mining (ICDM)

Chen YC, Weng JTY, Hui L (2015) A novel algorithm for mining closed temporal patterns from interval-based data. KAIS 46(1):151–183

Faloutsos C, Ranganathan M, Manolopoulos Y (1994) Fast subsequence matching in time-series databases. In: Proceedings of the 1994 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '94, pp 419–429

Finkel RA, Bentley JL (1974) Quad trees: a data structure for retrieval on composite keys. Acta Inf 4:1–9. doi:10.1007/BF00288933

Fradkin D, Mörchen F (2015) Mining sequential patterns for classification. Knowl Inf Syst 45(3):731–749

Gaede V, Günther O (1998) Multidimensional access methods. ACM Comput Surv 30(2):170–231
Giannotti F, Nanni M, Pedreschi D (2006) Efficient mining of temporally annotated sequences. In: Proceedings of the 6th SIAM Data Mining Conference, vol 124, pp 348–359
Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proceedings of the 25th International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, VLDB '99, pp 518–529. http://dl.acm.org/citation.cfm?id=645925.671516
Guttman A (1984) R-trees: a dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '84, pp 47–57. doi:10.1145/602259.602266
Han TS, Ko SK, Kang J (2007) Efficient subsequence matching using the longest common subsequence with a dual match index. In: International Workshop on Machine Learning and Data Mining in Pattern Recognition, Springer, pp 585–600
Hjaltason G, Samet H (2003) Properties of embedding methods for similarity searching in metric spaces. IEEE Trans Pattern Anal Mach Intell 25(5):530–549
Höppner F (2001) Discovery of temporal patterns: learning rules about the qualitative behaviour of time series. In: Proceedings of the European Conference on Principles of Knowledge Discovery in Databases, pp 192–203
Höppner F, Klawonn F (2001) Finding informative rules in interval sequences. In: Proceedings of the International Symposium on Advances in Intelligent Data Analysis, pp 123–132
Hwang SY, Wei CP, Yang WS (2004) Discovery of temporal patterns from process instances. Comput Ind 53(3):345–364
Itakura F (1975) Minimum prediction residual principle applied to speech recognition. IEEE Trans Acoust Speech and Signal Process 23(1):67–72
Kam P, Fu AW (2000) Discovering temporal patterns for interval-based events. In: Proceedings of the 2nd International Conference on Data Warehousing and Knowledge Discovery, pp 317–326
Keogh E (2002) Exact indexing of dynamic time warping. In: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), pp 406–417
Klimov D, Shknevsky A, Shahar Y (2015) Exploration of patterns predicting renal damage in patients with diabetes type II using a visual temporal analysis laboratory. J Am Med Inform Assoc 22(2):275–289
Kosara R, Miksch S (2001) Visualizing complex notions of time. Stud Health Technol Inform 1:211–215
Kostakis O (2014) Classy: fast clustering streams of call-graphs. Data Min Knowl Discov 28(5–6):1554–1585
Kostakis O, Gionis A (2015) Subsequence search in event-interval sequences. In: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, pp 851–854
Kostakis O, Papapetrou P (2015) Finding the longest common sub-pattern in sequences of temporal intervals. Data Min Knowl Discov 29(5):1178–1210
Kostakis O, Papapetrou P, Hollmén J (2011a) Artemis: assessing the similarity of event-interval sequences. In: Proceedings of the Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD 2011), pp 229–244
Kostakis O, Papapetrou P, Hollmén J (2011b) Distance measure for querying arrangements of temporal intervals. In: Proceedings of Pervasive Technologies Related to Assistive Environments
Kotsifakos A, Papapetrou P, Athitsos V (2013) IBSM: Interval-based sequence matching. In: Proceedings of SIAM Conference on Data Mining, pp 596–604
Kruskall JB, Liberman M (1983) The symmetric time warping algorithm: from continuous to discrete. In: Time warps, Addison-Wesley
Laxman S, Sastry P, Unnikrishnan K (2007) Discovering frequent generalized episodes when events persist for different durations. IEEE Trans Knowl Data Eng 19(9):1188–1201
Levenshtein VI (1966) Binary codes capable of correcting deletions, insertions, and reversals. Sov Phys 10(8):707–710
Li C, Lu J, Lu Y (2008) Efficient merging and filtering algorithms for approximate string searches. In: International Conference on data Engineering (ICDE)
Li Y, Patel JM, Terrell A (2012) Wham: a high-throughput sequence alignment method. ACM Trans Database Syst (TODS) 37(4):28
Lin JL (2003) Mining maximal frequent intervals. In: Proceedings of the ACM Symposium On Applied Computing, pp 624–629

Maier D (1978) The complexity of some problems on subse- quences and supersequences. J ACM 25(2):322–336
Mooney C, Roddick JF (2004) Mining relationships between interacting episodes. In: Proceedings of the 4th SIAM International Conference on Data Mining
Mörchen F (2007) Unsupervised pattern mining from symbolic temporal data. SIGKDD Explor Newsl 9:41–55
Mörchen F (2010) Temporal pattern mining in symbolic time point and time interval data. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data mining, ACM, KDD '10, pp 2:1–2:1
Mörchen F, Fradkin D (2010) Robust mining of time intervals with semi-interval partial order patterns. In: Proceedings of the SIAM International Conference on Data Mining, pp 315–326
Moskovitch R, Shahar Y (2009) Medical temporal-knowledge discovery via temporal abstraction. Proceedings of the AMIA Annual Symposium 2009:452–456
Moskovitch R, Shahar Y (2014a) Classification-driven temporal discretization of multivariate time series. Data Min Knowl Discov 29(4):871–913
Moskovitch R, Shahar Y (2014b) Classification of multivariate time series via temporal abstraction and time intervals mining. Knowl Inf Syst 45(1):35–74
Moskovitch R, Shahar Y (2015) Fast time intervals mining using the transitivity of temporal relations. Knowl Inf Syst 42(1):21–48
Munkres J (1957) Algorithms for the assignment and transportation problems. J Soc Ind Appl Math 5(1):32–38
Needleman SB, Wunsch CD (1970) A general method applicable to the search for similarities in the amino acid sequence of two proteins. J Mol Biol 48(3):443–453
Orlandic R, Yu B (2002) A retrieval technique for high-dimensional data and partially specified queries. Data Knowl Eng 42(1):1–21. doi:10.1016/S0169-023X(02)00023-X
Pachet F, Ramalho G, Carrive J (1996) Representing temporal musical objects and reasoning in the MusES system. J New Music Res 25(3):252–275
Papapetrou P, Kollios G, Sclaroff S, Gunopulos D (2005) Discovering frequent arrangements of temporal intervals. In: Proceedings of IEEE International Conference on Data Mining, pp 354–361
Papapetrou P, Benson G, Kollios G (2006) Discovering frequent poly-regions in DNA sequences. In: Proceedings of the IEEE ICDM Workshop on Data Mining in Bioinformatics
Papapetrou P, Athitsos V, Kollios G, Gunopulos D (2009a) Reference-based alignment in large sequence databases. Proc VLDB Endow 2(1):205–216
Papapetrou P, Kollios G, Sclaroff S, Gunopulos D (2009b) Mining frequent arrangements of temporal intervals. Knowl Inf Syst 21:133–171
Papapetrou P, Athitsos V, Potamias M, Kollios G, Gunopulos D (2011) Embedding-based subsequence matching in time-series databases. ACM Trans Database Syst 36(3):17:1–17:39
Patel D, Hsu W, Lee M (2008) Mining relationships among interval-based events for classification. In: Proceedings of the 28th ACM SIGMOD International Conference on Management of Data, ACM, pp 393–404
Pissinou N, Radev I, Makki K (2001) Spatio-temporal modeling in video and multimedia geographic information systems. GeoInformatica 5(4):375–409
Rakthanmanon T, Campana B, Mueen A, Batista G, Westover B, Zhu Q, Zakaria J, Keogh E (2012) Searching and mining trillions of time series subsequences under dynamic time warping. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, pp 262–270
Sacchi L, Larizza C, Combi C, Bellazzi R (2007) Data mining with temporal abstractions: learning rules from time series. Data Min Knowl Discov 15(2):217–247
Sakoe H, Chiba S (1978) Dynamic programming algorithm optimization for spoken word recognition. Trans ASSP 26:43–49
Smith TF, Waterman MS (1981) Identification of common molecular subsequences. J Mol Biol 147(1):195–197
Umeyama S (1988) An eigendecomposition approach to weighted graph matching problems. IEEE Trans Pattern Anal Mach Intell 10(5):695–703
Venkateswaran J, Lachwani D, Kahveci T, Jermaine C (2006) Reference-based indexing of sequence databases. In: International Conference on Very Large Databases (VLDB), pp 906–917

Villafane R, Hua KA, Tran D, Maulik B (2000) Knowledge discovery from series of interval events. Intell Inf Syst 15(1):71–89

Weber R, Schek HJ, Blott S (1998) A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proceedings of the 24rd International Conference on Very Large Data Bases, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, VLDB '98, pp 194–205. http://dl.acm.org/citation.cfm?id=645924.671192

Winarko E, Roddick JF (2007) Armada: an algorithm for discovering richer relative temporal association rules from interval-based data. Data Knowl Eng 63(1):76–90

Wu SY, Chen YL (2007) Mining nonambiguous temporal patterns for interval-based events. IEEE Trans Knowl Data Eng 19(6):742–758

Yang X, Wang B, Li C (2008) Cost-based variable-length-gram selection for string collections to support approximate queries efficiently. In: Proceedings of the 2008 ACM SIGMOD international conference on Management of data, ACM, pp 353–364

Yi BK, Roh JW (2004) Similarity search for interval time sequences. In: International Conference on Database Systems for Advanced Applications, Springer, pp 232–243